

國立彰化師範大學資訊管理學系

人工智慧筆記

學生：S1061103 許鉅偉

指導教授：翁政雄 教授

中華民國 112 年 9 月 ~113 年 1 月

目錄

9/11 第一週	4
●課堂須知	4
●注意事項	4
●建立程式(輸入以下程式碼)	4
※小考	8
9/18 第二週	9
●課堂重點	9
●人工智慧介紹	9
●監督式學習/非監督式學習	10
●重點整理	10
●提高準確率的方法	11
●步驟	11
●建立程式	11
●混亂矩陣/混淆矩陣(Confusion Matrix)	14
●混亂矩陣(項目意義)	14
●衡量指標	14
●實際操作	15
※小考：	15
9/25 第三周	16
●課堂重點	16
●臉部辨識操作	17
※小考	20
10/2 第四周	21
●神經元架構(由左至右)	21
●實際操作	22
1. 範例一	22
●演練操作(計算神經元)	22
1. 演練一(神經元 4 和 5)(由左至右計算)	23
2. 演練二(神經元 6)(由左至右計算)	24
3. 演練三(神經元 6 誤差值)(由右至左計算(逆推))	25
4. 演練四(神經元 4 和 5 誤差值) (由右至左計算(逆推))	26
5. 演練五(權重計算)	27
6. 演練六(常數項權重計算)	28
7. 統整結果	29
8. 讓結果(輸出值)接近 1 (上課操作第幾次接近 0.8)	29
●Python 演練(講解與操作)(註解每行意思)	30
※小考	32

10/9 第五週 雙十連假	32
10/16 第六周	33
●Python 演練(講解與註解)	33
1. 演練一	33
2. 演練二(類似於演練一)(把 N/Y 轉換成 0/1)	35
3. 演練三(GridSearch)(用 gridsearch 搜尋最佳參數組合，可以去修改之前模型的參數)	37
# GridSearch 重點：	38
●激活函數講解	38
※小考	40
10/23 第七周	41
●Python 演練	41
1. 演練一(DNN)	41
2. 演練二(人工智慧及其應用課程)	46
3. 演練三	49
●深度學習、深度神經網路 課程講解(Deep neural network , DNN)	50
●損失函數	53
※小考	57
10/30 第八週	58
●Python 演練	58
1. 演練一(已切割好的檔案，去做模型的訓練和測試)	58
2. 演練二(用完整檔案去做切割，且做訓練和測試，並把模型儲存下來)	61
3. 演練三(建立新的檔案，且把剛剛儲存的模型套用做訓練和測試)	64
●公佈期中考考題	65
※小考	65
11/6 第九週 期中考試	66
11/13 第十週	69
●課堂重點	69
A. 期中考成績確認	69
B. 科技論文的解說	69
C. 解說期末專題報告事項	69
11/20 第十一週	70
●Python 演練(RNN)	70
1. 台積電股票預測_1	70
2. 台積電股票_2	74
●課堂重點	78
1. RNN	78
2. RNN 的典型結構	79
3. RNN 雙向	79
4. RNN 優缺點	80

5. 課堂製作期末專題報告	80
※小考	80
11/27 第十二週	81
●課堂重點	81
1. 資料集原始資料	81
2. 產生時間序列資料講解	81
●Python 演練	85
1. 演練一	85
2. 演練二	87
3. 演練三	89
4. 講解以上程式的換行過程	91
5. 台積電股票預測_講解	92
※小考	94
12/4 第十三週	95
●汽車辨識 CNN	95
●課堂講解	102
a. RNN 複習講解	102
LSTM 講解	106
RNN 與 LSTM 的應用	108
GRU 講解	109
※小考	109
12/11 第十四週	110
●Python 演練(交叉驗證)	110
※小考	117
12/18 第十五週	118
●重點整理 1	118
●重點整理 2	118
12/25 第十六週	120
●各組期末報告	120
●課堂重點	120
2024/1/1 第十七週	120
元旦放假	120
2024/1/8 第十八週	120
總結及計算	120

9/11 第一週

●課堂須知

有證照和畢業證書沒什麼了不起，一個只是證明有讀過大學，一個只是證明你有繳費，把課程修完而已，我們需要的是解決問題的能力，所以用小考的方式驗證實力。

1. 企業想錄取人才的標準:具備解決問題的能力
2. 期末報告:整合所有教過的能力，確認自身程度與成果

●注意事項

1. 確定組員
2. 期末專題題目
3. Dataset
4. 執型 anaconda 執行檔
5. 建立虛擬環境 TF

●建立程式(輸入以下程式碼)

程式碼 1 (帕金斯症數據分析)

```
# -*- coding: utf-8 -*-
"""

Created on Mon Sep 18 08:27:30 2023

@author: Nasser
"""

#機器學習演算法
import datetime
starttime = datetime.datetime.now()
print('開始時間:',starttime)

import pandas as pd

filename = "D:\wekdata\Parkinsons.csv"
df_Data = pd.read_csv(filename)
print(filename)

features = list(df_Data.columns[:22])
```

```

X_Data = df_Data[features]
y_Data = df_Data["status"]

X_Data_dum = pd.get_dummies(X_Data)
X_Data_dum_DF = pd.DataFrame(X_Data_dum)

print("特徵個數: ",len(features))
print("",X_Data_dum_DF.shape[0])

Algorithmtime = datetime.datetime.now()
print('演算時間: ',Algorithmtime)

X=X_Data_dum_DF.to_numpy()
y=y_Data.to_numpy()

from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import KFold
import numpy as np
from pandas import DataFrame

n_splits = 3
kf = KFold(n_splits=n_splits, shuffle=True)
model= MLPClassifier(solver='lbfgs', validation_fraction=0.1,\n                      alpha=1e-5,hidden_layer_sizes=(256,256),random_state=1)

list_reoprt=list()
i=1
for train_index,val_index in kf.split(X):
    model.fit(X[train_index], y[train_index])
    pred = model.predict(X[val_index])
    print("\nCross validation #",i)
    print(confusion_matrix(y[val_index], pred))
    print(classification_report(y[val_index],pred))

    report=classification_report([val_index], pred,output_dict=True)
    df=pd.DataFrame(report).transpose()
    precision=np.array(df[4:5][["precision"]])
    recall=np.array(df[4:5][["recall"]])
    f1=np.array(df[4:5][["f1-score"]])

```

```
accuracy=np.array(df[2:3]['f1-score'])
tupleOne=[round(precision[0],4),round(recall[0],4),round(f1[0],4),round(accuracy[0],4)]
list_reoprt.append(tupleOne)

i=i+1

array_report=np.array(list_reoprt)

avg_precision=round(np.average(array_report[:,0]),4)
avg_recall=round(np.average(array_report[:,1]),4)
avg_f1_score=round(np.average(array_report[:,2]),4)
avg_accuracy=round(np.average(array_report[:,3]),4)

tupleAvg=[avg_precision,avg_recall,avg_f1_score,avg_accuracy]
list_reoprt.append(tupleAvg)

df_reoprt=DataFrame(list_reoprt)
df_reoprt.columns=("precision","recall","f1-score","accuracy")
print("計算總平均(Cross validation)於置於 最後一行(row)")
print(df_reoprt)

endtime = datetime.datetime.now()
print('開始時間: ',starttime)
print('演算時間: ',Algorithmtime)
print('結束時間: ',endtime)
print('花費時間: '.endtime - starttime)

print("輸出至檔案")
df_reoprt.to_csv("D:/df_reoprt.csv",index=True)
```

結果：

D:\wekdata\Parkinsons.csv

特徵個數： 22

195

演算時間： 2023-10-01 20:49:38.053739

Cross validation # 1

[[4 10]
 [6 45]]

	precision	recall	f1-score	support
N	0.40	0.29	0.33	14
Y	0.82	0.88	0.85	51
accuracy			0.75	65
macro avg	0.61	0.58	0.59	65
weighted avg	0.73	0.75	0.74	65

※小考

1. 何謂人工智慧(AI)、機器學習、深度學習、大數據分析，彼此的關係為何？

由大到小排列，並說明原因？

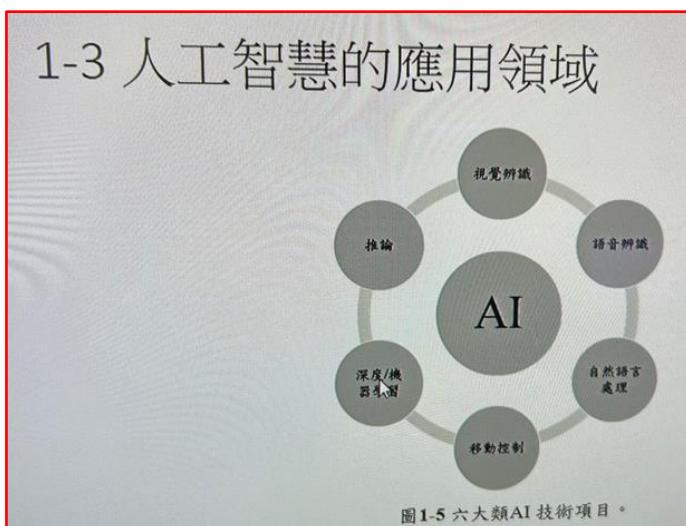
2. 今天有無任何問題？

●課堂重點

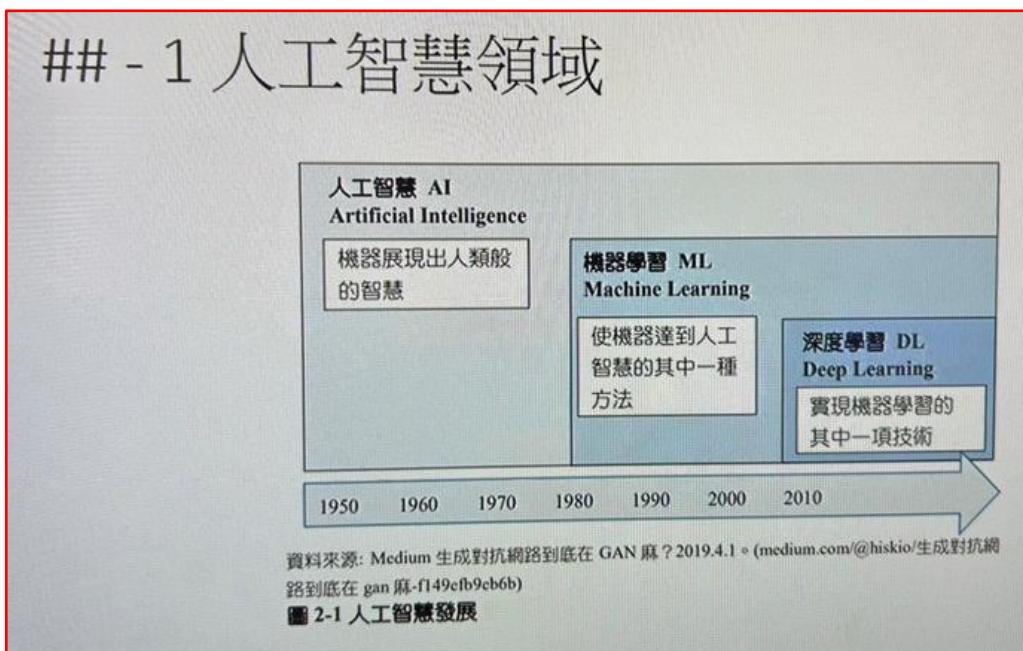
1. 人工智能的範圍廣泛，而大學期間學的人工智慧是哪一個部分？
2. 人工智能是誰給予的？
3. 公司有資料庫，能夠預設訂單、預測訂單、產能、分析資料的技術，如何用這些相關技術來幫助公司解決問題？

●人工智能介紹

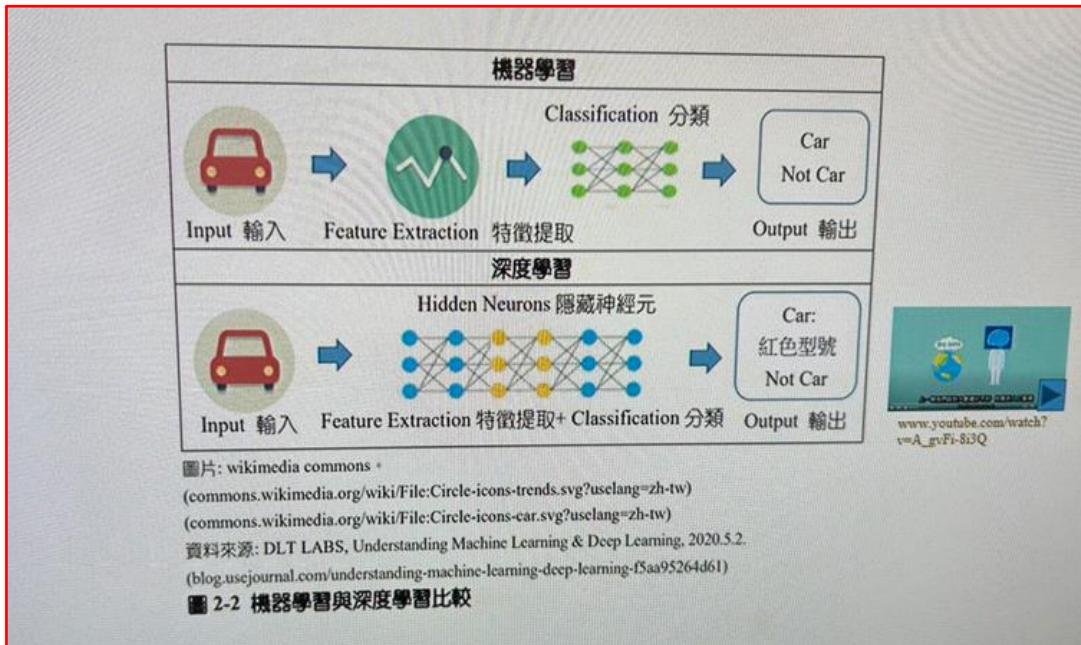
1. 人工智能領域



2. 人工智能發展圖



3. 機器學習與深度學習比較



●監督式學習/非監督式學習

類別	功能	演算法
監督式學習 (supervised)	預測 (predicting)	Linear Regression Decision Tree Random Forest Neural Network Gradient Booting Tree
	分類 (classification)	Decision Tree Naïve Bayes Logistic Regression SVM Neural Network Gradient Booting Tree
非監督式學習 (unsupervised)	分群 (clustering)	K-means
	關聯 (association)	Apriori
	降維 (dimension reduction)	PCA

●重點整理

1. 機器學習有分類演算法

2. 監督式學習是機器學習的一種，而監督式學習有一定的標準答案，又稱為二元式分類，類似是非題、對或錯
3. 必有一個欄位放輸出(預測欄位)(標籤值)(0/1)，其他欄位稱為特徵(屬性)，讓使用者來輸入。
例：預測模型(語音辨識、癌症腫瘤)
4. 分類演算法的 output 是一個類別，而不是一個值，每個輸入有對應的輸出
5. 深度學習比機器學習準(額外再去做判斷)，因為所花的時間更久，可以解決特徵學習
6. 如果在建立機器模型的時候，如何能夠確保我所選下來的特徵，是能夠提高我結果的準確率，因此機器就是幫我們分析資料、解決問題

● 提高準確率的方法

A. 特徵選擇方法：

- a. **相關性分析**：通過計算特徵和目標變數之間的相關性，可以確定哪此特徵與預測變數高度相關。
- b. **特徵重要性**：使用樹狀模型（如決策樹）可以計算每個特徵的重要性分數。
- c. **逐步選擇特徵**，從空模型開始，逐步添加最重要的特徵；或從所有特徵開始，逐步去除最不重要的特徵。
- d. 或將多個相關特徵組合成一個新的特徵，可以提供更多信息，並縮小特徵集的大小。

● 步驟

1. 如何蒐集資料(與題目相關的資料)，資料需要多大？
(資料量要越多越好，但是資料品質也要好)
2. 如何使用這些方法(決策樹、機器學習)做比較，來建立模型
3. 比對結果
#欄位數(特徵) (不一定要全部用上)，在相同的準確度時，越少越好
#準確率一樣的時候，模型越簡單越好

● 建立程式

程式碼 2

```
# -*- coding: utf-8 -*-
"""

Created on Sun Sep 24 18:04:46 2023

@author: Nasser
"""

# 機器學習演算法(Decision Tree) 範例 應用於 Parkinsons 資料集 #designed by 翁政雄 博士/
教授

import pandas as pd
```

```

df_Train = pd.read_csv("D:\wekdata\Parkinsons_Train.csv")

fnum=5
features = list(df_Train.columns[:fnum])

X_train= df_Train [features] #特徵欄位
y_train = df_Train["status"] #預測/分類的欄位

#print(y_Train)

#建立模型
#package
from sklearn.neural_network import MLPClassifier

#model = tree.DecisionTreeClassifier() model = tree.Decision TreeClassifier
(min_samples_leaf=20)

#指定參數
model = MLPClassifier(solver='lbfgs',validation_fraction=0.1,\n                      alpha=1e-5,hidden_layer_sizes=(50,50),random_state=1)
#訓練

model.fit(X_train,y_train)

#預測,評估模型好壞;使用訓練資料當測試資料 pred = model.predict(X_train)
pred = model.predict(X_train)
#輸出混亂矩陣,顯示準確率

from sklearn.metrics import confusion_matrix, classification_report
print("輸出混亂矩陣,顯示準確率:使用訓練資料")
print(confusion_matrix (y_train, pred))
print (classification_report (y_train, pred))

#預測, 評估模型好壞; 使用測試資料
df_Test = pd.read_csv("D:\wekdata\Parkinsons_Test.csv")
features = list(df_Test.columns[:fnum])
X_test= df_Test[features]
y_test = df_Test ["status"]
#預測, 評估模型好壞: 使用訓練資料當測試資料
pred = model.predict(X_test)

```

```

prob=model.predict_proba(X_test)
print("輸出混亂矩陣， 顯示準確率: 使用訓練資料")
print (confusion_matrix(y_test,pred))
print (classification_report(y_test,pred))
#模型輸出， 陽春圖形
#tree.plot_tree(mode1)
#模型輸出， 精緻圖形
#安裝 stable_windows_10_cmake_Release_x64_graphviz-instal1-2.46.1-win64
#####
import graphviz
dot_data = tree.export_graphviz(
    mode1,# (決策樹模型)
    out_file = None,
    feature_names = features, #模型中對應標籤名稱
    filled = True,
    impurity = False,
    rounded = True
    #import os
    #os.environ ["PATH"] += os.pathsep
    #os.environ["PATH"]7 +=os.pathsep
    'C:/Users/Jason/Anaconda3/pkgs/graphvi
    'C:/Program Files/Graphviz/bin
    eraph = graphviz.Source(dot_data) #選擇要可視化的 dot 數據
#####

```

結果：

```

輸出混亂矩陣， 顯示準確率: 使用訓練資料
[[ 1  2]
 [ 0 16]]
      precision    recall   f1-score   support
      N       1.00      0.33      0.50        3
      Y       0.89      1.00      0.94       16

  accuracy                           0.89        19
  macro avg       0.94      0.67      0.72        19
weighted avg       0.91      0.89      0.87        19

```

● 混亂矩陣 / 混淆矩陣(Confusion Matrix)

		Predicted Class	
		Positive	Negative
Actual Class	Positive	True-Positive(TP)	False-Negative(FN)
	Negative	False-Positive(FP)	True-Negative(TN)

● 混亂矩陣(項目意義)

* True Positive(TP) :

真陽性，實際上是正類別，而且真的被判定為正類別

* False Positive(FP) :

偽陽性，實際上是負類別，但是卻被判定為正類別

* True Negative(TN) :

真陰性，實際上是負類別，而且真的被判定為負類別

* False Negative(FN) :

偽陰性，實際上是正類別，但是卻被判定為負類別

● 衡量指標

		Predicted Class	
		Positive	Negative
Actual Class	Positive	True-Positive (TP)	False-Positive (FP)
	Negative	False-Negative (FN)	True-Negative (TN)

衡量指標

$$Sensitivity = Recall = TPR = \frac{TP}{TP + FN}$$

$$FNR = \frac{FN}{TP + FN} = 1 - TPR$$

$$Specificity = TNR = \frac{TN}{FP + TN}$$

$$FPR = \frac{FP}{FP + TN} = 1 - TNR$$

$$Precision = \frac{TP}{TP + FP}$$

$$Accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)}$$

$$F_{measure} = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

18

● 實際操作

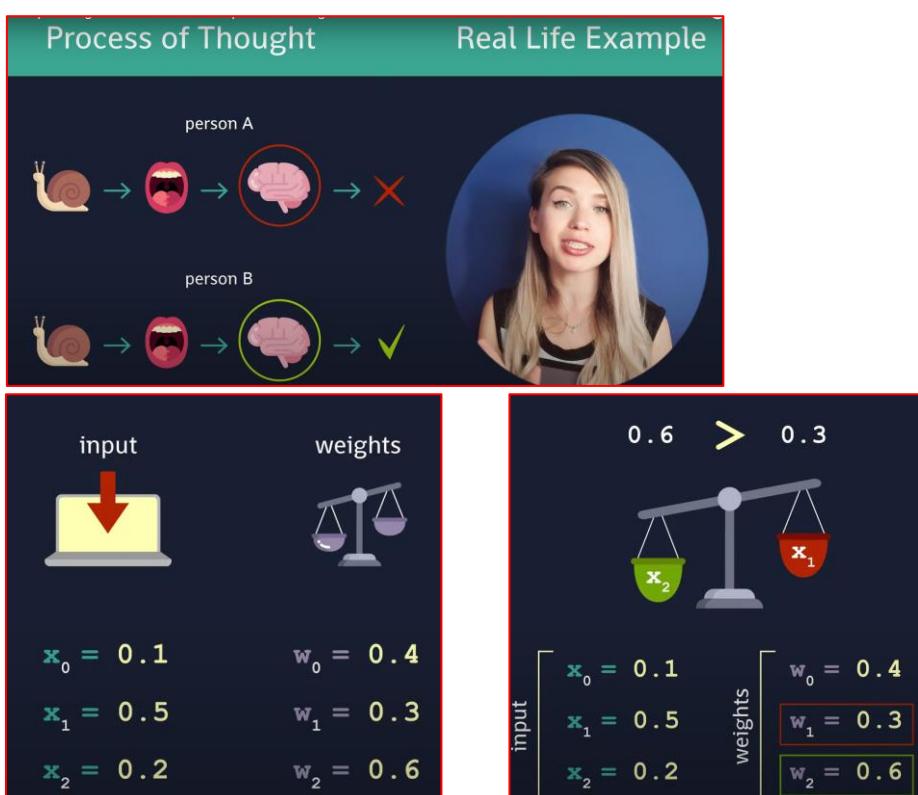
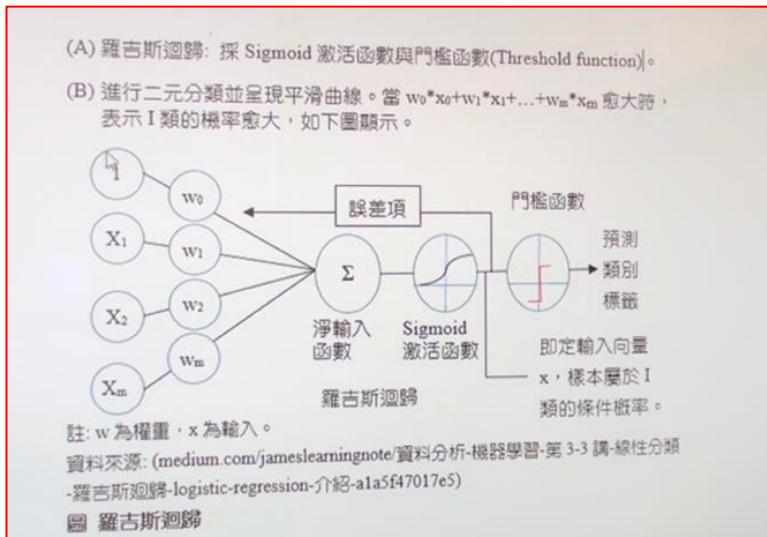
	A	B	C	D	E	F	G	H
1	N							
2		Real/Predict	N	Y		Accuracy	(TP+TN/TP+FP+FN+TN)	0.894737
3		N	TP	1	FN	2	Precision	(TP/TP+FP)
4		Y	FP	0	TN	16	Recall	(TP/TP+FN)
5								

※小考：

1. 今天上課學到了什麼？
2. 回家功課式甚麼？
3. 上課有無問題？

9/25 第三周

● 課堂重點



影片賞析(Perceptron Algorithm with Code Example - ML for beginners!)

#神經元的訓練，就是去修改它的權重，讓它的權重達到最佳化，也能讓它正確的辨識。

●臉部辨識操作

1. 建立虛擬環境

```
(TF) C:\Users\Nasser>pip install opencv_contrib_python
Collecting opencv_contrib_python
  Obtaining dependency information for opencv_contrib_python from https://files.pythonhosted.org/
  5bda09c69decc456cfb54f41d52fbef558fe91e6df7bdde6cce0/opencv_contrib_python-4.8.0.76-cp37-abi3-win_amd
  Downloading opencv_contrib_python-4.8.0.76-cp37-abi3-win_amd64.whl.metadata (20 kB)
Requirement already satisfied: numpy>=1.21.2 in c:\users\nasser\anaconda3\envs\tf\lib\site-packages \
(python) (1.23.5)
Downloading opencv_contrib_python-4.8.0.76-cp37-abi3-win_amd64.whl (44.8 MB)
    44.8/44.8 MB 12.8 MB/s eta 0:00:00
Installing collected packages: opencv_contrib_python
Successfully installed opencv_contrib_python-4.8.0.76

(TF) C:\Users\Nasser>
```

2. 先讓程式訓練照片(使用川普和蔡英文作為範例)

Face 程式碼

```
import cv2
import numpy as np
#detector = cv2.CascadeClassifier('xml/haarcascade_frontalface_default.xml') # 載入人臉追蹤
#model
#filename='C:/Users/Jason/anaconda3/envs/TF/Lib/site-
#packages/cv2/data/haarcascade_frontalface_default.xml'

#filename='C:/Users/Jason/.conda/envs/TF/Lib/site-
#packages/cv2/data/haarcascade_frontalface_default.xml'
#filename='D:/PythonCode/Opencv/haarcascade_frontalface_default.xml'
filename='haarcascade_frontalface_default.xml'
detector = cv2.CascadeClassifier(filename) # 載入人臉追蹤模型

recog = cv2.face.LBPHFaceRecognizer_create()      # 啟用訓練人臉模型方法
faces = [] # 儲存人臉位置大小的串列
ids = [] # 記錄該人臉 id 的串列

#注意圖片張數
for i in range(1,11):
    #img = cv2.imread(f'face01/{i}.jpg')           # 依序開啟每一張蔡英文的照片
    filename=f'C:/Users/Nasser/Desktop/大三上/人工智慧/課堂練習/Face/face1/{i}.jpg'

    print(filename)
    img = cv2.imread(filename)                      # 依序開啟每一張蔡英文的照片

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 色彩轉換成黑白
```

```

img_np = np.array(gray,'uint8')           # 轉換成指定編碼的 numpy 陣列
face = detector.detectMultiScale(gray)     # 擷取人臉區域
for(x,y,w,h) in face:
    faces.append(img_np[y:y+h,x:x+w])      # 記錄蔡英文人臉的位置和大小內像素
的數值
    ids.append(1)                          # 記錄蔡英文人臉對應的 id, 只能是
整數, 都是 1 表示蔡英文的 id 為 1

#注意圖片張數
for i in range(1,3):
    #img = cv2.imread(f'face02/{i}.jpg')        # 依序開啟每一張川普的照片
    filename=f'C:/Users/Nasser/Desktop/大三上/人工智慧/課堂練習/Face/face2/{i}.jpg'
    print(filename)
    img = cv2.imread(filename)                  # 依序開啟每一張川普的照片
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 色彩轉換成黑白
    img_np = np.array(gray,'uint8')            # 轉換成指定編碼的 numpy 陣列
    face = detector.detectMultiScale(gray)      # 擷取人臉區域
    for(x,y,w,h) in face:
        faces.append(img_np[y:y+h,x:x+w])      # 記錄川普人臉的位置和大小內像素的
數值
        ids.append(2)                          # 記錄川普人臉對應的 id, 只能是整
數, 都是 2 表示川普的 id 為 2

print('training...')                      # 提示開始訓練
recog.train(faces,np.array(ids))          # 開始訓練
recog.save('face.yml')                   # 訓練完成儲存為 face.yml
print('ok!')

```

結果：

```

File ~\anaconda3\envs\TF\lib\site-
packages\spyder_kernels\py3compat.py:356 in compat_exec
  exec(code, globals, locals)

File c:\users\nasser\desktop\大三上\人工智慧\課堂練習
\face2.py:23
  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 色彩轉換成
黑白

```

3. 程式偵測臉部辨識

Detection 偵測臉部

```
import cv2

recognizer = cv2.face.LBPHFaceRecognizer_create()          # 啟用訓練人臉模型方法
recognizer.read('face.yml')                                # 讀取人臉模型檔
#cascade_path = "xml/haarcascade_frontalface_default.xml" # 載入人臉追蹤模型
cascade_path="C:\\\\Users\\\\Nasser\\anaconda3\\Lib\\site-
packages\\cv2\\data\\haarcascade_frontalface_default.xml"
#filename='C:/Users/Jason/anaconda3/envs/TF/Lib/site-
packages/cv2/data/haarcascade_frontalface_default.xml'
filename="C:\\\\Users\\\\Nasser\\anaconda3\\Lib\\site-
packages\\cv2\\data\\haarcascade_frontalface_default.xml"
face_cascade = cv2.CascadeClassifier(filename)            # 啟用人臉追蹤

cap = cv2.VideoCapture(0)                                 # 開啟攝影機
if not cap.isOpened():
    print("Cannot open camera")
    exit()
while True:
    ret, img = cap.read()
    if not ret:
        print("Cannot receive frame")
        break
    img = cv2.resize(img,(540,300))                      # 縮小尺寸，加快辨識效率
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)          # 轉換成黑白
    faces = face_cascade.detectMultiScale(gray)           # 追蹤人臉（目的在於標記出外框）

    # 建立姓名和 id 的對照表
    name = {
        '1':'Tsai',
        '2':'Trump',
        '3':'oxxostudio'
    }

    # 依序判斷每張臉屬於哪個 id
    for(x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)      # 標記人臉外框
        idnum,confidence = recognizer.predict(gray[y:y+h,x:x+w]) # 取出 id 號碼以及信心
指數 confidence
        if confidence < 10:
```

```

text = name[str(idnum)]                                # 如果信心指數小於
60, 取得對應的名字
else:
    text = '???'                                     # 不然名字就是 ???
    # 在人臉外框旁加上名字
    cv2.putText(img, text, (x,y-5),cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 2, cv2.LINE_AA)

cv2.imshow('oxxostudio', img)
if cv2.waitKey(5) == ord('q'):
    break      # 按下 q 鍵停止
cap.release()
cv2.destroyAllWindows()

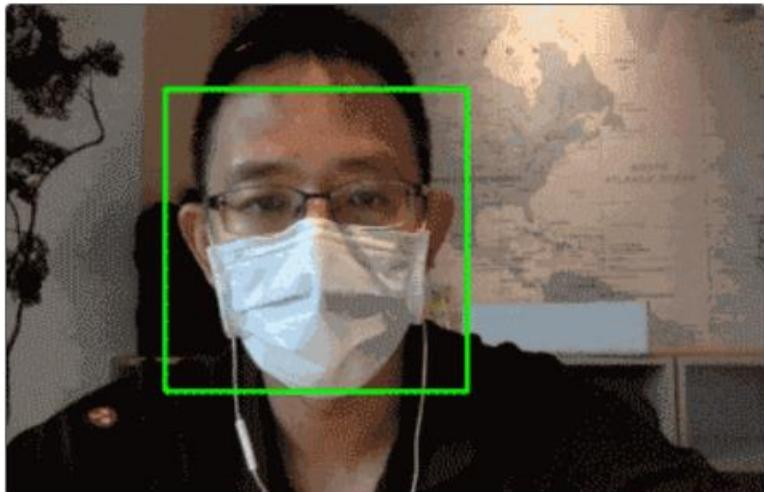
```

結果：

```

File c:\users\nasser\desktop\大三上\人工智慧\課堂練習
\detection.py:3
    recognizer.read('face.
# 讀取人臉模型檔

```



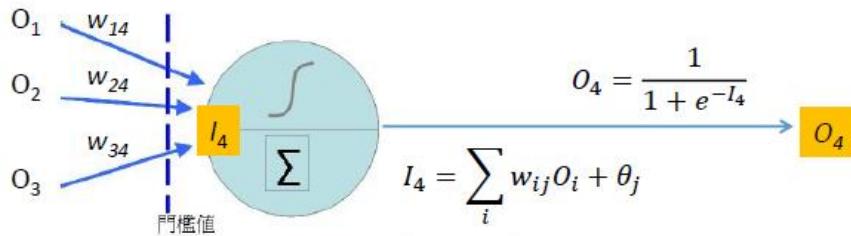
※小考

1. 回家練習(作業)是什麼？
2. 蔡英文和川普各需要幾張訓練的照片？成功的話，可以辨識幾張照片？準確率是多少？模型的訓練如何？
3. 今天上課有無任何問題？

●神經元架構(由左至右)

神經元架構

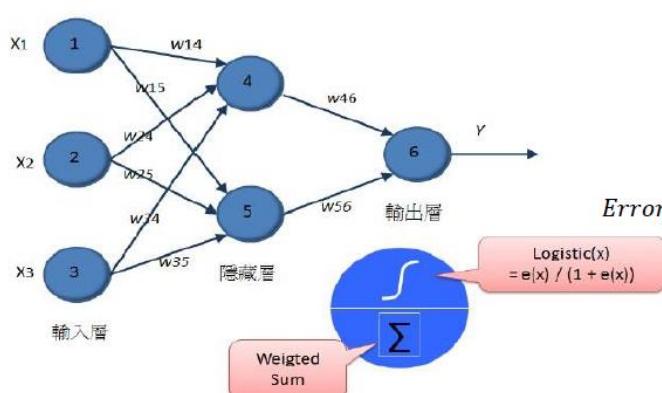
$$O_i \quad I_j = \sum_i w_{ij} O_i + \theta_j \quad I_j \quad o_j = \frac{1}{1 + e^{-I_j}} \quad O_j$$



1. 0 : 代表 output
2. i or j : 代表第幾個神經元

●計算公式

公式



$$I_j = \sum_i w_{ij} O_i + \theta_j$$

$$o_j = \frac{1}{1 + e^{-I_j}}$$

$$Error_j = o_j(1 - o_j)(T - o_j)$$

$$Error_j = o_j(1 - o_j) \sum_k Error_k \times w_{jk}$$

$$\Delta w_{ij} = l \times Error_j \times O_i$$

$$w_{ij} = w_{ij} + \Delta w_{ij}$$

$$\Delta \theta_j = l \times Error_j$$

$$\theta_j = \theta_j + \Delta \theta_j$$

公式簡化

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

$$\begin{array}{l} \text{---} \rightarrow \\ 0 \rightarrow \text{logit}(x) \rightarrow 1 \\ -\infty \rightarrow \end{array}$$

● 實際操作

1. 範例一

Excel 操作

	A	B
1	100	1
2	0	0.5

A 欄為原始值

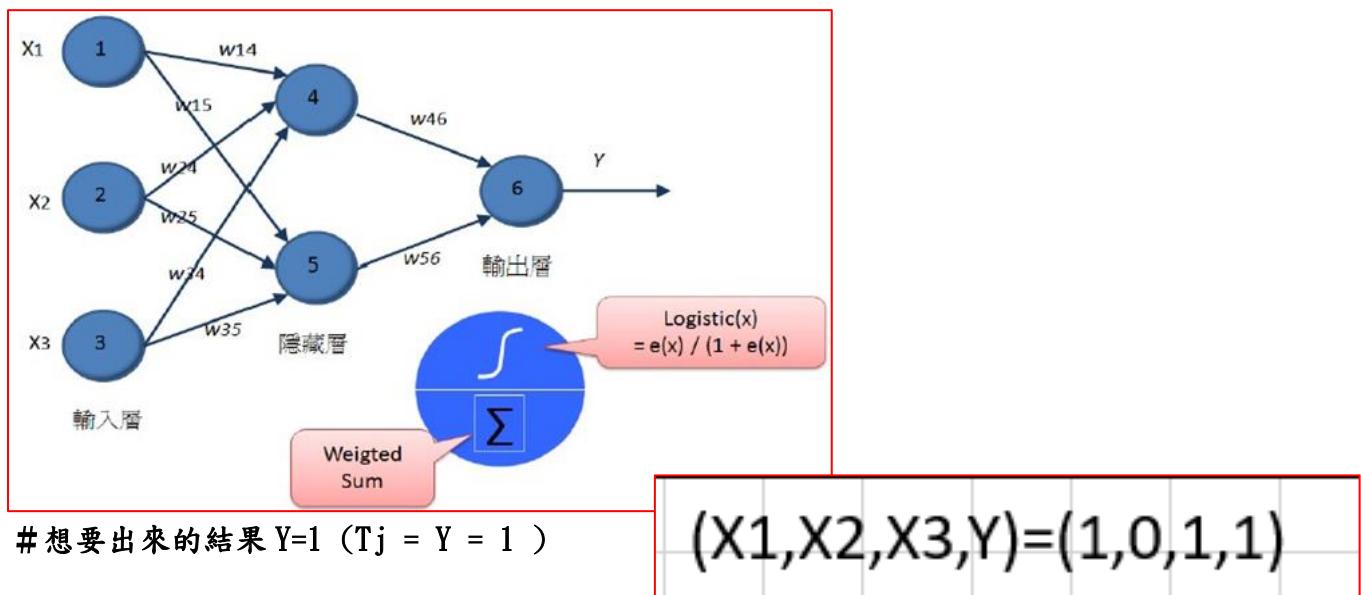
B 欄為計算

計算：

EXCEL 函數 = $1/(1+\text{EXP}(-A1))$ # EXP 為指數

類神經網路就是不斷的去修正，修正誤差值和權重，最後找出想要的結果

● 演練操作(計算神經元)



w14	w15	w24	w25	w34	w35	w46	w56	θ4	θ5	θ6	
0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1	原始
											R1

1. 演練一(神經元 4 和 5)(由左至右計算)

圖示：

B	C	D	E	F
神經元	X1	X2	X3	
X		1	0	1
w		0.2	0.4	-0.5
X*w	X1*w14	0.2	0	-0.5
SUM		-0.7	圖下	
Logistic(x)	0.331812228	圖上		

sum=sum(C4:F4)=(x*w)那行的加總

log(x)=1/(1+EXP(-C5)) C5=sum=-0.7

以此類推……

結果：

神經元 4 和 5 的計算結果

A	B	C	D	E	F
1	神經元	4			
2	X		1	0	1 Θ4
3	w		0.2	0.4	-0.5
4	X*w		0.2	0	-0.5 -0.4
5 I4	SUM	-0.7	圖下		
6 O4	Logistic(x)	0.331812228	圖上		
7					
8	神經元	5			
9	X		1	0	1 Θ5
10	w		-0.3	0.1	0.2
11	X*w		-0.3	0	0.2 0.2
12 I5	SUM	0.1			
13 O5	Logistic(x)	0.524979187			
14					

2. 演練二(神經元 6)(由左至右計算)

圖示

神經元4 的 $\log(x)$ 神經元5 的 $\log(x)$

神經元	6			
x	0.3318	0.5245	Θ_6	
w	w46 -0.3	-0.2	w56	
x*w	-0.09954	-0.105	0.1	
SUM	-0.10444			
Logistic(x)	0.473913708			

相乘數值

解：

$\text{sum} = (x*w)$ 那行的加總 = $\text{SUM}(C18:F18)$

$\log(x) = \frac{1}{1 + \text{EXP}(-\text{C19})}$ C19 = -0.10444

結果：

14					
15	神經元	6			
16	x	0.3318	0.5245	Θ_6	
17	w	-0.3	-0.2		
18	x*w	-0.09954	-0.105	0.1	
19	I6	SUM	-0.10444		
20	O6 (Y)	Logistic(x)	0.473913708		
21					

3. 演練三(神經元 6 誤差值)(由右至左計算(逆推))

圖解

		神經元4 的 log(x)		神經元5 的 log(x)	
相乘數值	神經元	6			
	x	0.3318	0.5245	Θ_6	
	w	w ₄₆ -0.3	-0.2	w ₅₆	
	x*w	-0.09954	-0.105	0.1	
	SUM	-0.10444			
Logistic(x)		0.473913708			
$Error_j = O_j \times (1-O_j) \times (T_j - O_j)$			神經元 6 誤差值		
使用輸出層的公式 T _j = Y = 1 想要的結果=1			O _j	0.47391	
			1-O _j	0.52609	
			T _j -O _j	0.52609	(T _j) 1 - O _j
			Error _j	0.13116	相乘

結果

神經元	6	誤差值
O _j	0.47391	
1-O _j	0.52609	
T _j -O _j	0.52609	
Error _j	0.13116	
(輸出層公式)		

4. 演練四(神經元 4 和 5 誤差值) (由右至左計算(逆推))

圖解

B	C	D	E	F	H	I	J
神經元	4				神經元	4 誤差值	
x	1	0	1	Θ_4	O_j	0.33181	
w	0.2	0.4	-0.5		$1-O_j$	0.66819	
$x \cdot w$	0.2	0	-0.5	0.4	$Error_k \cdot W_{jk}$	-0.03935 j=4,k=6	Error 6=0.13116(神經元6誤差)*W46(-0.3)
SUM	-0.7	圖下			$Error_j$	-0.00872	
Logistic(x)	0.331812228	圖上			(使用隱藏層公式)		
神經元	5				神經元	5 誤差值	
x	1	0	1	Θ_5	O_j	0.52498	
w	-0.3	0.1	0.2		$1-O_j$	0.47502	
$x \cdot w$	-0.3	0	0.2	-0.2	$Error_k \cdot W_{jk}$	-0.02623 j=5,k=6	Error 6=0.13116(神經元6誤差)*W56(-0.2)
SUM	0.1				$Error_j$	-0.00654	
Logistic(x)	0.524979187				(使用隱藏層公式)		
$Error_j = O_j \times (1 - O_j) \times \sum_k Error_k \times w_{jk}$ 使用隱藏層公式							

結果：

H	I	J
神經元	4 誤差值	
O_j	0.33181	
$1-O_j$	0.66819	
$Error_k \cdot W_{jk}$	-0.03935 j=4,k=6	
$Error_j$	-0.00872	
(使用隱藏層公式)		
神經元	5 誤差值	
O_j	0.52498	
$1-O_j$	0.47502	
$Error_k \cdot W_{jk}$	-0.02623 j=5,k=6	
$Error_j$	-0.00654	
(使用隱藏層公式)		

5. 演練五(權重計算)

圖解

$$\Delta W_{ij} = l \times Error_j \times O_i$$

$$\Delta W = \text{learing rate} * Error_j * O_i$$

Error6 Error5

權重	W(R)	W	ΔW	J	Errorj	Oi	
w56	-0.138028	-0.2	0.06197	0.9	0.13116	0.52498	O4數值
w46	-0.26083	-0.3	0.03917	0.9	0.13116	0.33181	O5數值
w35	0.1941124	0.2	-0.00589	0.9	-0.00654	1	X3
w25	0.1	0.1	0	0.9	-0.00654	0	X2
w15	-0.305888	-0.3	-0.00589	0.9	-0.00654	1	X1
w34	-0.507852	-0.5	-0.00785	0.9	-0.00872	1	X3
w24	0.4	0.4	0	0.9	-0.00872	0	X2
w14	0.1921482	0.2	-0.00785	0.9	-0.00872	1	X1
	修正後	原始	修正量		Error4		

$$W(R) = W + \Delta W \quad W = \text{對應原始權重}$$

Learning rate=0.9~1

結果

L	M	N	O	P	Q	R
權重	W(R)	W	ΔW	J	Errorj	Oi
w56	-0.138028	-0.2	0.06197	0.9	0.13116	0.52498
w46	-0.26083	-0.3	0.03917	0.9	0.13116	0.33181
w35	0.1941124	0.2	-0.00589	0.9	-0.00654	1
w25	0.1	0.1	0	0.9	-0.00654	0
w15	-0.305888	-0.3	-0.00589	0.9	-0.00654	1
w34	-0.507852	-0.5	-0.00785	0.9	-0.00872	1
w24	0.4	0.4	0	0.9	-0.00872	0
w14	0.1921482	0.2	-0.00785	0.9	-0.00872	1
	修正後	原始	修正量			

6. 演練六(常數項權重計算)

圖解

Θ_4	Θ_5	Θ_6	原始
-0.4	0.2	0.1	

對應

$$\Delta\theta_j = l \times Error_j$$

兩數相乘

常數項	$\Theta(R)$	Θ	$\Delta\Theta$	l	Errorj	
Θ_6	0.2180472	0.1	0.11805	0.9	0.13116	Error6
Θ_5	0.1941124	0.2	-0.00589	0.9	-0.00654	Error5
Θ_4	-0.407852	-0.4	-0.00785	0.9	-0.00872	Error4

$\Theta(R) = \Theta + \Delta\Theta$

Learning rate=0.9~1

結果：

常數項	$\Theta(R)$	Θ	$\Delta\Theta$	l	Errorj
Θ_6	0.2180472	0.1	0.11805	0.9	0.13116
Θ_5	0.1941124	0.2	-0.00589	0.9	-0.00654
Θ_4	-0.407852	-0.4	-0.00785	0.9	-0.00872

7. 統整結果

圖解

L	W(R)	W
w56	-0.138028	
w46	-0.26083	
w35	0.1941124	
w25	0.1	
w15	-0.305888	
w34	-0.507852	
w24	0.4	
w14	0.1921482	

S	T	U	V	W	X	Y	Z	AA	AB	AC	AD
w14	w15	w24	w25	w34	w35	w46	w56	Θ4	Θ5	Θ6	
0.2	-0	0.4	0.1	-1	0.2	-0.3	-0.2	-0.4	0.2	0.1	原始

0.19	-0.3	0.4	0.1	-0.5	0.19	-0.3	-0.14	-0.41	0.2	0.2	R1

結果：

w14	w15	w24	w25	w34	w35	w46	w56	Θ4	Θ5	Θ6	
0.2	-0	0.4	0.1	-1	0.2	-0.3	-0.2	-0.4	0.2	0.1	原始
0.19	-0.3	0.4	0.1	-0.5	0.19	-0.3	-0.14	-0.41	0.2	0.2	R1

8. 讓結果(輸出值)接近 1 (上課操作第幾次接近 0.8)

圖解

S	I	U	V	W	X	Y	Z	AA	AB	AC	AD
w14	w15	w24	w25	w34	w35	w46	w56	Θ4	Θ5	Θ6	
0.2	-0	0.4	0.1	-1	0.2	0.1	0.4	-0.4	0.2	1.2	原始

神經元	6	0.17	-0.3	0.4	0.1	-0.5	0.22	0.06	0.39	-0.43	0.2	1.2	R1
x		0.3318	0.5245	Θ6									
w		0.050439008	0.3763										
x*w		0.016735663	0.1974		1.2								
SUM		1.418662903											
Logistic(x)		0.805128716											

複製貼上 18 次 接近 0.8

●Python 演練(講解與操作)(註解每行意思)

```
# -*- coding: utf-8 -*-
"""

Created on Mon Oct  2 11:18:29 2023

@author: Nasser
"""

import pandas as pd
df_Train = pd.read_csv("D:\wekdata\Parkinsons_Train.csv")

features = list(df_Train.columns[22])
x_train= df_Train[features]      #特徵取了 22 個 特徵屬性
y_train= df_Train["status"]       #標籤屬性
print(x_train)                  #輸出 訓練資料集的特徵屬性值
print(y_train)                  #輸出 訓練資料集的標籤屬性值
print(df_Train)                 #輸出 訓練資料集 全部資料

#載入 package
from sklearn.neural_network import MLPClassifier
#建立模型
model = MLPClassifier(solver='lbfgs', alpha=1e-5,
                      hidden_layer_sizes=(5,3,3), random_state=2)
#訓練
model.fit(x_train,y_train)

#預測，評估模型好壞；使用訓練資料當測試資料
pred = model.predict(x_train)

#輸出混亂矩陣，顯示準確率
from sklearn.metrics import confusion_matrix, classification_report
print("輸出魂論矩陣，顯示準確率： 使用訓練資料")
print(confusion_matrix(y_train,pred))
print(classification_report(y_train,pred))

#將預測結果與真實資料 合併成 DataFrame
from pandas import DataFrame
df_PredResult=DataFrame({"Real":y_train,"Pred":pred})
print(df_PredResult)
```

```

df_PredResult["Res"] = 0
df_PredResult.loc[df_PredResult["Real"] == df_PredResult["Pred"], ["Res"]] = 1

print(df_PredResult)

```

#輸出至檔案

```

print("輸出至檔案")
df_PredResult.to_csv("D:\wekdata\Parkinsons_TrainDNN.csv", index=True)

```

結果：

輸出魂論矩陣，顯示準確率： 使用訓練資料

		precision	recall	f1-score	support
	N	0.00	0.00	0.00	45
	Y	0.74	1.00	0.85	131
accuracy				0.74	176
macro avg	0.37	0.50	0.43	176	
weighted avg	0.55	0.74	0.64	176	

	Real	Pred	
0	Y	Y	
1	Y	Y	
2	Y	Y	
3	Y	Y	
4	N	Y	
..	
171	Y	Y	
172	Y	Y	
173	N	Y	
174	Y	Y	
175	N	Y	

	Real	Pred	Res
0	Y	Y	1
1	Y	Y	1
2	Y	Y	1
3	Y	Y	1
4	N	Y	0
..
171	Y	Y	1
172	Y	Y	1
173	N	Y	0
174	Y	Y	1

產出新的 excel 檔案

A	B	C	D	E
1	Real	Pred	Res	
2	0 Y	Y	1	
3	1 Y	Y	1	
4	2 Y	Y	1	
5	3 Y	Y	1	
6	4 N	Y	0	
7	5 Y	Y	1	
8	6 Y	Y	1	
9	7 Y	Y	1	
10	8 Y	Y	1	
11	9 Y	Y	1	
12	10 Y	Y	1	
13	11 Y	Y	1	
14	12 Y	Y	1	
15	13 Y	Y	1	
16	14 Y	Y	1	
17	15 Y	Y	1	
18	16 N	Y	0	
19	17 Y	Y	1	
20	18 N	Y	0	
21	19 Y	Y	1	
22	20 Y	Y	1	
23	21 Y	Y	1	
24	22 N	Y	0	

※小考

1. 回家功課要查詢什麼資料？
2. 上課有無任何問題？

10/9 第五週 雙十連假

10/16 第六周

●Python 演練(講解與註解)

1. 演練一

延續上次上課的程式和修改，來檢測模型準確率(用權重和誤差值去修改，讓值接近 1)

```
# -*- coding: utf-8 -*-
"""

Created on Mon Oct 16 09:43:40 2023

@author: Nasser
"""

import pandas as pd
df_Train = pd.read_csv("D:\wekdata\Parkinsons_Train.csv")

features = list(df_Train.columns[:22])
X_train= df_Train[features]
y_train = df_Train["status"]

from sklearn.neural_network import MLPClassifier
model = MLPClassifier(solver='adam',
                      activation='logistic',
                      alpha=1e-5,
                      hidden_layer_sizes=(50,40),
                      random_state=1)

model.fit(X_train,y_train)

#預測，評估模型好壞，使用訓練資料當測試資料
pred = model.predict(X_train)

#輸出混亂矩陣，顯示準確率
from sklearn.metrics import confusion_matrix, classification_report
print("輸出混亂矩陣，顯示準確率： 使用訓練資料")
print(confusion_matrix(y_train,pred))
print(classification_report(y_train,pred))

#預測，評估模型好壞，使用測試資料
df_Test = pd.read_csv("D:\wekdata\Parkinsons_Test.csv")
```

```
features = list(df_Test.columns[:22])
```

```
X_test= df_Test[features]
```

```
y_test = df_Test["status"]
```

#預測，評估模型好壞，使用訓練資料當測試資料

```
pred = model.predict(X_test)
```

#輸出 預測值 N/Y(0/1) (想知道他的類別、標籤, 到底是 Y 或 N 用 predict)

```
print(pred)
```

```
prob = model.predict_proba(X_test)
```

#想知道 N 和 Y 的機率是多少，需要用到 proba 函數

```
print(prob) #輸出 預測值(N/Y)的機率值
```

#將預測結果與真實資料 合併成 DataFrame

```
from pandas import DataFrame
```

```
df=DataFrame(prob)
```

```
df.columns=["N","Y"]
```

```
df_PredResult=DataFrame({"Real":y_test,"Pred":pred,"N":df["N"],"Y":df["Y"]})
```

```
df_PredResult.to_csv("D:\wekdata\PredRes_Parkinsons.csv") #讓格式轉成 excel 檔案儲存
```

結果：

輸出混亂矩陣，顯示準確率： 使用訓練資料

```
[[ 0 45]
 [ 0 131]]
```

precision recall f1-score support

N	0.00	0.00	0.00	41
Y	0.74	1.00	0.85	133

	accuracy	macro avg	gated avg	0.74	170
accuracy	0.74	0.43	0.61	170	170
macro avg	0.37	0.50	0.74	0.43	170
gated avg	0.55	0.74	0.61	0.61	170

```
[0.23590705 0.76409295]  
[0.21282745 0.78717255]  
[0.21064847 0.78935153]  
[0.31405733 0.68594267]  
[0.29932798 0.70067202]  
[0.25240816 0.74759184]  
[0.21329323 0.78670677]  
[0.26123717 0.73876283]  
[0.21325366 0.78674634]
```

N(0)

Y(1)

A	B	C	D	E	F
1	Real	Pred	N	Y	
2	0 Y	Y	0.235907	0.764093	
3	1 Y	Y	0.212827	0.787173	
4	2 Y	Y	0.210648	0.789352	
5	3 Y	Y	0.314057	0.685943	
6	4 N	Y	0.299328	0.700672	
7	5 Y	Y	0.252408	0.747592	
8	6 Y	Y	0.213293	0.786707	
9	7 Y	Y	0.261237	0.738763	
10	8 Y	Y	0.213254	0.786746	
11	9 Y	Y	0.215425	0.784575	
12	10 Y	Y	0.280469	0.719531	
13	11 Y	Y	0.227981	0.772019	
14	12 Y	Y	0.282308	0.717692	
15	13 Y	Y	0.244031	0.755969	
16	14 Y	Y	0.224625	0.775375	
17	15 Y	Y	0.212756	0.787244	
18	16 N	Y	0.215166	0.784834	
19	17 Y	Y	0.306157	0.698843	
20	18 N	Y	0.233528	0.766472	
21	实测值	预测值			

實際值

預測值

因為 Y 的機率比較大(取最大值)， $Y > N$ 因此顯示 Y (二元分類)

2. 演練二(類似於演練一)(把 N/Y 轉換成 0/1)

```
# -*- coding: utf-8 -*-
.....
Created on Mon Oct 16 09:07:51 2023

@author: Nasser
.....



# -*- coding: utf-8 -*-
.....
Created on Mon Oct 16 08:46:18 2023

@author: Nasser
.....



import pandas as pd
df_Train = pd.read_csv("D:\wekdata\Parkinsons_Train.csv")

features = list(df_Train.columns[:22])
X_train= df_Train[features]
y_train = df_Train["status"]

from sklearn.neural_network import MLPClassifier
model = MLPClassifier(solver='lbfgs', alpha=1e-5,
                      hidden_layer_sizes=(50,50), random_state=1)

model.fit(X_train,y_train)

pred = model.predict(X_train)

from sklearn.metrics import confusion_matrix, classification_report
print("輸出混亂矩陣， 顯示準確率： 使用訓練資料")
print(confusion_matrix(y_train,pred))
print(classification_report(y_train,pred))

#預測， 評估模型好壞， 使用測試資料
df_Test = pd.read_csv("D:\wekdata\Parkinsons_Test.csv")
features = list(df_Test.columns[:22])
X_test= df_Test[features]
y_test = df_Test["status"]
```

```
#預測，評估模型好壞，使用訓練資料當測試資料
pred = model.predict(X_test) #輸出 預測值 N/Y
print(pred)
prob = model.predict_proba(X_test)
print(prob) #輸出 預測值(N/Y)的機率值

#依據機率值高低，決定 lable
import numpy as np
prob_pred = np.argmax(prob, axis=1)
#依據機率值高低，決定 lable(把機率值變成標籤值(0/1)，用 prob)
print(prob_pred)

#將預測結果與真實資料 合併成 DataFrame
from pandas import DataFrame

df=DataFrame(prob)
df.columns=["0","1"]
df_PredResult=DataFrame({"Real":y_test,"Pred":pred,"0":df["0"],"1":df["1"]})
```

結果：

```
[0.09447477 0.90552523]
[0.06240032 0.93759968]
[1 1 0 1 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1]
```

3. 演練三(GridSearch)(用 gridsearch 搜尋最佳參數組合，可以去修改之前模型的參數)

```
# -*- coding: utf-8 -*-
.....
Created on Mon Oct 16 09:16:02 2023

@author: Nasser
.....



#load dataset
import pandas as pd
df = pd.read_csv("D:\wekdata\ParkinsonsDNN.csv")
features = list(df.columns[:22])
X = df[features]
y = df["status"]

from sklearn.model_selection import GridSearchCV
parameter_space = {
    'hidden_layer_sizes':[[10,5],[20,10],[30,20],[40,30],[50,40]],
    'activation':['relu','logistic','relu'],
    'solver':['lbfgs', 'sgd', 'adam']
}

#建立模型
#package
from sklearn.neural_network import MLPClassifier
mlp=MLPClassifier(max_iter=500)
clf = GridSearchCV(mlp, parameter_space, n_jobs=-1)
clf.fit(X, y)

# Print best parameters found
print("Best parameters found: ", clf.best_params_)

# Print all results
means = clf.cv_results_['mean_test_score']
stds = clf.cv_results_['std_test_score']
params = clf.cv_results_['params']
for mean, std, params in zip(means, stds, params):
    print("%0.3f (+/-%0.03f) for %r" % (mean, std * 2, params))
```

結果：

```
In [6]: runfile('C:/Users/Nasser/Desktop/人工智慧筆記/課堂練習/1016_3 gridsearch.py', wdir='C:/Users/Nasser/Desktop/人工智慧筆記/課堂練習')
C:\Users\Nasser\anaconda3\envs\TF\lib\site-packages\sklearn\model_selection\_validation.py:425: FitFailedWarning:
10 fits failed out of a total of 50.
```

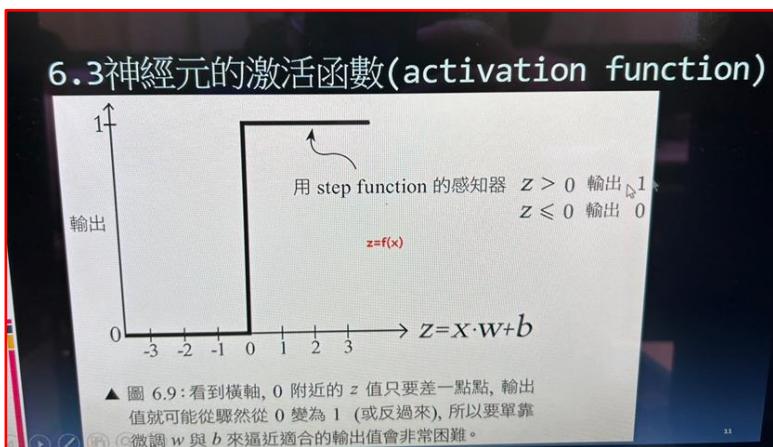
```
Best parameters found:
{'activation': 'logistic', 'hidden_layer_sizes': [50, 40]}
```

#GridSearch 重點：

1. 搜尋 gridsearch 找出最佳參數組合，可以去修改之前模型的參數
2. GridSearchCV 實作了「擬合」和「評分」方法。它包含了「score_samples」、「predict」、「predict_proba」、「decision_function」、「transform」和「inverse_transform」
3. 用以上這些方法估計的參數，透過參數網格上的交叉驗證來搜尋和優化。
4. 在所有候選的參數中，透過循環測試，嘗試每一種可能性，表現最好的參數就是最終的結果
例：參數 a 有 5 種可能，參數 b 有 8 種可能，把所有可能性列出來，可以表示 5×8 的表格，其中每個 cell 就是一個網格，循環過程就像是透過在每個網格測試、搜索，因此叫做 grid search

●激活函數講解

1.



2.

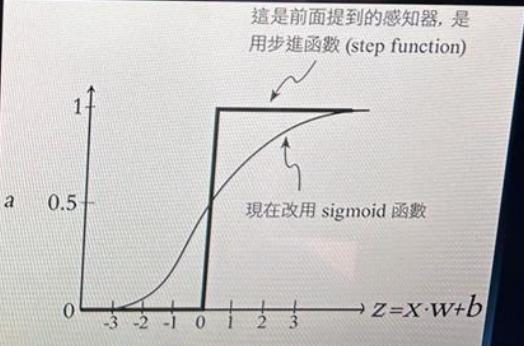
6.3.1 sigmoid 激活函數

- z 為 $x \cdot w + b$ 算出來的值。
- e 為自然常數。

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

```
from math import e  
  
def sigmoid(z):  
    return 1/(1+e**-z)  
  
sigmoid(.00001)
```

0.5000024999999999



這是前面提到的感知器, 是用步進函數 (step function)

現在改用 sigmoid 函數

a

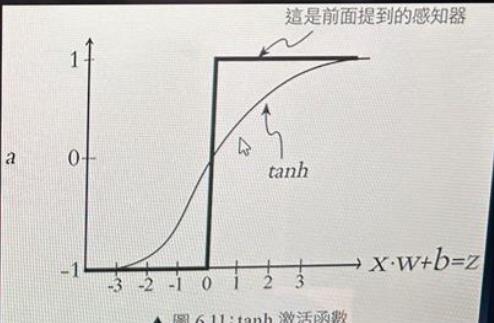
$z = X \cdot W + b$

▲ 圖 6.10 : sigmoid 函數的圖形

3.

6.3.2 tanh 激活函數

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



這是前面提到的感知器

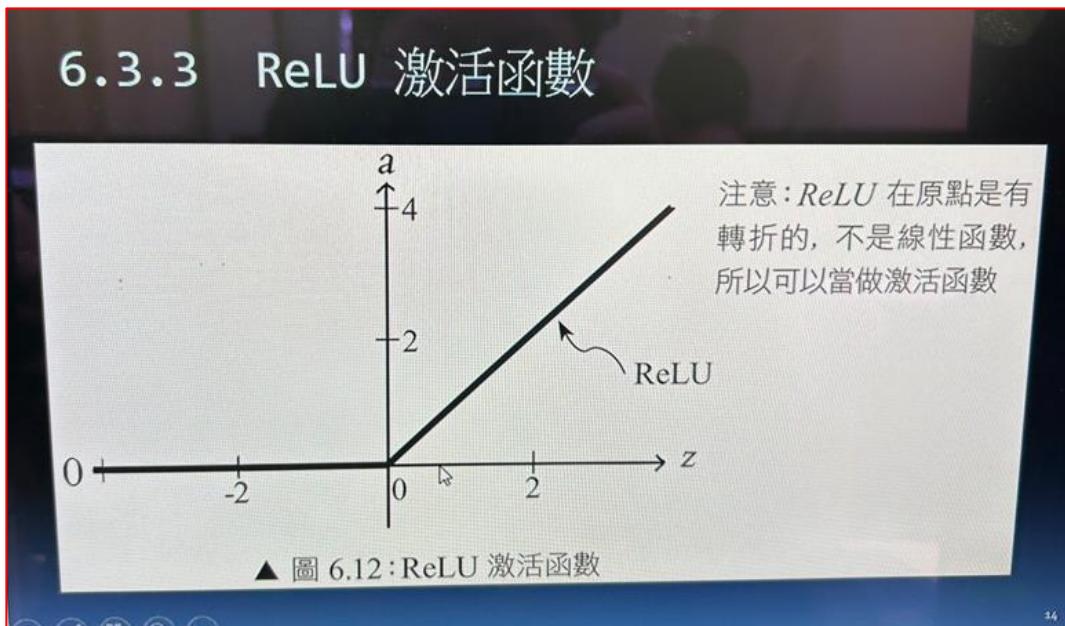
a

$x \cdot w + b = z$

\tanh

▲ 圖 6.11 : tanh 激活函數

4.



5. 激活函數的選擇

- * 感知器
- * sigmoid
- * Tanh
- * ReLU
- * Tf. Keras 提供了 leakyReLU、參數化 ReLU(parametric ReLU)…等「進階」激活函數

※小考

1. 何謂 GridSearch ?

2. 何謂 MLP 中，solver, activation 之用途為何？

Activation function for the hidden layer.

有{' identity', ' logistic', ' tanh', ' relu'}, 預設通常='relu'
(隱藏層的激活函數)

The solver for weight optimization.

(用於解決權重的最佳化)

有{' lbfsgs', ' sgd', ' adam'}, 預設通常='adam'

在訓練時間和驗證分數而言，「adam」在相對較大的資料集（具有數千個或更多訓練樣本）上運作得很好。對於小型資料集，「lbfsgs」可以收斂得更快且效能更好。

3. 今天上課有無任何問題？

●Python 演練

1. 演練一(DNN)

```
# -*- coding: utf-8 -*-
"""

Created on Mon Oct 23 10:02:11 2023

@author: Nasser
"""

#機器學習演算法 (DNN) 範例 應用於 Parkinsons 資料集
#designed by 翁政雄 博士/教授

import numpy as np
import pandas as pd
df_Train = pd.read_csv("D:\wekdata\Parkinsons_TrainDNN.csv")#所有欄位必須為數字型態
#建立 model：告訴資料 X、Y(有標籤值) EX.這是一部手機/ EX.這不是一部手機
features = list(df_Train.columns[:22])
X_train= df_Train[features]
y_train = df_Train["status"]
#print(y_train)

''' #由此開始與 MLP 不一樣
#建立模型
#package
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(256, input_dim=22, activation='relu'))

#model.add(Dense(32,activation='relu'))
#model.add(Dense(256,activation='relu'))
#from keras.layers import Dropout
#model.add(Dropout(0.2))
#model.add(Dense(9, activation='relu'))
model.add(Dense(1,activation='sigmoid')) #輸出層依照需要之輸出結果(EX.二元)所需之激活函數不一樣
```

```

# compile the keras model
#model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.compile(loss='binary_crossentropy', optimizer='sgd', metrics=['accuracy'])
#優化器

#訓練
model.fit(X_train, y_train, validation_split=0.1, epochs=15, batch_size=5, verbose=2)

''' '' #由此結束與 MLP 不一樣

#預測,評估模型好壞;使用訓練資料當測試資料
#pred = model.predict_classes(X_train, verbose=0)
#pred = model.predict_classes(X_train)

pred = model.predict(X_train)
pred = np.argmax(pred, axis=-1) #依據機率值高低決定 label

#輸出混亂矩陣,顯示準確率
from sklearn.metrics import confusion_matrix, classification_report
print("輸出混亂矩陣,顯示準確率:使用訓練資料")
print(confusion_matrix(y_train, pred))
print(classification_report(y_train, pred))

#預測,評估模型好壞;使用測試資料
df_Test = pd.read_csv("D:/wekdata/Parkinsons_TestDNN.csv")
#給資料讓建立好的 model 預測結果 (給 X 求 Y) EX.這是不是一部手機
features= list(df_Test.columns[:22])
X_test= df_Test[features]
y_test = df_Test["status"]

#預測,評估模型好壞;使用訓練資料當測試資料
#pred = model.predict_classes(X_test)
pred = model.predict(X_test)
pred = np.argmax(pred, axis=-1) #依據機率值高低決定 label

print("輸出混亂矩陣,顯示準確率:使用測試資料")
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))

```

```

#將預測結果與真實資料 合併成 DataFrame
from pandas import DataFrame
#df_PredResult=DataFrame({"Pred": pred, "Real":y_test})
#因為 DNN 的 pred = model.predict_classes(X_test) 的輸出非一維
df_PredResult=DataFrame({"Pred": pred, "Real":y_test})
#print("DataFrame: Pred, Real")
print(df_PredResult)

print(model.summary()) #



#將預測結果與真實資料 合併成 DataFrame
from pandas import DataFrame
#將 predicted_Test 轉成 dataframe
#df_PredProb=DataFrame(model.predict(X_test))
df_PredProb=DataFrame(pred)
#print(df_Predprob)

#axis=1 水平合併
df_myResult=pd.concat([df_PredResult, df_PredProb], axis=1)
#print(df_myResult)

#Compute ROC curve and ROC area for each class
#(y_test, pred) 必須是[0,1]轉換 ; N->0, Y->1
#df=DataFrame({"Pred": pred, "Real":y_test})
#df["PredNew"]=0
#df.loc[df["Pred"] == "Y", ["PredNew"]] = 1
#df["RealNew"]=0
#df.loc[df["Real"] == "Y", ["RealNew"]] = 1

#Compute ROC curve and ROC area for each class
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

fpr, tpr, threshold =roc_curve (y_test, pred) #計算真正率&假正率
roc_auc = auc (fpr, tpr) #計算 auc 的值
plt.figure()
lw = 2
#plt.figure(figsize=(10,10))
plt.plot(fpr, tpr, color='darkorange',
         lw=lw, label='ROC curve (area = %0.4f)' % roc_auc)

```

```

#假正率為橫坐標，真正率為縱座標做曲線
plt.plot([0,1], [0,1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')

#plt.legend(loc="lower right") #標籤位置
plt.legend()
plt.show()
print("\nROC_auc area=%f" % (roc_auc))

#lr_probs = model.predict_proba(X_test)
#print(1r_probs)
#keep probabilities for the positive outcome only
#lr_probs = lr_probs[:, 1]
# predict class values
#yhat = model.predict(X_test)
from sklearn.metrics import precision_recall_curve
#lr_precision, lr_recall, _ = precision_recall_curve(y_test, lr_probs)
lr_precision, lr_recall, _ = precision_recall_curve(y_test, pred)

from sklearn.metrics import f1_score
lr_f1, lr_auc = f1_score(y_test, pred), auc(lr_recall, lr_precision)
# summarize scores
#print("MLP (ANN): f1=%f PRC_auc area=%f" % (lr_f1, lr_auc)) # f1 乃是 label=1 的 f1

print("PRC_auc area=%f" % (lr_auc))

# plot the precision-recall curves
no_skill= len(y_test[y_test==1]) / len(y_test)

#plt.figure(figsize=(10,10))
#plt.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill')
plt.plot([0, 1], [1, 0], color='navy', lw=lw, linestyle='--')
plt.plot(lr_recall, lr_precision, color= 'darkorange',
         lw=lw, label='PRC curve (area = %0.4f)' % lr_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])

```

```
# axis labels  
plt.xlabel('Recall')  
plt.ylabel('Precision')  
plt.title('PRC Curve')  
  
# show the legend  
plt.legend()  
#show the plot  
plt.show()
```

結果：

輸出混亂矩陣，顯示準確率：使用訓練資料

```
[[ 45  0]  
 [131  0]]  
      precision    recall   f1-score   support  
  
      0          0.26      1.00      0.41       45  
      1          0.00      0.00      0.00      131  
  
accuracy                           0.26      176  
macro avg       0.13      0.50      0.20      176  
weighted avg        Model: "sequential_1"  
      precision    recall   f1-score   support  
dense_2 (Dense)      0.26      0.26      0.26       45  
dense_3 (Dense)      0.00      0.00      0.00      131  
accuracy                           0.26      176  
macro avg       0.13      0.50      0.20      176  
weighted avg
```

```
Model: "sequential_1"  
      Layer (type)           Output Shape        Param #  
=====  
      dense_2 (Dense)        (None, 256)        5888  
      dense_3 (Dense)        (None, 1)         257  
=====  
      Total params: 6,145  
      Trainable params: 6,145  
      Non-trainable params: 0
```

```
ROC_auc area=0.5000  
PRC_auc area=0.9211
```

2. 演練二(人工智慧及其應用課程)

```
# -*- coding: utf-8 -*-
"""

Created on Mon Oct 23 10:40:10 2023

@author: Nasser
"""

from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras import optimizers
from tensorflow.keras.utils import plot_model
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
import numpy as np

(X_train, y_train), (X_test, y_test) = mnist.load_data()

#print (X_train.shape)

np.set_printoptions (linewidth=np.inf)
#print (X_train[0])
#print(y_train [0:5])

#print(y_train.shape)
#y_train [0:12]
plt.figure(figsize=(5,5))
for k in range(12):
    plt.subplot(3, 4, k+1)
    plt.imshow(X_train [k], cmap='gray')
plt.tight_layout()
plt.show()

#5.2.4 資料預處理
X_train = X_train.reshape(60000, 784).astype('float32')
X_test = X_test.reshape(10000, 784).astype('float32')
X_train /= 255
X_test /= 255

#print(X_train[0])
```

```

n_classes = 10

y_train = to_categorical(y_train, n_classes)
y_test=to_categorical(y_test, n_classes)

#print (y_train[0])

#5.2.5 開始建立神經網路模型!5 行程式就搞定!

model=Sequential()

model.add(Dense (64, activation='relu', input_shape=(784, )))

model.add(Dense (10, activation='softmax'))
model.compile(loss='mean_squared_error',
              optimizer=optimizers.SGD(learning_rate=0.01),
              metrics=['accuracy'])

#5.2.6 訓練神經網路模型
model.fit(X_train,y_train,
           batch_size=128,
           epochs=50,
           verbose=1,
           validation_data=(X_test, y_test))
#註:由於神經網路的初始權重參數是隨機設定的,參雜了隨機性,因此底下(或您重跑一次)的結果會與書中

#-----翁老師新增部分
test_feature_normalize=X_test
test_label_onehot=y_test
#評估準確率
scores = model.evaluate(test_feature_normalize, test_label_onehot)
print('\n 準確率=', scores[1])

#預測
#prediction=model. predict_classes (test_feature_normalize) #tensorflow=-2.12.0 不支援
prediction=model.predict(test_feature_normalize)
prediction=np.argmax(prediction, axis=-1) #多元分類 依據機率值高低 決定 label

```

```

def show_images_labels_predictions (images, labels,
                                  predictions, start_id, num=10):
    plt.gcf().set_size_inches (12, 14)
    if num>25: num=25
    for i in range(0, num):
        ax=plt.subplot(5,5, 1+i)
        #顯示黑白圖片
        ax.imshow(images [start_id], cmap='binary')
        #有 AI 預測結果資料,才在標題顯示預測結果
        if(len(predictions) > 0):
            title = 'ai = ' + str(predictions[i])
            #預測正確顯示(o),錯誤顯示(x)
            title += (' (o)' if predictions[i]==labels [i] else ' (x)')
            title += '\nLabel = ' + str(labels[i])
            # 沒有 AI 預測結果資料,只在標題顯示真實數值
        else:
            title= 'Label = ' + str(labels[i])
        #X, Y 軸不顯示刻度
        ax.set_title(title, fontsize=12)
        ax.set_xticks([]); ax.set_yticks([])
        start_id+=1
    plt.show()

(X_train, y_train), (X_test, y_test)= mnist.load_data()
#顯示圖像、預測值、真實值
show_images_labels_predictions (X_test,y_test, prediction,0)

```

準確率= 0.8883000016212463
 313/313 [=====] - 0s
 669us/step

3. 演練三

```
# -*- coding: utf-8 -*-
.....
Created on Mon Oct 23 11:01:33 2023

@author: Nasser
.....



#機器學習替演算法 (DNN)範例 應用於 Parkinsons 資料集
#designed by 令政離 博士/教授

import numpy as np
import pandas as pd
df_Train = pd.read_csv("D:\wekdata\Parkinsons_TrainDNN.csv")

features = list(df_Train.columns[:22])
X_train= df_Train[features]
y_train = df_Train["status"]

#建立模型
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(256,input_dim=22,activation='relu'))

print(model.summary())
```

結果和解析：

```
Model: "sequential_3"
```

Layer (type)	Output Shape	Param #
dense_6 (Dense)	(None, 256)	5888

```
=====
Total params: 5,888
Trainable params: 5,888
Non-trainable params: 0
```

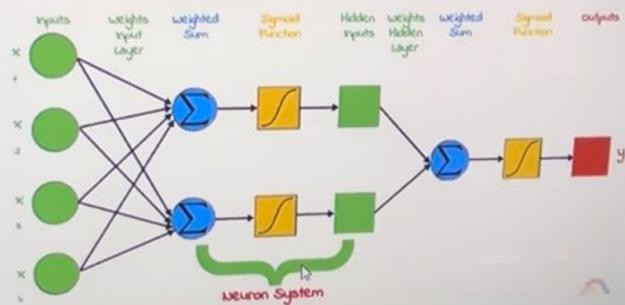
256	22	5632
		256
		5888
256	1	256+1=257 (6145)

●深度學習、深度神經網路 課程講解(Deep neural network ,DNN)

1

多層感知器(MLP) (Multi-layer Perceptron)

Neural networks: Multi-layer perceptron

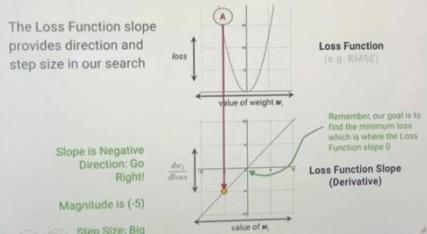


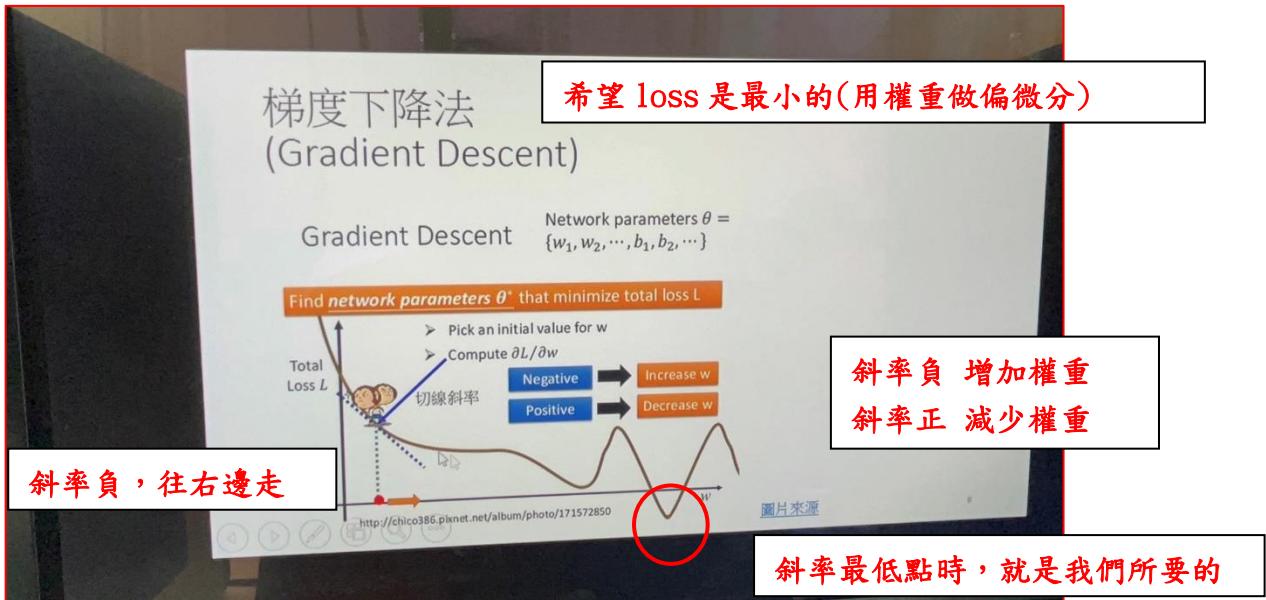
2. 梯度下降法

梯度下降法 (Gradient Descent)

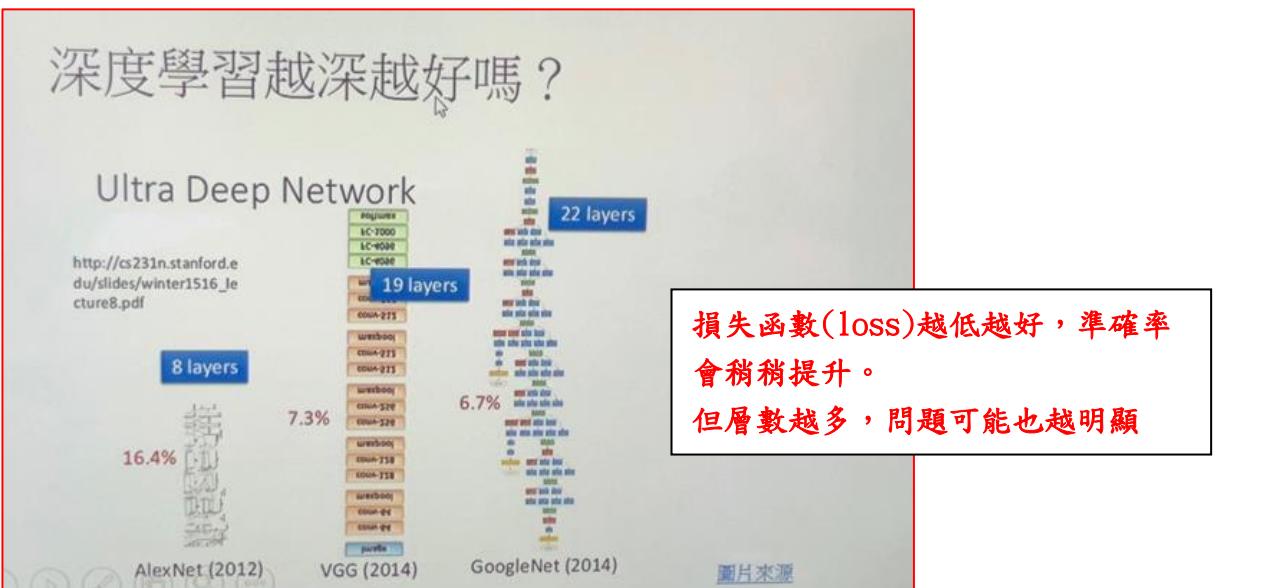
- 深度學習想像成有一百萬個學生同時在寫答案，他們每個人都有不同的思考方式，最後每個人都交出一個不同的答案(一個數字)
- 將所有的答案跟標準答案相減之後(技術上稱為 loss)，畫成一條折線圖(或是複雜一點的 3D 圖)，離標準答案最接近的那個答案，就會在這張圖的最低點，深度學習的目標就是要找到這個最低點
- 最低點代表什麼呢？代表寫出這個答案的學生，擁有最接近正確答案的思考方式，接下來我們讓其他學生向這位學生學習，並繼續測試，是否都能回答正確。理論上，隨著測試次數越多，正確率就會越高，表示這個機器已經通過測試，可以投入實戰分析了

梯度下降調整 (A 點有一個負斜率，向右尋找最小值)





3.



深度學習越深越好嗎？

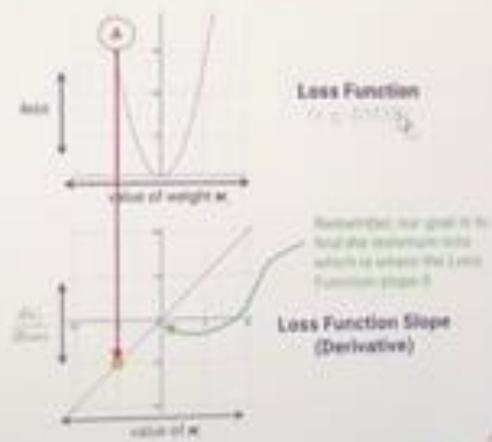
- 從錯誤率（紅字）來看，神經網路疊得越深似乎越好，但可以注意它的架構也不是簡單的堆疊，而是變得很複雜。
- 堆疊上百層的神經網路，常常會導致「Vanishing Gradient」，也就是因為每一層運算讓數值不斷收斂，導致最後的 output 越來越小，跟正確答案相減之後也就看不到顯著的最小值，看起來到處都是最小值。我們面對的可能不只一百萬個答案，很可能是千萬或上億個答案。
- 神經網路疊加的越多層，這個問題就會越明顯，因此需要設計不同的架構，跟特殊的運算過程，才能避免找不到最低點。有時候反而 layer 少一點，正確率還更高。

4.

梯度下降調整 (A 點有一個負斜率，向右尋找最小值)

The Loss Function slope provides direction and step size in our search

Slope is Negative
Direction: Go Right!
Magnitude is (5)
Step Size: 5%

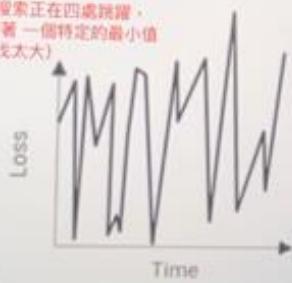


斜率正 往左邊走
斜率負 往右邊走

5.

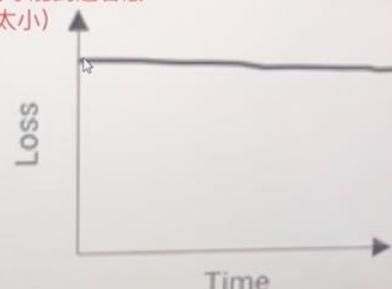
找最佳解 (調整的步伐)

這意味著我們的搜索正在四處跳躍，
而不是我們希望朝著一個特定的最小值
(步伐太大)



找最佳解 (調整的步伐)

這意味著我們可能仍在同一山谷中，
但是要很長時間才能到達谷底
(步伐太小)



6.

機器學習的評價指標(loss function) SSE、MSE、RMSE、MAE、R-Squared

- SSE(和方差)

$$SSE = \sum_{i=1}^n w_i (y_i - \hat{y}_i)^2$$

$$\frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$$

- 均方誤差(MSE)

- 均方根誤差(RMSE)

$$\sqrt{\frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2}$$

$$\frac{1}{m} \sum_{i=1}^m |y_i - \hat{y}_i|$$

- MAE(平均絕對誤差)

- R Squared

- 確定係數

- [0,1] 越大越好，解釋能力越強

$$R-square = \frac{SSR}{SST} = \frac{SST - SSE}{SST} = 1 - \frac{SSE}{SST}$$

$$SSR = \sum_{i=1}^n w_i (\hat{y}_i - \bar{y}_i)^2$$

$$SST = \sum_{i=1}^n w_i (y_i - \bar{y}_i)^2$$

$$SST = SSE + SSR$$

15

●損失函數

1.

8.1 損失函數 (loss function)

■ 8.1.1 均方誤差 (MSE) 損失函數

$$C = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (\text{公式 8.1})$$

- 將每一筆資料預測值的平方誤差加總起來。
- 除以總輸入樣本數 n。



2.

8.1.2 交叉熵 (Cross Entropy) 損失函數

$$C = -\frac{1}{n} \sum_{i=1}^n [y_i \ln \hat{y}_i + (1 - y_i) \ln(1 - \hat{y}_i)] \quad (\text{公式 8.2})$$

```
from numpy import log

def cross_entropy(y, yhat):
    return -1*(y*log(yhat) + (1-y)*log(1- yhat))
```

3.

交叉熵 (Cross Entropy) 損失函數

▼ 不同 y 與 $yhat$ 所計算出的交叉熵

y	$yhat$	交叉熵
1	0.9997	0.0003
1	0.9	0.1
1	0.6	0.5
1	0.1192	2.1
0	0.1192	0.1269

最後一列也可看到，若 y 改成 0, $yhat$ 為 0.1192，肉眼可知兩者差距算小，實際上算出的交叉熵值也就不高

4.

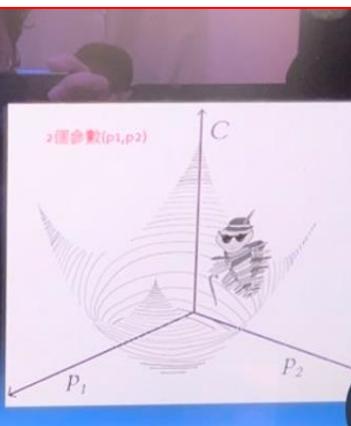
8.2 藉由訓練讓誤差值最小化

- 梯度下降法 (gradient descent)
 - C (損失值/總誤差值)，亦可用 L 表示
 - p (某權重參數)，亦可用 w 表示
 - 視「梯度(虛線)的正負」，決定「左移/右移」，到達最低點(最佳參數值)
 - 梯度($+$) \rightarrow 左移
 - 梯度($-$) \rightarrow 右移

5.

2個參數(p_1, p_2)

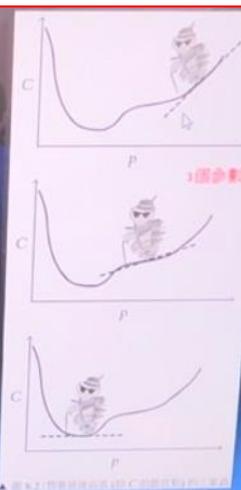
- 梯度下降法 (gradient descent)
 - C (損失值/總誤差值)，亦可用 L 表示
 - p (某權重參數)，亦可用 w 表示
 - 視「梯度(虛線)的正負」，決定「左移/右移」，到達最低點(最佳參數值)
 - 梯度($+$) \rightarrow 左移
 - 梯度($-$) \rightarrow 右移
- 使用梯度下降法，不斷移動/修正權重參數，達到最低點(最佳參數值)



6.

8.2 藉由訓練讓誤差值最小化

- 梯度下降法 (gradient descent)
 - C (損失值/總誤差值)，亦可用 L 表示
 - p (某權重參數)，亦可用 w 表示
 - 視「梯度(虛線)的正負」，決定「左移/右移」，到達最低點(最佳參數值)
 - 梯度($+$) \rightarrow 左移
 - 梯度($-$) \rightarrow 右移



7.

週期設定

- 若驗證資料(validation data)的損失值(val_loss)隨著訓練週期增加而下降(沒有上升) \rightarrow 增加訓練週期，讓損失值更低
- 若驗證資料(validation data)的損失值(val_loss)不減反增，代表訓練週期設太多了，模型已經過度適配(overfitting)
 - 模型對訓練資料做過度學習
 - 對訓練資料的預測能力很強(準)
 - 對測試資料的預測能力很弱(差)

```
▼ 節錄自 ch05-shallow_net_in
model.fit(X_train,y_train,
           batch_size=128,
           epochs=200,
           verbose=1,
           validation_data=
```

8.

8.2.4 從局部最小值(local minimum)脫離

- 局部最小值(local minimum)
- 全局最小值(global minimum)
- SGD以外的演算法
 - Momentum, Adagrad, RMSProp, Adam, ...
- SGD的可能機會
 - 若學習率夠大(步伐夠大)，有機會跨過L凹谷

9.

8.4 規劃隱藏層與各層神經元的數量

- 範例(右圖)：5個隱藏層
 - 最接近輸出層的第5個隱藏層學習速度最快！
 - 與反向傳播的運算有關
 - 靠近輸出層，損失值影響較大
 - 靠近輸入層，損失值影響較小(被稀釋)
- 梯度消失(無法修正權重)
 - 太多神經層造成前端層(靠近輸入層)的權重參數修正緩慢甚至失敗的問題

訓練前的考量 (一)：隱藏層要設幾層？

- 建議從2~4個隱藏層開始
- 架構越簡單越好(隱藏層數減少)
 - 若能解決問題(損失值接受)，神經網路架構越簡單越好，隱藏層能少則少
- 隱藏層數增加
 - 驗證資料的損失值減少，繼續增加隱藏層數

訓練前的考量(二)：各層神經元的數量要設多少？

- 若輸入資料的低階特徵（顯而易見的特徵）較多，模型前幾層(接近輸入層)放更多神經元可能會管用
- 若特徵不明顯，則後面幾層(接近輸出層)多設點神經元會較好
- 經驗法則
 - 將特定層的神經元數量 $\times 2$ 的倍數來增減， $64 \rightarrow 128, 64 \rightarrow 32$

※小考

1. 常用的 Activation function 有那些？其有缺點為何？(數學公式、優缺點)
2. 常用的 Loss function 有那些？其有缺點為何？(數學公式、優缺點)
3. 何謂 underfitting? overfitting?
4. 今天上課有無問題？

10/30 第八週

●Python 演練

1. 演練一(已切割好的檔案，去做模型的訓練和測試)

```
# -*- coding: utf-8 -*-
"""

Created on Mon Oct 30 10:00:16 2023

@author: Nasser
"""

#機器學習演算法(MLP) 範例 應用於 Parkinsons 資料集
#訓練資料集、測試資料集,不同檔案
#designed by 翁政雄 博士/教授

import pandas as pd
#讀檔案(資料集)
df_Train = pd.read_csv("D:\wekdata\Parkinsons_Train.csv")

#擷取特徵屬性
features =list(df_Train.columns[:22])

#自變數(特徵屬性)
X_Train= df_Train[features]
#應變數(預測結果)
y_Train= df_Train["status"]

#建立模型(MLP)
from sklearn.neural_network import MLPClassifier
model = MLPClassifier (solver='lbfgs', alpha=1e-5, hidden_layer_sizes= (5,3), random_state=1)

#訓練模型
model =model.fit(X_Train, y_Train)

#輸出決策樹(陽春圖形)
#tree.plot_tree (classifier)

#預測使用訓練資料(Train),進行預測
pred =model.predict(X_Train)
```

```

#輸出混亂矩陣,顯示準確率
from sklearn.metrics import classification_report
print("輸出混亂矩陣,顯示準確率:使用訓練資料")
print(classification_report(y_Train, pred))

#將預測結果與真實資料 合併成 DataFrame
from pandas import DataFrame
myPredResult=DataFrame({"Real":y_Train, "Pred": pred})

#設定預設值
myPredResult["Ans"] = 0
TPTN=0
for i in range(len(myPredResult)):
    if myPredResult.iat [i,0]==myPredResult.iat [i,1]:
        myPredResult.iat[i,2]=1
        TPTN=TPTN+1
    else:
        myPredResult.iat[i,2]=0

#輸出預測結果(train)
print("訓練資料 Accuracy_Cal=", round(TPTN/len(myPredResult), 2))

filename="D:\wekdata\Parkinsons_Train_final.csv"
print("將預測結果(Train)輸出至檔案:",filename)
myPredResult.to_csv(filename, index=True)

#####
#預測,評估模型好壞:使用測試資料
df_Test = pd.read_csv("D:\wekdata\Parkinsons_Test.csv")
#給資料讓建立好的 model 預測結果 (給 X 求 Y) EX.這是不是一部手機

#擷取特徵資料
features = list(df_Test.columns[:22])

X_test= df_Test[features]
y_test = df_Test ["status"]

#預測, 評估模型好壞: 使用測試資料當測試資料

```

```
pred = model.predict(X_test) #輸出 預測值(N/Y)

#輸出混亂矩陣,顯示準確率
from sklearn.metrics import classification_report
print("輸出混亂矩陣,顯示準確率:使用訓練資料")
print(classification_report(y_test, pred))

#將預測結果與真實資料 合併成 DataFrame
from pandas import DataFrame
myPredResult=DataFrame({"Real":y_test, "Pred": pred})

#設定預設值
myPredResult["Ans"] = 0
TPTN = 0
for i in range(len(myPredResult)):
    if myPredResult.iat[i,0]==myPredResult.iat[i,1]:
        myPredResult.iat[i,2]=1
        TPTN=TPTN+1
    else:
        myPredResult.iat[i,2]=0

#輸出預測結果(train)
print("訓練資料 Accuracy_Cal=", round(TPTN/len(myPredResult), 2))

filename="D:\wekdata\Parkinsons_Test_final.csv"
print("將預測結果(Train)輸出至檔案:", filename)
myPredResult.to_csv(filename, index=True)
```

輸出混亂矩陣，顯示準確率:使用訓練資料				
	precision	recall	f1-score	support
N	0.00	0.00	0.00	45
Y	0.74	1.00	0.85	131
accuracy			0.74	176
macro avg	0.37	0.50	0.43	176
weighted avg	0.55	0.74	0.64	176

Parkinsons_Test_final		2023/10/30 下午 09:38		Microsoft Excel 垣...	
輸出混亂矩陣，顯示準確率：使用訓練資料					
		precision	recall	f1-score	support
N	0.00	0.00	0.00	0.00	3
Y	0.84	1.00	0.91	0.91	16
accuracy				0.84	19
macro avg	0.42	0.50	0.46	0.46	19
weighted avg	0.71	0.84	0.77	0.77	19

2. 演練二(用完整檔案去做切割，且做訓練和測試，並把模型儲存下來)

```
import pandas as pd
from sklearn.neural_network import MLPClassifier
from pandas import DataFrame
from sklearn.model_selection import train_test_split

# 讀取訓練資料集
df_Train = pd.read_csv("D:\wekdata\Parkinsons.csv")

# 擷取特徵屬性和應變數
features = list(df_Train.columns[:22])
X = df_Train[features]
y = df_Train["status"]

# 切割資料集成訓練集和測試集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

# 建立 MLP 模型
model = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 3),
random_state=1)

# 訓練模型
model.fit(X_train, y_train)

# 預測使用訓練資料
pred = model.predict(X_train)

# 輸出混淆矩陣，顯示準確率
from sklearn.metrics import classification_report
print("輸出混淆矩陣，顯示準確率：使用訓練資料")
print(classification_report(y_train, pred))

# 將預測結果與真實資料合併成 DataFrame
myPredResult = DataFrame({"Real": y_train, "Pred": pred})
myPredResult["Ans"] = 0

# 計算準確率
TPTN = 0
for i in range(len(myPredResult)):
    if myPredResult.iat[i, 0] == myPredResult.iat[i, 1]:
```

```

        myPredResult.iat[i, 2] = 1
        TPTN += 1
    else:
        myPredResult.iat[i, 2] = 0

# 輸出準確率
print("訓練資料 Accuracy 計算: ", round(TPTN / len(myPredResult), 2))

# 將預測結果輸出成 CSV 檔案
filename = "D:\wekdata\MLP_Parkinsons_Test.csv"
print("將預測結果 (Train) 輸出至檔案: ", filename)
myPredResult.to_csv(filename, index=True)

#####
# 預測使用訓練資料
pred = model.predict(X_test)

# 輸出混淆矩陣，顯示準確率
from sklearn.metrics import classification_report
print("輸出混淆矩陣，顯示準確率: 使用訓練資料")
print(classification_report(y_test, pred))

# 將預測結果與真實資料合併成 DataFrame
myPredResult = DataFrame({"Real": y_test, "Pred": pred})
myPredResult["Ans"] = 0

# 計算準確率
TPTN = 0
for i in range(len(myPredResult)):
    if myPredResult.iat[i, 0] == myPredResult.iat[i, 1]:
        myPredResult.iat[i, 2] = 1
        TPTN += 1
    else:
        myPredResult.iat[i, 2] = 0

# 輸出準確率
print("訓練資料 Accuracy 計算: ", round(TPTN / len(myPredResult), 2))

# 將預測結果輸出成 CSV 檔案

```

```

filename ="D:\wekdata\MLP_Parkinsons_Test.csv"
print("將預測結果 (Train) 輸出至檔案: ", filename)
myPredResult.to_csv(filename, index=True)

from joblib import dump

# 儲存 MLP 模型
filename = "D:\wekdata\MLP_Parkinsons_Test.csv_MLP.joblib"
dump(model, filename)

print("儲存模型成功, 檔案位置:", filename)

```

結果：

輸出混淆矩陣，顯示準確率：使用訓練資料

	precision	recall	f1-score	support
N	0.00	0.00	0.00	45
Y	0.74	1.00	0.85	130
accuracy			0.74	175
macro avg	0.37	0.50	0.43	175
weighted avg	0.55	0.74	0.63	175

訓練資料Accuracy計算： 0.74

輸出混淆矩陣，顯示準確率：使用訓練資料

	precision	recall	f1-score	support
N	0.00	0.00	0.00	3
Y	0.85	1.00	0.92	17
accuracy			0.85	20
macro avg	0.42	0.50	0.46	20
weighted avg	0.72	0.85	0.78	20

訓練資料Accuracy計算： 0.85

 MLP_Parkinsons_Test	2023/10/30 下午 09:45	Microsoft Excel 逗...	1 KB
 MLP_Parkinsons_Test.csv_MLP.joblib	2023/10/30 下午 09:45	JOBLIB 檔案	7 KB

3. 演練三(建立新的檔案，且把剛剛儲存的模型套用做訓練和測試)

```
#機器學習演算法(MLP) 範例 應用於 Parkinsons 資料集
#(載入模型)+測試新資料
#designed by 翁政雄 博士/教授

#預測,使用新資料(測試),將 Parkinsons_Test 視為新資料
import pandas as pd
df_New= pd.read_csv("D:\wekdata\Parkinsons _New.csv")
features =list (df_New.columns[:22])
X_New =df_New[features]
y_New= df_New["status"]

#load model*****
from joblib import load #dump, load
filename="D:\wekdata\MLP_Parkinsons_Test.csv_MLP.joblib"
print("載入模型檔名:", filename)
model=load(filename)

#預測,評估模型好壞;使用訓練資料當測試資料
pred=model.predict(X_New)

#輸出混亂矩陣,顯示準確率
from sklearn.metrics import classification_report
print("輸出混亂矩陣,顯示準確率:使用 New 資料")
print(classification_report (y_New, pred))

#將預測結果與真實資料 合併成 DataFrame,並且判斷是否預測正確(Ans)
from pandas import DataFrame
myPredResult=DataFrame({"Real":y_New, "Pred": pred})
myPredResult["Ans"] = 0
```

結果：

```
輸出混淆矩陣，顯示準確率：使用訓練資料
precision    recall   f1-score   support
      N       0.00      0.00      0.00      45
      Y       0.74      1.00      0.85     130

accuracy                           0.74      175
macro avg       0.37      0.50      0.43      175
weighted avg     0.55      0.74      0.63      175

訓練資料Accuracy計算： 0.74
```

```
輸出混淆矩陣，顯示準確率：使用訓練資料
precision    recall   f1-score   support
      N       0.00      0.00      0.00       3
      Y       0.85      1.00      0.92      17

accuracy                           0.85      20
macro avg       0.42      0.50      0.46      20
weighted avg     0.72      0.85      0.78      20

訓練資料Accuracy計算： 0.85
```

●公佈期中考考題

112 學年度第一學期期中考試試題

一、問答題

1. (60 分) 簡答題。

- (20 分)何謂機器學習、深度學習、大數據、人工智慧？
- (20 分)常用的 Activation function 有那些？其有缺點為何？(數學公式、優缺點)(至少 4 個)。
- (20 分)常用的 Loss function 有那些？其有缺點為何？(數學公式、優缺點)(至少 4 個)。

2. (20 分) 衡量指標：計算 Precision, Recall, F1, Accuracy。(公式及計算過程)。

		預測	
		P	N
真實	P	I	
	N		

3. (20 分)類神經網路，神經元 45 之輸出值(公式及計算過程)。

※小考

1. 何謂機器學習、深度學習、大數據、人工智慧？

11/6 第九週 期中考試

1.

- 人工智慧 (AI)：人工智慧是一門含蓋多個領域的跨科學領域，它專注於如何創造智能系統，模仿人類的思考和學習能力，以解決問題。AI 包含機器學習、深度學習、自然語言等多個領域，都是實現 AI 不可或缺的技術。
- 機器學習 (Machine Learning)：機器學習是人工智慧的分支，目標是開發能夠從數據學習和改進的計算機系統，無需明確的編程。
→ 根據過去的數據經驗做出預測和決策，並不斷優化。
- 深度學習 (Deep Learning)：深度學習是機器學習的一個分支，它基於類似神經網路的結構，包含許多隱藏層，這些隱藏層允許模型自動學習多層次的特徵表示。
→ 可做圖像辨別、語言辨識等領域，得到較好的成就，實現人類水平的性能。
- 大數據 (Big Data)：指龐大且複雜的數據集，通常無法使用傳統的數據處理工具處理，而它有四項特徵：數據大量、數據多樣、數據真實性、產生速度快。

2. Activation function

• ReLU: $f(x) = \max(0, x)$

優：
(1) (隨機梯度下降法) 演算法的收斂速度比 sigmoid 和 tanh 快
(2) 不會出現梯度飽合，梯度消失的問題

缺：
(1) 輸出不是 0 均值的

(2) 可能導致 Dead ReLU Problem (神經元壞死現象)，在負數 ($x < 0$)
梯度為 0 時 → 參數永不被更新

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

優：
(1) 在輸出 (0, 1) 之間，輸出範圍有限，最佳化穩定，可做輸出層
(2) 連續函數，便於求導

缺：
(1) 輸出不是 0 均值的 → 會對梯度產生影響

(2) 當 x 取絕對值非常大的正或負值時 → 出現飽合現象，
表示函數會變得很平，並對輸入的微小改變不敏感
(容易梯度消失)

$$\bullet \text{Tanh: } f(x) = \frac{1-e^{-2x}}{1+e^{-2x}} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

優: (1) tanh 比 sigmoid 函數收斂速度更快

(2) 因輸入值壓縮到 $-1 \sim 1$ 的範圍 → 因此是 0 均值，方便
下一層網路的學習

缺: (1) 因導數較大或較小時，導數接近 0 → 造成梯度消失的問題

$$\bullet \text{Leaky Relu: } f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{if } x < 0 \end{cases}$$

優:

(1) 在輸入為負值時，給予輸入值很小的斜率，解決 Relu
在負輸入情況下的 0 梯度問題上，也解決 Dead Relu 問題

缺:

(1) 理論上 → 具有比 Relu 更好的效果，但
實際上 → 效果不穩定，故應用並不多

3. Loss function

$$\bullet \text{MSE (均方誤差)} \quad \text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

優: (1) 因是平滑函數，求解最佳化問題時，利於誤差梯度計算
(2) 誤差減小 → 梯度也減小 → 較快收斂到最小值

缺: (1) 對異常值敏感 → 降低預測結果
(2) 訓練初期不穩定 → 容易梯度爆炸

$$\bullet \text{MAE (平均絕對誤差)} \quad \text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

優: (1) 相較於 MSE，對異常值較穩定
(2) 梯度較穩定，即使損失很小，仍不變

缺:

(1) 由於微分不連續，容易在快學習到最佳解時，更新的解
進不到最佳的地方（收斂速度很慢）

$$\bullet \text{Cross Entropy (交叉熵)}$$

$$L(y, \hat{y}_i) = -\sum_{i=1}^n y_i \cdot \log(\hat{y}_i)$$

優: (1) 更好比較各個模型間的優劣，更有利於模型之後的學習
(2) 在模型較差的時候，學習速度較快；模型效果較好時，學習
速度變慢

缺: (1) 只關心對於正確標籤預測概率的準確性，
忽略了其它非正確標籤的差異
→ 導致學習到的特徵較散

• NLL (對數似然損失): $NLL = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^K y_{ij} \log(p_{ij})$

優:

(1) 最大化觀察到樣本的似然，使它能更好地擬合到觀察到的數據，提高模型的預測性能。

缺:

(1) 數學計算複雜，會占用許多資源來完成計算

4.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

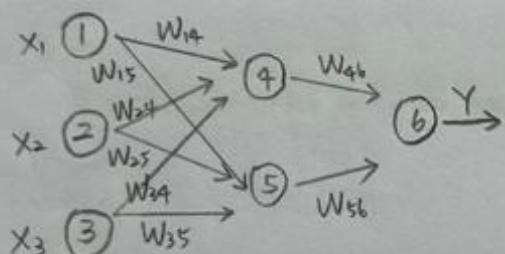
$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Recall} + \text{Precision}}$$

R →	↓ P	
	P	N
	P	TP
	N	FP
		TN

5.



$$(x_1, x_2, x_3, Y) = (1, 0, 1, 1)$$

w ₁₄	w ₁₅	w ₂₄	w ₂₅
0.2	-0.3	0.4	0.1

w ₃₄	w ₃₅	w ₄₆	w ₅₆
-1	0.2	-0.3	-0.2

神經元 4

神經元 5

$$x \quad 1 \quad 0 \quad 1 \quad \theta_4$$

$$x \quad 1 \quad 0 \quad 1 \quad \theta_5$$

$$\theta_4 \quad \theta_5 \quad \theta_6$$

$$w \quad 0.2 \quad 0.4 \quad -1$$

$$w \quad -0.3 \quad 0.1 \quad 0.2$$

$$-0.4 \quad 0.2 \quad 0.1$$

$$x \cdot w \quad 0.2 \quad 0 \quad -1 \quad -0.4$$

$$x \cdot w \quad -0.3 \quad 0 \quad 0.2 \quad 0.2$$

$$\text{SUM} \quad -1.2$$

$$\text{SUM} \quad 0.1$$

$$\log(x) = \frac{e^{-1.2}}{1+e^{-1.2}}$$

$$\log(x) = \frac{e^{0.1}}{1+e^{0.1}}$$

$$\log(x) = \frac{e(x)}{1+e(x)}$$

11/13 第十週

●課堂重點

A. 期中考成績確認

B. 科技論文的解說

1 緒論(研究動機 研究目的 why? 刺激你做的研究 ex 肺癌)

文獻探討

2-1 造成的原因

2-2 相關資料 相關因素

2-3 跟前面資料做比對(跟我們做的研究方法比較 介紹)

(文獻的指出 大量蒐集已有研究的報告(前五年的文獻) 相關報告 從報告找基本資料來了解
ex 工作地點 環境 相關資料 相關因素)

3-1 研究方法：詳細介紹所使用的方法與架構，例如什麼是 DNN、CNN

3-2 實驗

4-1 資料的來源跟欄位的說明

4-2 證明我的方法比較好(假設使用 cnn) 需跟之前做比較或延伸基礎優於其他的方法(cnn 和 dnn 或其他比較)

5 結論(因研究的限制 未來的研究 自我期許)，這篇論文用來解決什麼問題，提出未來研究將補足點(目前尚未完成處)

6 參考文獻(APA 文獻的引用)

C. 解說期末專題報告事項

報告內容中需有的資料

1 緒論(研究動機、目的、預期成果、資料集說明)

2 資料前處理(程式碼、實驗結果)

3 決策樹(程式碼、實驗結果、參數比較、管理意涵)

4 隨機森林(RF)(程式碼、實驗結果、參數比較、管理意涵)

5 深度神經網絡(DNN)(程式碼、實驗結果、參數比較、管理意涵)

6 支援向量機(SVM)(程式碼、實驗結果、參數比較、管理意涵)

7 結論(實驗結果比較)

11/20 第十一週

●Python 演練(RNN)

1. 台積電股票預測_1

```
# -*- coding: utf-8 -*-
"""

Created on Mon Nov 20 10:12:07 2023

@author: Nasser
"""

from math import sqrt
from numpy import concatenate
from matplotlib import pyplot
from pandas import read_csv, DataFrame, concat
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from keras.models import Sequential
from keras.layers import Dense, SimpleRNN
from sklearn.metrics import mean_squared_error, mean_absolute_error

# Convert series to supervised learning
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]
    df = DataFrame(data)
    cols, names = list(), list()

    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
        names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]

    for i in range(0, n_out):
        cols.append(df.shift(-i))
        if i == 0:
            names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
        else:
            names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]

    agg = concat(cols, axis=1)
    agg.columns = names
```

```

if dropnan:
    agg.dropna(inplace=True)

return agg

# Load dataset
filename = 'C:/Users/Nasser/Desktop/人工智慧筆記/1120/StockTSMC.csv'
filenameOut = 'C:/Users/Nasser/Desktop/人工智慧筆記/1120/StockTSMC_pred.csv'
filenameModel = 'C:/Users/Nasser/Desktop/人工智慧筆記/1120/StockTSMC_model.h5'

dataset = read_csv(filename, header=0, index_col=0)
values = dataset.values

# Integer encode direction
encoder = LabelEncoder()
values[:, 4] = encoder.fit_transform(values[:, 4])

# Ensure all data is float
values = values.astype('float32')

# Normalize features
scaler = MinMaxScaler(feature_range=(0, 1))
scaled = scaler.fit_transform(values)

# Specify the number of lag hours
n_hours = 5
n_features = 5

# Frame as supervised learning
reframed = series_to_supervised(scaled, n_hours, 1)
print(reframed.shape)

# Split into train and test sets
values = reframed.values
n_train_hours = 200 * 5
train = values[:n_train_hours, :]
test = values[n_train_hours:, :]

# Split into input and outputs
n_obs = n_hours * n_features

```

```

train_X, train_y = train[:, :n_obs], train[:, -n_features]
test_X, test_y = test[:, :n_obs], test[:, -n_features]

# Reshape input to be 3D [samples, timesteps, features]
train_X = train_X.reshape((train_X.shape[0], n_hours, n_features))
test_X = test_X.reshape((test_X.shape[0], n_hours, n_features))
print(train_X.shape,len(train_X),train_y.shape)

# Design network
model = Sequential()
model.add(SimpleRNN(50, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')

# Fit network
history = model.fit(train_X, train_y, epochs=5, batch_size=72, validation_data=(test_X, test_y))

# Plot history
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()

# Make a prediction
yhat = model.predict(test_X)

# Invert scaling for forecast
test_X = test_X.reshape((test_X.shape[0], n_hours * n_features))
inv_yhat = concatenate((yhat, test_X[:, -(n_features-1):]), axis=1)
inv_yhat = scaler.inverse_transform(inv_yhat)
inv_yhat = inv_yhat[:, 0]

# Invert scaling for actual
test_y = test_y.reshape((len(test_y), 1))
inv_y = concatenate((test_y, test_X[:, -(n_features-1):]), axis=1)
inv_y = scaler.inverse_transform(inv_y)
inv_y = inv_y[:, 0]

# Calculate RMSE and MAE
rmse = sqrt(mean_squared_error(inv_y, inv_yhat))

```

```

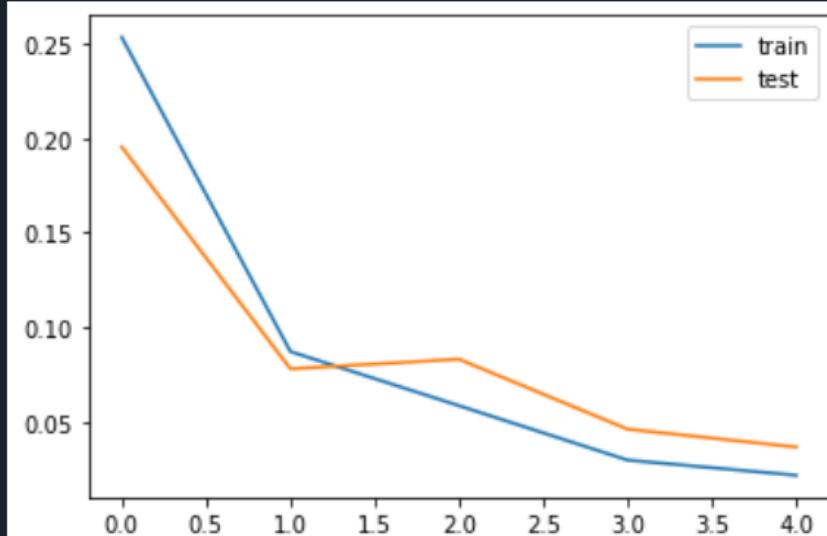
mae = mean_absolute_error(inv_y, inv_yhat)
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)

# Save the model
filenameModel = 'C:/Users/Nasser/Desktop/人工智慧筆記/1120/StockTSMC_model.h5'
model.save(filenameModel)
print("Saved model:", filenameModel)

# Combine the predicted results with the real data and save to a CSV file
filenameOut = 'C:/Users/Nasser/Desktop/人工智慧筆記/1120/StockTSMC_pred.csv'
df_PredResult = DataFrame({"Real": inv_y, "Pred": inv_yhat})
print(df_PredResult.head())
df_PredResult.to_csv(filenameOut)

```

結果：



```

Saved model: C:/Users/Nasser/Desktop/人工智慧筆記/
1120/StockTSMC_model.h5
      Real          Pred
0  566.0  576.627136
1  572.0  576.267883
2  585.0  575.857056
3  594.0  575.815735
4  599.0  600.993774

```

2. 台積電股票_2

```
# -*- coding: utf-8 -*-
"""

Created on Mon Nov 20 10:12:07 2023

@author: Nasser
"""

import pandas as pd
import numpy as np
from math import sqrt
from numpy import concatenate
from matplotlib import pyplot
from pandas import read_csv, DataFrame, concat
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from plotly.graph_objs import Scatter
#from keras.layers import Dense, SimpleRNN
from keras.layers import Dense, SimpleRNN
from sklearn.metrics import mean_squared_error, mean_absolute_error

import matplotlib.pyplot as plt

# Convert series to supervised learning
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]
    df = DataFrame(data)
    cols, names = list(), list()

    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
        names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]

    for i in range(0, n_out):
        cols.append(df.shift(-i))
        if i == 0:
            names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
        else:
            names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
```

```

agg = concat(cols, axis=1)
agg.columns = names

if dropnan:
    agg.dropna(inplace=True)

return agg

# Load dataset
filename = 'C:/Users/Nasser/Desktop/人工智慧筆記/1120/StockTSMC.csv'
filenameOut = 'C:/Users/Nasser/Desktop/人工智慧筆記/1120/StockTSMC_pred.csv'
filenameModel = 'C:/Users/Nasser/Desktop/人工智慧筆記/1120/StockTSMC_model.h5'

dataset = read_csv(filename, header=0, index_col=0)
values = dataset.values

# Integer encode direction
encoder = LabelEncoder()
values[:, 4] = encoder.fit_transform(values[:, 4])

# Ensure all data is float
values = values.astype('float32')

# Normalize features
scaler = MinMaxScaler(feature_range=(0, 1))
scaled = scaler.fit_transform(values)

# Specify the number of lag hours
n_hours = 5
n_features = 5

# Frame as supervised learning
reframed = series_to_supervised(scaled, n_hours, 1)
print(reframed.shape)

# Split into train and test sets
values = reframed.values
n_train_hours = 200 * 5
train = values[:n_train_hours, :]
test = values[n_train_hours:, :]

```

```

# Split into input and outputs
n_obs = n_hours * n_features
train_X, train_y = train[:, :n_obs], train[:, -n_features]
test_X, test_y = test[:, :n_obs], test[:, -n_features]

# Reshape input to be 3D [samples, timesteps, features]
train_X = train_X.reshape((train_X.shape[0], n_hours, n_features))
test_X = test_X.reshape((test_X.shape[0], n_hours, n_features))
print(train_X.shape,len(train_X),train_y.shape)

# Design network
model = Sequential()
model.add(SimpleRNN(50, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')

# Fit network
history = model.fit(train_X, train_y, epochs=5, batch_size=72, validation_data=(test_X, test_y))

# Plot history
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()

# Make a prediction
yhat = model.predict(test_X)

# Invert scaling for forecast
test_X = test_X.reshape((test_X.shape[0], n_hours * n_features))
inv_yhat = concatenate((yhat, test_X[:, -(n_features-1):]), axis=1)
inv_yhat = scaler.inverse_transform(inv_yhat)
inv_yhat = inv_yhat[:, 0]

# Invert scaling for actual
test_y = test_y.reshape((len(test_y), 1))
inv_y = concatenate((test_y, test_X[:, -(n_features-1):]), axis=1)
inv_y = scaler.inverse_transform(inv_y)

```

```

inv_y = inv_y[:, 0]

# Calculate RMSE and MAE
rmse = sqrt(mean_squared_error(inv_y, inv_yhat))
mae = mean_absolute_error(inv_y, inv_yhat)
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)

# Save the model
filenameModel = 'C:/Users/Nasser/Desktop/人工智慧筆記/1120/StockTSMC_model.h5'
model.save(filenameModel)
print("Saved model:", filenameModel)

# Combine the predicted results with the real data and save to a CSV file
filenameOut = 'C:/Users/Nasser/Desktop/人工智慧筆記/1120/StockTSMC_pred.csv'
df_PredResult = DataFrame({"Real": inv_y, "Pred": inv_yhat})
print(df_PredResult.head())
df_PredResult.to_csv(filenameOut)

predict_y=inv_yhat
test_y=inv_y

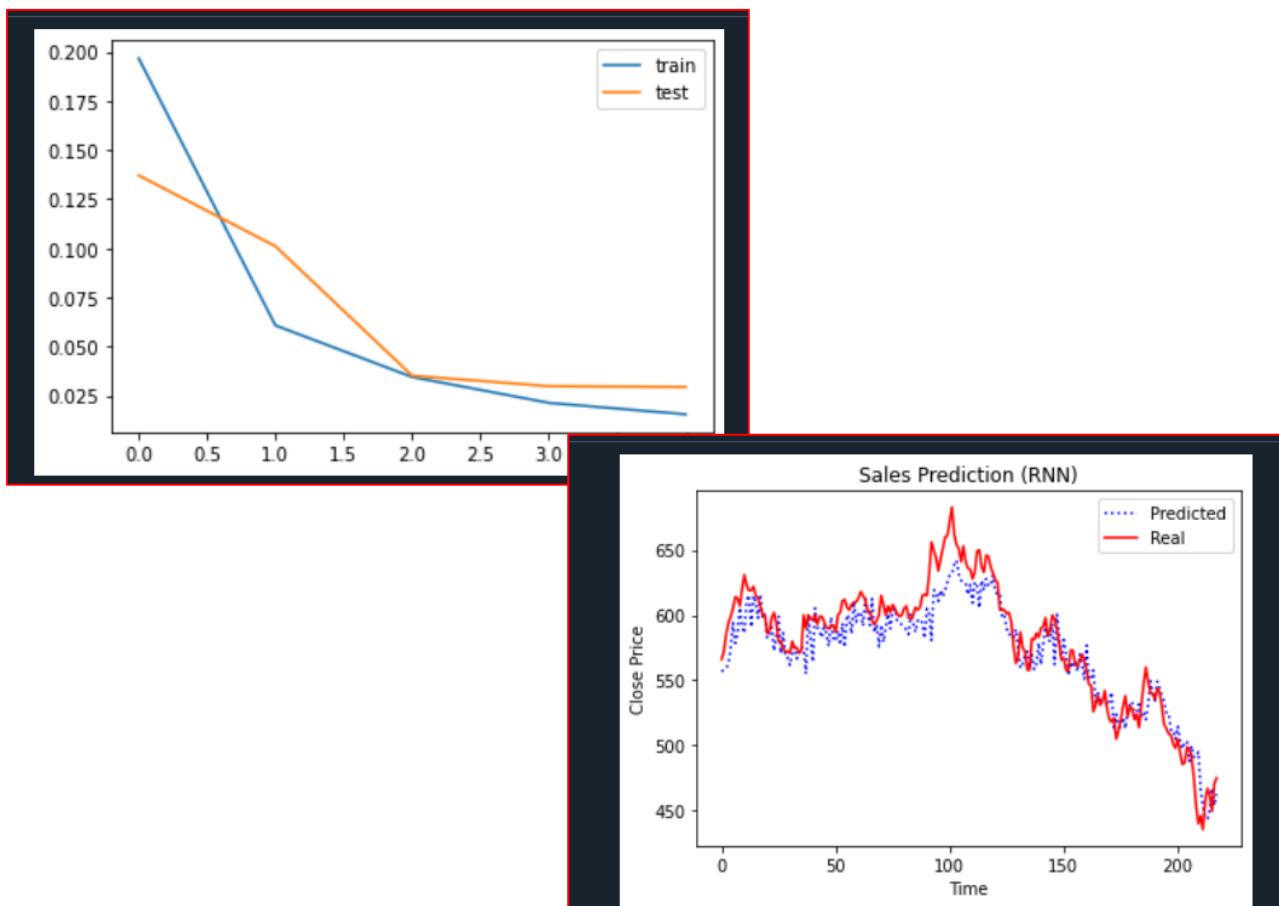
#畫圖 股價趨勢圖
plt.plot(predict_y, 'b:', label='Predicted') # 預測
plt.plot(test_y, 'r-', label='Real') # 實際
plt.legend(['Predicted', 'Real'])
plt.xlabel("Time")
plt.ylabel("Close Price")
plt.title("Sales Prediction (RNN)")
plt.show()

#建立 DataFrame, 加入 predict_y、test_y, 準備以 plotly 繪圖
df2 = pd.DataFrame({"predict": list(predict_y), "real": list(test_y)})

#轉換為 numpy 陣列, 並轉為 float
df2["predict"] = np.array(df2["predict"]).astype(float)
df2["real"] = np.array(df2["real"]).astype(float)

```

結果：



●課堂重點

1. RNN

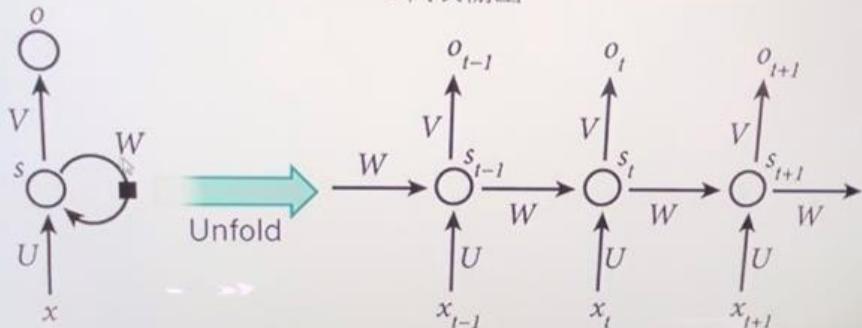
遞歸神經網路(RNN) (Recurrent Neural Network)

- RNN
 - 傳統的神經網絡常常假設輸入(輸出)是獨立於彼此的，這對於某些應用來說是不可行的
 - Recurrent就是循環往復的意思，網絡對於序列數據的每個元素進行同樣的操作，網絡的輸出取決於之前的計算
 - 應用(序列數據)
 - 文本，是字母和詞彙的序列
 - 語音，是音節的序列
 - 視頻，是圖像的序列；氣象觀測數據，股票交易數據

2. RNN 的典型結構

RNN的典型結構

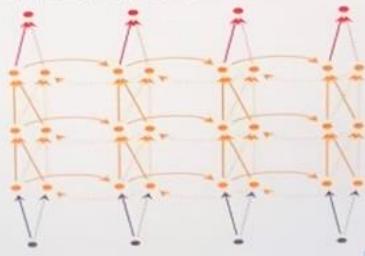
x指的是某一時刻網絡的輸入
s是隱藏狀態，代表著網絡的「記憶」
o代表輸出



3. RNN 雙向

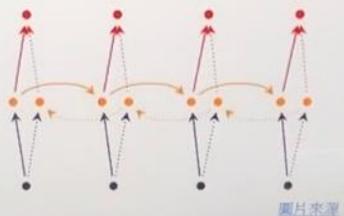
RNN (雙向深度RNN)

- 雙向RNN類似，只是每個step需要訓練多層的網絡，這使得模型更為強大(也需要更多的數據來訓練)



RNN (雙向RNN)

- 主要輸入不僅與過去的輸入，並且與將來的輸入有關的理念
- 需要預測一個句子中間缺失的詞語
 - 雙向RNN的結構很簡單，只是把兩個RNN在頂部結合，其輸出取決於了兩個RNN的隱藏狀態



4. RNN 優缺點

RNN 優缺點

• 優點

- 時間遞迴神經網路可以描述動態時間行為，因為和前饋神經網路接受較特定結構的輸入不同，
- RNN將狀態在自身網路中迴圈傳遞，因此可以接受更廣泛的時間序列結構輸入

• 缺點

- 簡單遞迴神經網路無法處理隨著遞迴，梯度爆炸或者梯度消失的問題，並且**難以捕捉長期時間關聯**
- 有效的處理方法是忘掉錯誤的資訊，記住正確的資訊
 - LSTM能夠比較好的解決這個問題

5. 課堂製作期末專題報告

※小考

1. RNN 有什麼優缺點？

優點：

1. RNN 很適合處理序列數據，因為考慮了先前的資訊。

2. 可以和 CNN 一起使用得到更好的效果。

缺點：

1. 由於訓練過程中 W , U 和 b 是同一組參數，容易出現梯度爆炸或梯度消失的情況。

2. RNN 相較於其他 CNN 和全連接需要更多的顯存空間，更難訓練。

3. 如果使用 tanh、relu 作為啟動函數，沒辦法處理太長的數列。

11/27 第十二週

●課堂重點

1. 資料集原始資料(row data)格式可能與程式碼所需不相同→資料前處理

(1)行列互換

(2)資料欄位處理(調整欄位裡的資料)

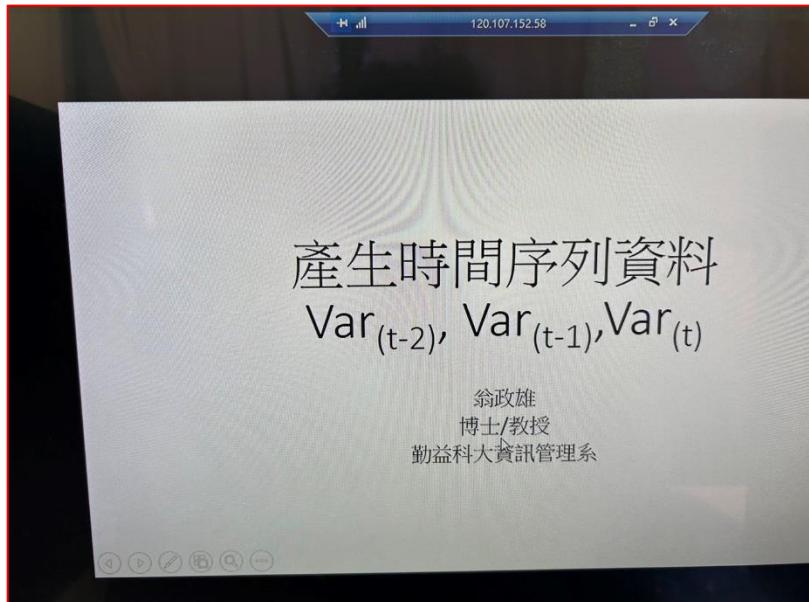
(3)處理空值、遺失值

(4)若是做 RNN、LSTM，須轉為時間序列資料

(EX. 用前 3 小時預測第 4 小時的結果，必須 4 小時的資料，合併成 1 列(row))

2. 產生時間序列資料講解

a.



b.

Data shift

($i=2, 1$)

```
n_in=2 # t-2,(n_in=2)
for i in range(n_in, 0, -1):
    cols.append(df.shift(i))
names += [(f"var%d(t-%d)" % (j+1, i)) for j in range(n_vars)]
print(cols)
agg = concat(cols, axis=1)#欄合併
agg.columns = names
```

A	B	C	D
2000/1/1 00:00	1	10	11
2000/1/1 12:00	2	20	22
2000/1/2 00:00	3	30	33
2000/1/2 12:00	4	40	44
2000/1/3 00:00	5	50	55



	var1(t-2)	var2(t-2)	var3(t-2)	var4(t-2)	var1(t-1)	var2(t-1)	var3(t-1)	var4(t-1)
2000/1/1 00:00					1	10	11	12
2000/1/1 12:00					2	20	22	24
2000/1/2 00:00	1	10	11	12	2	20	22	24
2000/1/2 12:00	2	20	22	24	3	30	33	35
2000/1/3 00:00	3	30	33	35	4	40	44	36

c.

Raw data

Spyder (Python 3.6)

```
#0.1: DataFrame.shift(periods=1, freq=None, axis=0)
# Importing pandas as pd
import pandas as pd
# Using pandas' built-in concat function
from pandas import concat
# Creating raw index values for our data frame
# We have taken time frequency to be of 12 hours interval
# We are generating five index value using "period = "S" parameter
ind = pd.date_range('01 / 01 / 2000', periods = 5, freq = '12H')
# Creating a dataframe with 4 columns
# Using pandas' built-in DataFrame function
df = pd.DataFrame({'A':[1, 2, 3, 4, 5],
                    'B':[10, 20, 30, 40, 50],
                    'C':[11, 22, 33, 44, 55],
                    'D':[12, 24, 35, 36, 21]},
                    index = ind)

# Print the dataframe
print(df)
print(df.to_csv('UnstackedPollutionLSTMData.csv'))
df.to_csv('filenameOut')

# shift index axis by two periods in positive direction
# axis = 0 is set by default
# df.shift(2).shift(2, axis = 0)
```

Usage

Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help will also appear automatically after writing a left-clicked note to an object. You can activate this behavior in Preferences > Help.

New to Spyder? Read our tutorial

Variable explorer Help Run Cell

2000-01-01 00:00:00 1.0 10.0 11.0 ... 20.0
2000-01-01 12:00:00 2.0 20.0 22.0 ... 24.0
2000-01-02 00:00:00 3.0 30.0 33.0 51.0
2000-01-02 12:00:00 3.0 30.0 33.0 51.0
2000-01-03 00:00:00 4.0 40.0 44.0 36.0

Data (0 rows)
var1(t) var2(t) var3(t) var4(t) ... var2(t-2) var2(t-1) var1(t-2) var1(t-1) var2(t-2) var2(t-1) ... var1(t-2) var1(t-1)
2000-01-01 00:00:00 1.0 10.0 11.0 ... 20.0
3 10 11 20 22 24 1 10 11 12 2 20 22 24
2000-01-02 12:00:00 2.0 20.0 22.0 ... 24.0
4 30 33 40 44 36 2 30 33 35 3 40 44 36
2000-01-03 00:00:00 3.0 30.0 33.0 ... 50.0
5 40 44 36 35 2 3 40 44 36 4 50 55 21

[5 rows x 12 columns]

In [4]:

ipython3.6

d.

Data sequence

$\text{Var}_{(t-2)}, \text{Var}_{(t-1)}, \text{Var}_{(t)}$

	A	B	C	D
2000/1/1 00:00	1	10	11	12
2000/1/1 12:00	2	20	22	24
2000/1/2 00:00	3	30	33	51
2000/1/2 12:00	4	40	44	36
2000/1/3 00:00	5	50	55	2

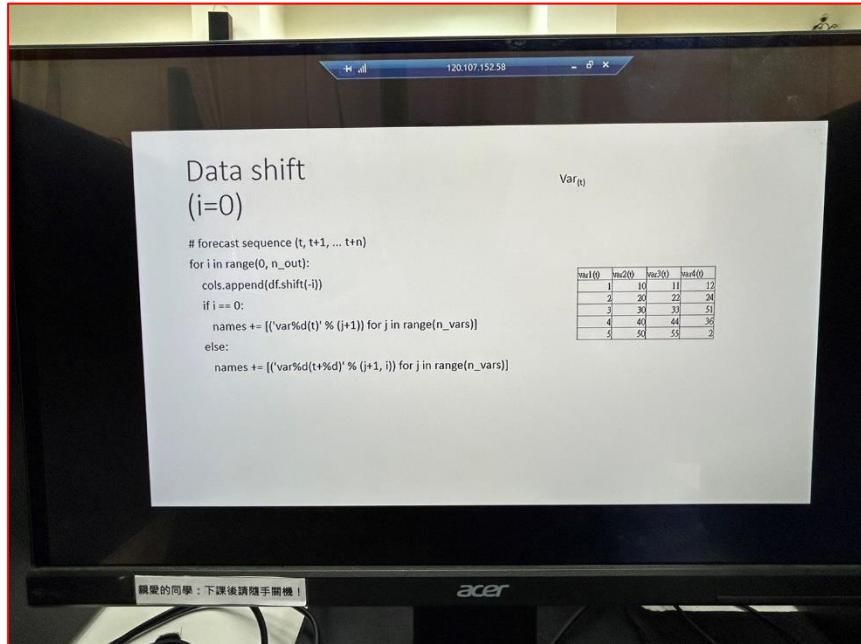
The screenshot shows an Excel spreadsheet titled "ABCD - Excel". The data is organized into four columns: A, B, C, and D. The first column contains dates and times. The second column contains the value 1. The third column contains the value 10. The fourth column contains the value 11. This pattern repeats for each row, corresponding to the data sequence provided above. The Excel interface includes a ribbon menu at the top and various toolbars and status bars.

e.

Data sequence
with NA

	A	B	C	D	E	F	G	H	I	J	K	L	M	
1		var1(t-2)	var2(t-2)	var3(t-2)	var4(t-2)	var1(t-1)	var2(t-1)	var3(t-1)	var4(t-1)	var1(t)	var2(t)	var3(t)	var4(t)	
2	2000/1/1 00:00									1	10	11	12	
3	2000/1/1 12:00						1	10	11	12	2	20	22	24
4	2000/1/2 00:00	1	10	11	12	2	20	22	24	3	30	33	51	
5	2000/1/2 12:00	2	20	22	24	3	30	33	51	4	40	44	36	
6	2000/1/3 00:00	3	30	33	51	4	40	44	36	5	50	55	2	

f.



●Python 演練

1. 演練一

```
# -*- coding: utf-8 -*-
"""

Created on Mon Nov 27 10:06:48 2023

@author: Nasser
"""

# 用法: DataFrame.shift(periods=1, freq=None, axis=0)

# importing pandas as pd
import pandas as pd
from pandas import concat
from pandas import DataFrame

# Creating row index values for our data frame
# We have taken a time frequency of 12 hours interval
# We are generating five index values using the "periods = 5" parameter

ind = pd.date_range('01 / 01 / 2000', periods=5, freq='12H')

# Creating a dataframe with 4 columns
# using "ind" as the index for our dataframe

df = pd.DataFrame({'A': [1, 2, 3, 4, 5],
                   'B': [10, 20, 30, 40, 50],
                   'C': [11, 22, 33, 44, 55],
                   'D': [12, 24, 51, 36, 2]}, index=ind)

df=DataFrame({'D':df['D'], 'C':df['C'], 'B':df['B'], 'A':df['A']})

# Print the dataframe

print('原始 df')
print(df)
filenameOut = 'C:/Users/Nasser/Desktop/人工智慧筆記/ABCD.csv'
df.to_csv(filenameOut)
```

```

# shift index axis by two periods in the positive direction
# axis = 0 is set by default
# df_shift2 = df.shift(2, axis=0)
# print('df.shift(2, axis = 0)')
# print(df_shift2)

n_in = 2 # t-2, (n_in=2)
n_vars = 4 # features
n_out = 1 # t (n_out)
cols, names = list(), list()
# input sequence (t-n, ... t-1)
print('Data shift')

for i in range(n_in, 0, -1):
    cols.append(df.shift(i))
    names += [('var%d(t-%d)' % (j + 1, i)) for j in range(n_vars)]
print(cols)

# forecast sequence (t, t+1, ... t+n)
for i in range(0, n_out):
    cols.append(df.shift(-i))
    if i == 0:
        names += [('var%d(t)' % (j + 1)) for j in range(n_vars)]
    else:
        names += [('var%d(t+%d)' % (j + 1, i)) for j in range(n_vars)]

agg = concat(cols, axis=1) # 欄合併
# agg = DataFrame(cols)
agg.columns = names

# print(agg)
filenameOut = 'C:/Users/Nasser/Desktop/人工智慧筆記/ABCD.csv'
agg.to_csv(filenameOut)

# drop rows with NaN values
agg.dropna(inplace=True)
filenameOut = 'C:/Users/Nasser/Desktop/人工智慧筆記/ABCDdropna.csv'
agg.to_csv(filenameOut)

```

結果：

原始df

		D	C	B	A
2000-01-01	00:00:00	12	11	10	1
2000-01-01	12:00:00	24	22	20	2
2000-01-02	00:00:00	51	33	30	3
2000-01-02	12:00:00	36	44	40	4
2000-01-03	00:00:00	2	55	50	5

Data shift

		D	C	B	A
2000-01-01	00:00:00	NaN	NaN	NaN	NaN
2000-01-01	12:00:00	NaN	NaN	NaN	NaN
2000-01-02	00:00:00	12.0	11.0	10.0	1.0
2000-01-02	12:00:00	24.0	22.0	20.0	2.0
2000-01-03	00:00:00	51.0	33.0	30.0	3.0,
2000-01-01	00:00:00	NaN	NaN	NaN	NaN
2000-01-01	12:00:00	12.0	11.0	10.0	1.0
2000-01-02	00:00:00	24.0	22.0	20.0	2.0
2000-01-03	00:00:00	51.0	33.0	30.0	3.0,

2. 演練二

```
# -*- coding: utf-8 -*-
"""

Created on Mon Nov 27 09:23:48 2023

@author: Nasser
"""

#用法: DataFrame.shift (periods=1, freq=None, axisse)
# importing pandas as pd
import pandas as pd
from pandas import concat
from pandas import DataFrame

ind = pd.date_range('01 / 01 / 2000', periods = 5, freq = '12H')
# Creating a dataframe with 4 columns
# using "ind" as the index for our dataframe
df = pd.DataFrame ({"A": [1, 2, 3, 4, 5],
                     "B": [10, 20, 30, 40, 50],
                     "C": [11, 22, 33, 44, 55],
                     "D": [12, 24, 51, 36, 2]},
                     index = ind)

# Print the dataframe
print('原始 df')
print(df)
filenameout='C:/Users/Nasser/Desktop/人工智慧筆記/ABCD.csv'
```

```

df.to_csv(filenameout)

n_in=2      # t-2, (n_in=2)
n_vars=4    #features
n_out=1
cols, names = list(), list()

print ('Data shift')
for i in range(n_in, 0, -1):
    cols.append(df.shift (i))
    names += [('var%d (t-%d)' % (j+1,1)) for j in range (n_vars)]
print(cols)

for i in range(0,n_out):
    cols.append(df.shift (-i))
    if i == 0:
        names += [('var%d (t)' % (j+1)) for j in range (n_vars)]
    else:
        names += [('var%d (t+%d)' % (j+1,i)) for j in range (n_vars)]

agg = concat(cols,axis=1)
agg.columns = names

filenameOut= 'C:/Users/Nasser/Desktop/人工智慧筆記/ABCD.csv'
agg.to_csv(filenameOut)

agg.dropna(inplace=True)
filenameOut='C:/Users/Nasser/Desktop/人工智慧筆記/ABCDdropna.csv'

```

結果：

原始df				
		A	B	C
2000-01-01	00:00:00	1	10	11
2000-01-01	12:00:00	2	20	22
2000-01-02	00:00:00	3	30	33
2000-01-02	12:00:00	4	40	44
2000-01-03	00:00:00	5	50	55
				2

Data shift				
		A	B	C
2000-01-01	00:00:00	NaN	NaN	NaN
2000-01-01	12:00:00	NaN	NaN	NaN
2000-01-02	00:00:00	1.0	10.0	11.0
2000-01-02	12:00:00	2.0	20.0	22.0
2000-01-03	00:00:00	3.0	30.0	33.0
2000-01-01	00:00:00	NaN	NaN	NaN
2000-01-01	12:00:00	1.0	10.0	11.0
2000-01-02	00:00:00	2.0	20.0	22.0
2000-01-02	12:00:00	3.0	30.0	33.0

3. 演練三

```
# -*- coding: utf-8 -*-
#####
Created on Mon Nov 27 10:32:25 2023

@author: Nasser
#####

# -*- coding: utf-8 -*-
#####
Created on Mon Nov 27 09:23:48 2023

@author: Nasser
#####

#用法: DataFrame.shift (periods=1, freq=None, axisse)
# importing pandas as pd
import pandas as pd
from pandas import concat
from pandas import DataFrame

ind = pd.date_range('01 / 01 / 2000', periods = 5, freq = '12H')
# Creating a dataframe with 4 columns
# using "ind" as the index for our dataframe
df = pd.DataFrame ({"A": [1, 2, 3, 4, 5],
                     "B": [10, 20, 30, 40, 50],
                     "C": [11, 22, 33, 44, 55],
                     "D": [12, 24, 51, 36, 2]},
                     index = ind)

# Print the dataframe
print('原始 df')
print(df)
filenameout='C:/Users/Nasser/Desktop/人工智慧筆記/ABCD.csv'
df.to_csv(filenameout)

n_in=2      # t-2, (n_in=2)
n_vars=4   #features
n_out=1
cols, names = list(), list()
```

```

print ('Data shift')
for i in range(n_in, 0, -1):
    cols.append(df.shift (i))
    names += [('var%d (t-%d)' % (j+1,1)) for j in range (n_vars)]
print(cols)

for i in range(0,n_out):
    cols.append(df.shift (-i))
    if i == 0:
        names += [('var%d (t)' % (j+1)) for j in range (n_vars)]
    else:
        names += [('var%d (t+%d)' % (j+1,i)) for j in range (n_vars)]

agg = concat(cols, axis=1)
agg.columns = names

filenameOut= 'C:/Users/Nasser/Desktop/人工智慧筆記/ABCD.csv'
agg.to_csv(filenameOut)

agg.dropna(inplace=True)
filenameOut='C:/Users/Nasser/Desktop/人工智慧筆記/ABCDdropna.csv'
agg.to_csv(filenameOut)
print('Data 欄合併')
print(agg)

n_features = 4
aggNumpy=agg.to_numpy()
print(aggNumpy[:, -n_features]) #取 x
print(aggNumpy[:, -n_features]) #取 y

```

結果：

原始df

		A	B	C	D
2000-01-01	00:00:00	1	10	11	12
2000-01-01	12:00:00	2	20	22	24
2000-01-02	00:00:00	3	30	33	51
2000-01-02	12:00:00	4	40	44	36
2000-01-03	00:00:00	5	50	55	2

Data shift

		A	B	C	D
2000-01-01	00:00:00	NaN	NaN	NaN	NaN
2000-01-01	12:00:00	NaN	NaN	NaN	NaN
2000-01-02	00:00:00	1.0	10.0	11.0	12.0
2000-01-02	12:00:00	2.0	20.0	22.0	24.0
2000-01-03	00:00:00	3.0	30.0	33.0	51.0
2000-01-01	00:00:00	NaN	NaN	NaN	NaN
2000-01-01	12:00:00	1.0	10.0	11.0	12.0
2000-01-02	00:00:00	2.0	20.0	22.0	24.0
2000-01-02	12:00:00	3.0	30.0	33.0	51.0
2000-01-03	00:00:00	1.0	10.0	11.0	12.0

4. 講解以上程式的換行過程

換行前

	A	B	C	D	E	F	G
1							
2	1	2	3	4		1月1日	t-3
3	11	22	33	44		1月2日	t-2
4	111	222	333	444		1月3日	t-1
5	1111	2222	3333	4444		1月4日	t
6	11111	22222	33333	44444		1月5日	

過程

10	t-3	t-2	t-1	t
11				
12	1	2	3	4
13	11	22	33	44
14	111	222	333	444
15	1111	2222	3333	4444
16	11111	22222	33333	44444
17	11111	22222	33333	44444

換行後

10	1	2	3	4	11	22	33	44	111	222	333	444	1111	2222	3333	4444
11	11	22	33	44	111	222	333	444	1111	2222	3333	4444	11111	22222	33333	44444
12	111	222	333	444	1111	2222	3333	4444	11111	22222	33333	44444				
13	1111	2222	3333	4444	11111	22222	33333	44444								
14	11111	22222	33333	44444												

5. 台積電股票預測_講解

```
# Normalize features #資料轉換成 0/1 資料正規
scaler = MinMaxScaler(feature_range=(0, 1))
scaled = scaler.fit_transform(values)

# Specify the number of lag hours
n_t = 3
#用前 n_t(3) t-3 預測第 n_t(t)小時的結果，即第 n_t+1(4)小時的結果
#用(t-3),(t-2),(t-1)預測(t)的結果
n_features = 5 #特徵數

# Frame as supervised learning
reframed = series_to_supervised(scaled, n_t, 1)
print(reframed.shape)

# Split into train and test sets
values = reframed.values
n_train_t = 200 * 5 #取前 200 周每周 5 天的資料作為訓練資料，剩餘資料做驗證
train = values[:n_train_t, :]
test = values[n_train_t:, :]

# Split into input and outputs
n_obs = n_t * n_features #3*5=15
train_X, train_y = train[:, :n_obs], train[:, -n_features]
test_X, test_y = test[:, :n_obs], test[:, -n_features]

# Reshape input to be 3D [samples, timesteps, features]
train_X = train_X.reshape((train_X.shape[0], n_t, n_features))
#(200,15)->(200,3,5)
test_X = test_X.reshape((test_X.shape[0], n_t, n_features))
#把 一維資料 轉成 三維資料

print("train_X.shape, len(train_X), train_y.shape")
print(train_X.shape, len(train_X), train_y.shape)

# Design network
model = Sequential()
model.add(SimpleRNN(50, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
```

```

model.compile(loss='mae', optimizer='adam')

# Fit network
history = model.fit(train_X, train_y, epochs=5, batch_size=72, validation_data=(test_X, test_y))

# Plot history
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()

# Make a prediction
yhat = model.predict(test_X)

# Invert scaling for forecast (反正規化)
test_X = test_X.reshape((test_X.shape[0], n_t * n_features))
inv_yhat = concatenate((yhat, test_X[:, -(n_features-1):]), axis=1)
inv_yhat = scaler.inverse_transform(inv_yhat)
inv_yhat = inv_yhat[:, 0]

# Invert scaling for actual
test_y = test_y.reshape((len(test_y), 1))
inv_y = concatenate((test_y, test_X[:, -(n_features-1):]), axis=1)
inv_y = scaler.inverse_transform(inv_y)
inv_y = inv_y[:, 0]

# Calculate RMSE and MAE
rmse = sqrt(mean_squared_error(inv_y, inv_yhat))
mae = mean_absolute_error(inv_y, inv_yhat)
print('Test RMSE: %.3f' % rmse)
print('Test MAE: %.3f' % mae)

# Save the model
filenameModel = 'C:/Users/Nasser/Desktop/人工智慧筆記/1120/StockTSMC_model.h5'
model.save(filenameModel)
print("Saved model:", filenameModel)

```

輸出結果中有 RMSE、MAE，需要讓這兩個(誤差)值越小越好

※小考

今日無小考

12/4 第十三週

●汽車辨識 CNN

```
##Readme
#CNN 汽車辨識 Python designed by 翁政雄博士/教授
#PythonCode_CNNCar CNNCarGrey

import numpy as np
from keras.utils import np_utils
np.random.seed(10)
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense

#自訂函數
#大雄自訂函數,顯示預測結果
def show_labels_predictions(labels, predictions, start_id, num):
    for i in range(0, num):
        # 有 AI 預測結果資料,才在標題顯示預測結果
        if( len(predictions) > 0 ) :
            title = 'ai =' + str(predictions[i])
            #預測正確顯示(o),錯誤顯示(x)
            title += ': label=' + str(labels[i])
            title += (': 0' if predictions[i]==labels[i] else ' X')
            #輸出 1: ai = 1: label = 1:0
            print(i+1, ":", title)
        # 沒有 AI 預測結果資料,只在標題顯示真實數值
        else:
            title = 'label =' + str(labels[i])
            start_id+=1

#大雄自訂函數,計算 TPR,TNR...等指標值
def show_predictions (labels, predictions, start_id, num):
    TP=0
    TN=0
    FP=0
    FN=0
    for i in range(0, num):
        # 有 AI 預測結果資料,統計 TP, TN, FP,FN,僅 2 元分類有意義
```

```

if(len(predictions) > 0 ):
    if(predictions[i]==labels[i]):
        if (predictions[i]==1):
            TP=TP+1
        else:
            TN=TN+1
    else:
        if (predictions[i]==1):
            FP=FP+1
        else:
            FN=FN+1
    else:
        print("\n No Data")
        start_id+=1

print("\n 各種準確率指標")
print("TP=",TP)
print("FN=",FN)
print("TN=",TN)
print("FP=", FP)
print("TPR=",TP/(TP+FN))
print("TNR=",TN/(TN+FP))
print("Accuracy=", (TP+TN)/(TP+FN+TN+FP))
print("Precision=", TP/(TP+FP))

#顯示圖片、實際標籤、預測標籤、預測結果
def show_images_labels_predictions(images, labels, predictions, start_id,num=5):
    ax=plt
    plt.figure(figsize=(8,4))

    for i in range(0, num):
        # 有 AI 預測結果資料,才在標題顯示預測結果
        if(len(predictions) > 0 ) :
            title = 'label =' + str(labels[i])
            title += ', ai =' + str(predictions[i])
            #預測正確顯示(o),錯誤顯示(x)
            title += ('0' if predictions[i]==labels[i] else 'X')
            #輸出 1: ai = 1: label = 1 : 0
            print(i+1, ":" ,title)
            #顯示黑白圖片

```

```

        ax.imshow(images[i], cmap='binary')
        plt.show()

    # 沒有 AI 預測結果資料,只在標題顯示真實數值
    else:
        title = 'label=' + str(labels[i])
        start_id+=1

#####
#####自訂函數 結束

#讀取訓練資料
#讀取圖檔(jpg);圖檔前處理完成
#讀取圖檔標籤(label)
import os

import numpy as np
from PIL import Image

#設定檔案路徑(目錄)
filepath="C:/Users/Nasser/Desktop/人工智慧筆記/CNNCarData/TrainDataCar"

#計算圖檔數量
fn=0
for f in os.listdir(filepath):
    fn=fn+1
print("訓練資料檔案數:",fn)

#宣告圖片陣列空間
imageNum=fn
train_feature = np.zeros((imageNum, 28, 28), dtype=np.int)

#擷取檔案路徑(目錄)的檔案清單
for root, dirs, files in os.walk(filepath):
    # print(root) #當前目錄路徑
    # print(dirs) #當前路徑下所有子目錄
    # print(files) #當前路徑下所有非目錄子檔案
    files
#擷取訓練資料圖檔及標籤
i=0

```

```

Lablestr=""
for f in files:
    filename=filepath+'/'+f
    im = Image.open(filename)
#From image to array
    train_feature[i,:,:]= np.array(im)
#擷取標籤(label)
    label=os.path.splitext(f)[0][0]
    #print(i+1,'.',f,' ',os.path.splitext(f)[0],' ',label)

#紀錄 圖檔的標籤,存成字串
    if (Lablestr==''):
        Lablestr=str(label)
    else:
        Lablestr=Lablestr+', '+str(label)
    i=i+1
### 結束 for f in files:
#標籤字串轉陣列

Lablestr=Lablestr.split(',')
train_label = np.array(Lablestr, dtype = int)
#print(Lablestr)
print("訓練資料 Label: ",train_label)

#####讀測試資料#####
#讀取圖檔(jpg);圖檔前處理完成
#讀取圖檔標籤(label)

import os
import numpy as np
from PIL import Image

#設定檔案路徑(目錄)

filepath="C:/Users/Nasser/Desktop/人工智慧筆記/CNNCarData/TestDataCar"

#計算圖檔數量
fn=0
for f in os.listdir(filepath):
    fn=fn+1

```

```

print("讀測試資料檔案數:",fn)

#宣告圖片陣列空間
imageNum=fn
test_feature = np.zeros((imageNum, 28, 28), dtype=np.int)

#擷取檔案路徑(目錄)的檔案清單
for root, dirs, files in os.walk(filepath):
    # print(root) #當前目錄路徑
    # print(dirs) #當前路徑下所有子目錄
    # print(files) #當前路徑下所有非目錄子檔案
        files

#擷取測試資料圖檔及標籤
i=0
Lablestr=""
for f in files:
    filename=filepath+'/'+f
    im = Image.open(filename)
    #From image to array,
    test_feature[i,:,:]= np.array(im)
    #擷取標籤(label)
    label=os.path.splitext(f)[0][0]
    #print(i+1,'.',f,' ',os.path.splitext(f)[0],' ',label)

#紀錄 圖檔的標籤,存成字串
    if (Lablestr== ""):
        Lablestr=str(label)
    else:
        Lablestr=Lablestr+','+str(label)

    i=i+1
### 結束 for f in files:

#標籤字串轉陣列
Lablestr=Lablestr.split(',')
test_label = np.array(Lablestr, dtype = int)

#print(Lablestr)
print(test_label)

```

```

print("測試資料 Label: ", test_label)

#####
#####大雄修改 End

#將 Features 特徵值換為 N*28*28*1 的 4 維矩陣
train_feature_vector = train_feature.reshape(len(train_feature), \
                                               28,28,1).astype('float32')
test_feature_vector = test_feature.reshape(len(test_feature), \
                                            28,28,1).astype('float32')

#Features 特徵值標準化
train_feature_normalize = train_feature_vector/255
test_feature_normalize = test_feature_vector/255

#label 轉換為 One-Hot Encoding 編碼
train_label_onehot = np_utils.to_categorical(train_label)
test_label_onehot = np_utils.to_categorical(test_label)

#建立模型
model = Sequential()
#建立卷積層 1
model.add(Conv2D(filters=10, kernel_size=(3,3), padding='same', input_shape=(28,28,1),
activation= 'relu'))

#建立池化層 1
model.add(MaxPooling2D(pool_size=(2, 2))) #(10,14,14)

#建立卷積層 2
model.add(Conv2D(filters=20, kernel_size=(3,3), padding='same', activation= 'relu'))

#建立池化層 2
model.add(MaxPooling2D(pool_size=(2, 2))) #(20,7,7)

# Dropout 層防止過度擬合,斷開比例:0.2
model.add(Dropout(0.2))

#建立平坦層:20*7*7=980 個神經元
model.add(Flatten())

```

```
#建立隱藏層
model.add(Dense(units=256, activation='relu'))

#建立輸出層,units=3 只有 3 種輸出
model.add(Dense(units=3,activation='softmax'))

#定義訓練方式
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

#以(train_feature_normalize, train_label_onehot)資料訓練,
#訓練資料保留 20%作驗證,訓練 10 次、每批次讀取 200 筆資料,顯示簡易訓練過程

train_history=model.fit(x=train_feature_normalize, y=train_label_onehot, validation_split=0.2,
epochs=10, batch_size=30, verbose=0)

#評估準確率
scores = model.evaluate(test_feature_normalize, y=test_label_onehot)
print('\n 準確率=',scores[1])

#預測
#prediction=model.predict_classes(test_feature_normalize) #新版不支援
prediction=model.predict(test_feature_normalize)
prediction=np.argmax(prediction, axis=-1)#依據機率值高低,決定 label

#儲存模型
model.save("CNNScarGrey.h5")
print("儲存模型成功")

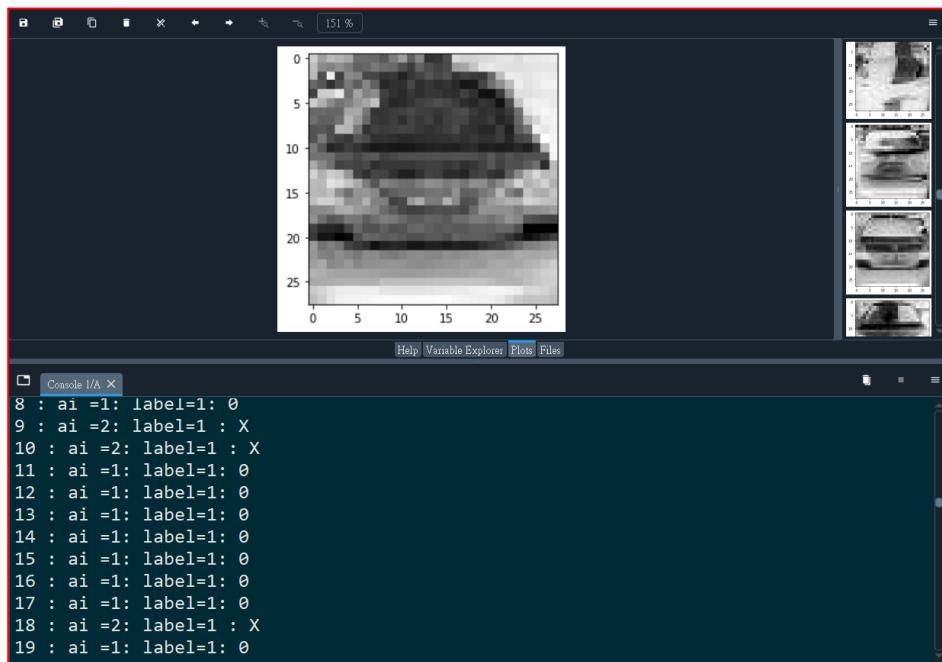
#####顯示結果

#顯示預測值、真實值
show_labels_predictions(test_label,prediction,0,imageNum)

#顯示 measures
show_predictions (test_label, prediction, 0, imageNum)

#顯示圖像、預測值、真實值
show_images_labels_predictions(test_feature, test_label, prediction, 0, imageNum)
```

結果：

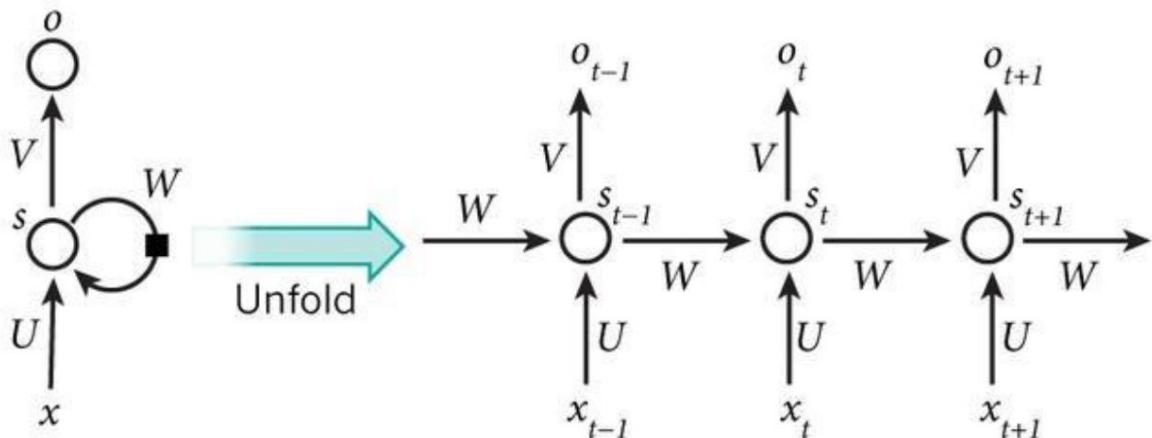


● 課堂講解

a. RNN 複習講解

RNN的典型結構

x 指的是某一時刻網絡的輸入
 s 是隱藏狀態，代表著網絡的「記憶」
 o 代表輸出

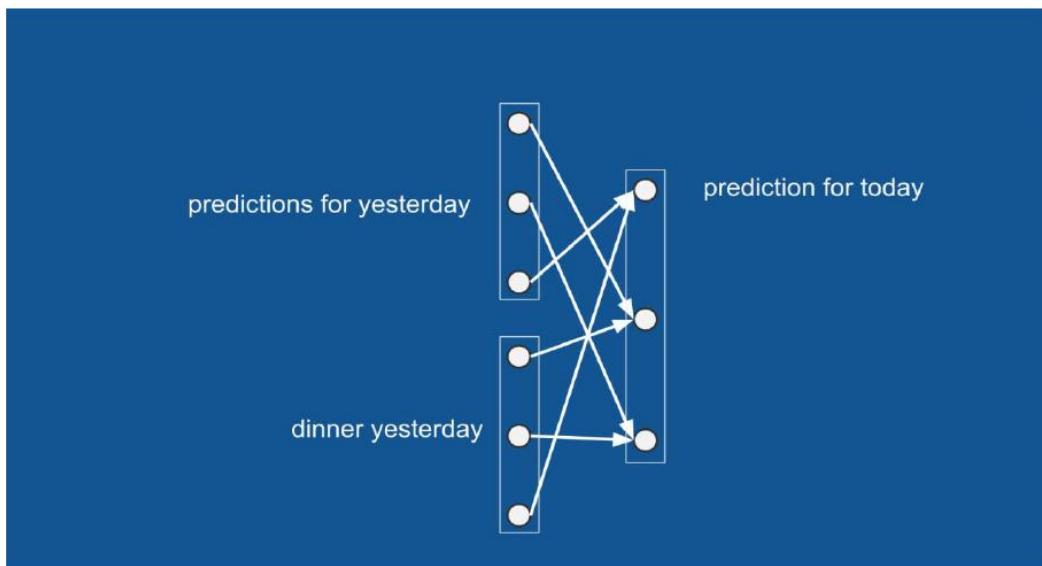


RNN 優缺點

- 優點

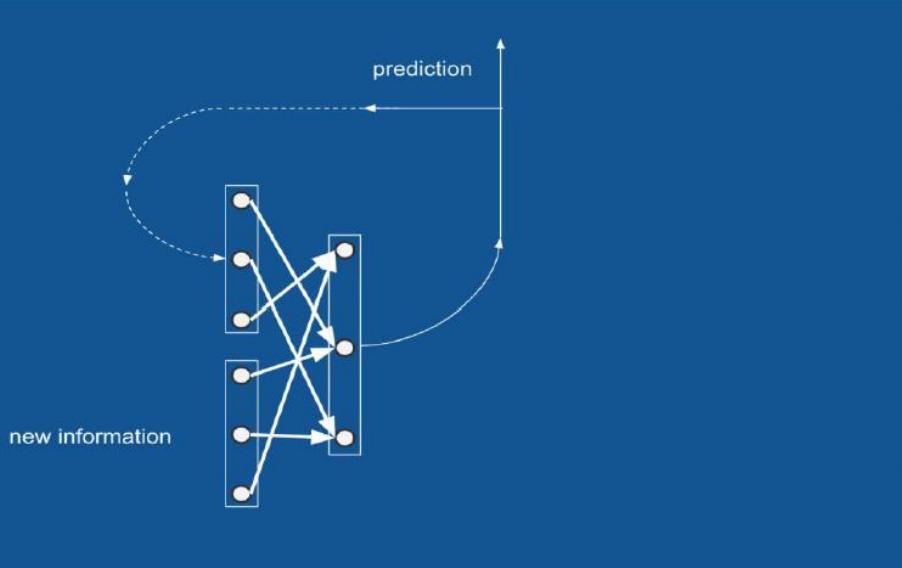
- 時間遞迴神經網路可以描述動態時間行為，因為和前饋神經網路接受較特定結構的輸入不同，
 - RNN將狀態在自身網路中迴圈傳遞，因此可以接受更廣泛的時間序列結構輸入

RNN (晚餐)預測(3組向量)



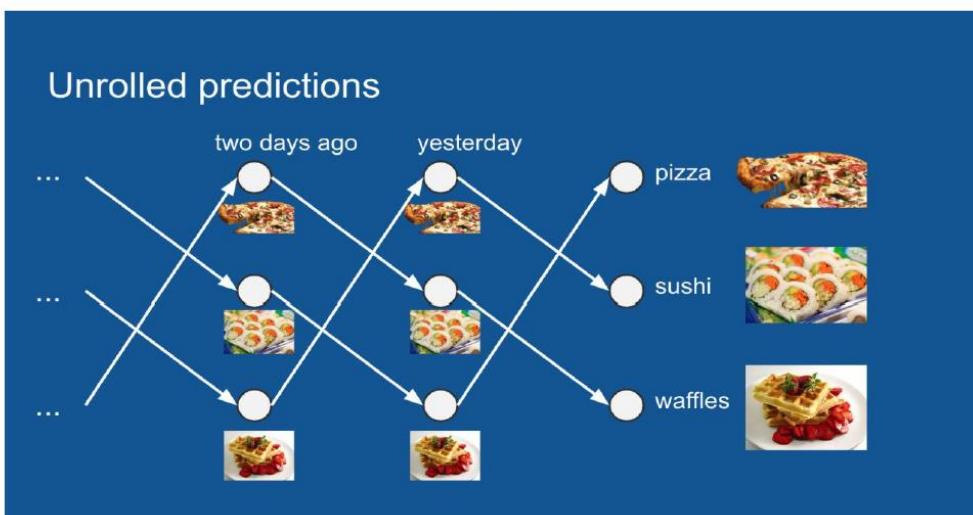
將昨天的預測、昨天的結果、以及今天的預測，用向量表示後的示意圖。[圖片來源](#)

RNN (晚餐)預測(保存 此次預測結果)



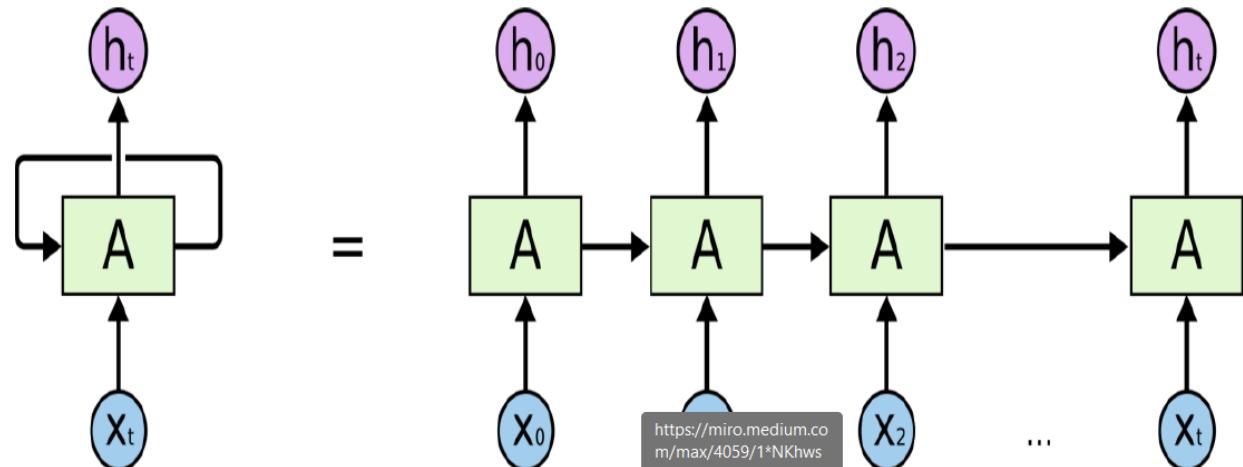
將今天預測的結果收回，變為明天的「昨日」預測。[圖片來源](#)

RNN (晚餐)



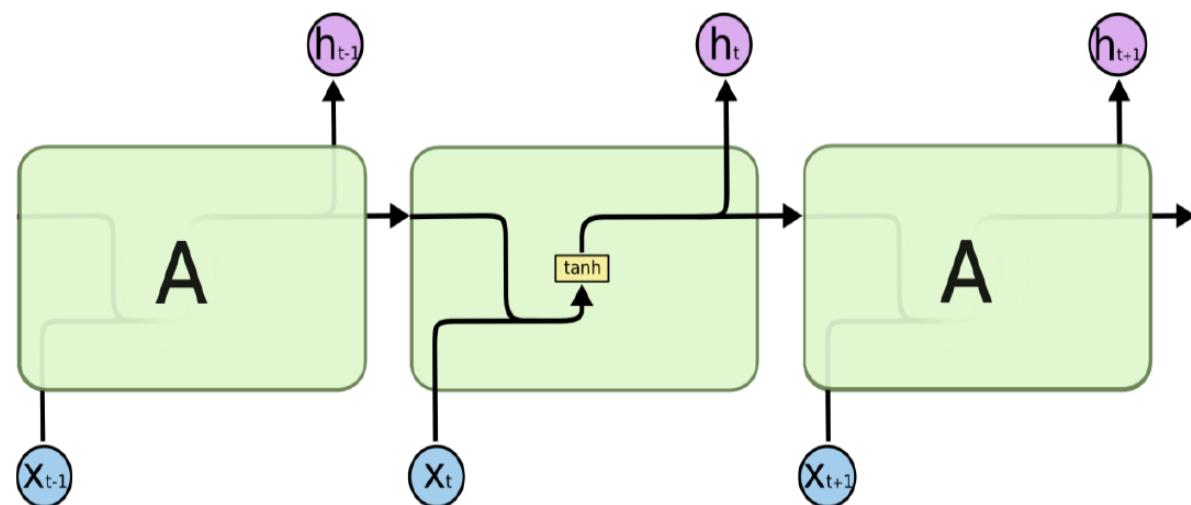
延伸過後的神經網路。[圖片來源](#)

RNN (模型)

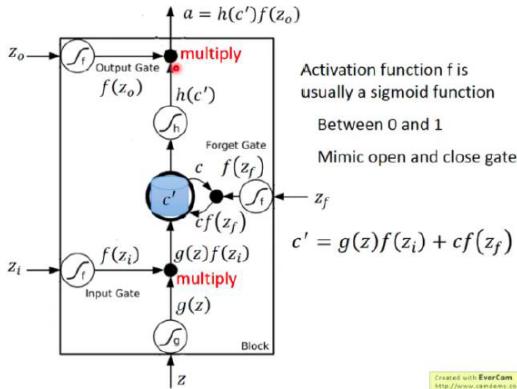


https://miro.medium.com/max/4059/1*NKhwsOYNUT5xU7Pyf6Znhg.png

RNN (模型)(雙曲函數 tanh)

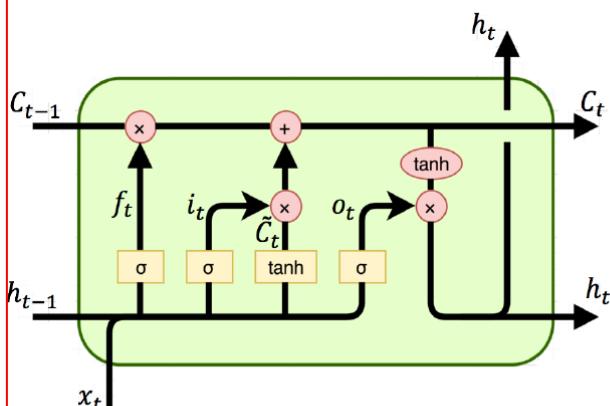


LSTM



- 資料input進一個LSTM cells，數學上可表示為 $g(z)$
- 第一個先遇到的就是input gate，input gate會使用 $f(z_i)$ (Activation function f)，來表示input gate開啟的機率
- 第二關會是Memory cell。首先，先紀錄當下input值加上前一次Memory cell裡的值並乘上forget gate的機率，看是否要遺忘前一次紀錄。
- 最後一關就是output gate，output gate會確認是否把值放出，也是以機率的方式來使用。

LSTM



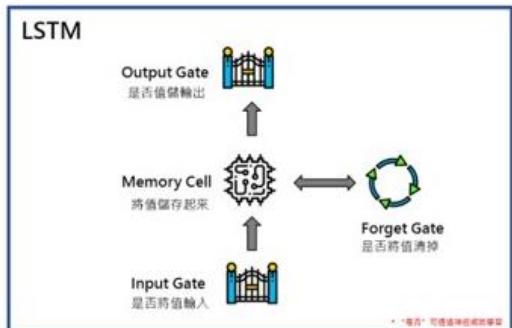
$$\begin{aligned}
 \text{Input gate: } & i_t = \sigma(W^{(ii)}\bar{x}_t + W^{(hi)}h_{t-1}) \\
 \text{Forget gate: } & f_t = \sigma(W^{(if)}\bar{x}_t + W^{(hf)}h_{t-1}) \\
 \text{Output gate: } & o_t = \sigma(W^{(io)}\bar{x}_t + W^{(ho)}h_{t-1}) \\
 \text{Process input: } & \tilde{C}_t = \tanh(W^{(i\tilde{C})}\bar{x}_t + W^{(h\tilde{C})}h_{t-1}) \\
 \text{Cell update: } & C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \\
 \text{Output: } & y_t = h_t = o_t * \tanh(C_t)
 \end{aligned}$$

LSTM 講解

LSTM 四大構成部分：

1. 輸入閘
2. 記憶單元
3. 輸出閘
4. 遺忘閘

LSTM



1. Input Gate:

當資料輸入時，input gate可以控制是否將這次的值輸入，並運算數值

2. Memory Cell:

將運算出的數值記憶起來，以利下個cell運用

3. Output Gate:

控制是否將這次計算出來的值output，若無此次輸出則為0

4. Forget Gate:

控制是否將Memory清掉(format)

←

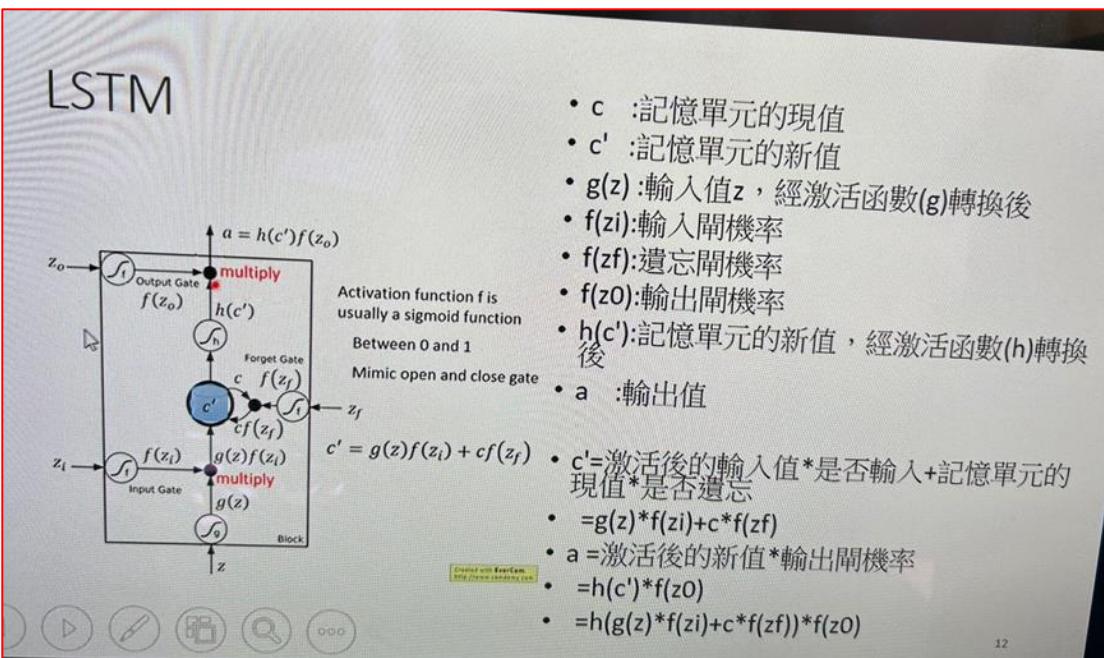
LSTM 講解

c' ：運算完只有 4 種情況

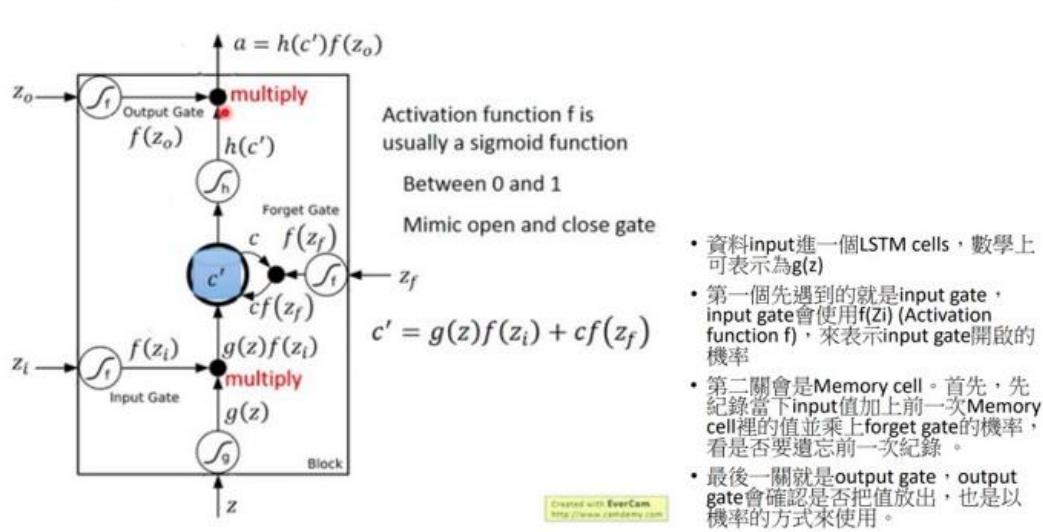
h ：上一次的 output

機率 0:不使用

機率 1:使用



LSTM 過程



RNN 與 LSTM 的應用

RNN 與 LSTM (應用方向)

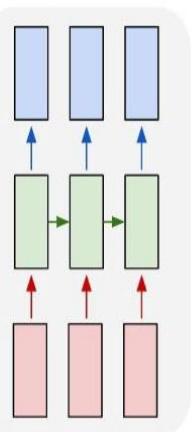
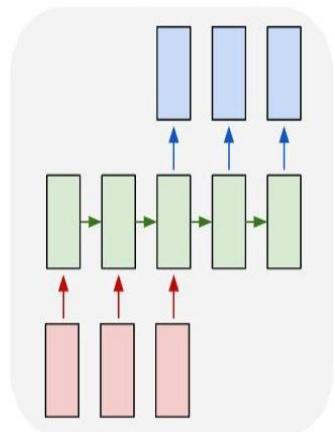
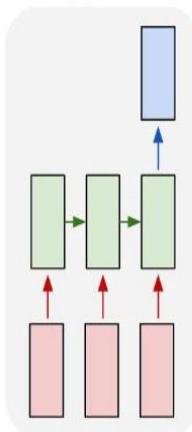
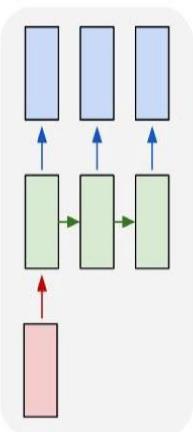
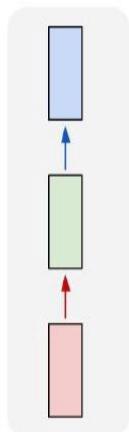
one to one

one to many

many to one

many to many

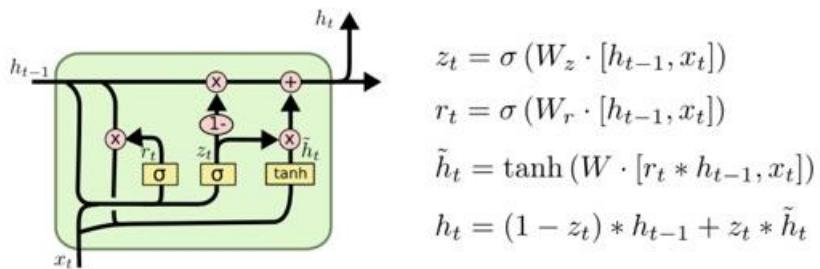
many to many



圖片來源

30

GRU



- 將 LSTM 中的遺忘閥 (forget gate) 與輸入閥 (input gate) 用一個更新閥 (update gate) 取代
- 把單元狀態 (cell state) 和隱藏狀態 (h_t) 進行合併，計算新資訊的方式和 LSTM 也有所不同

※小考

1. 回去要找什麼資料、上課教了什麼？

12/11 第十四週

●Python 演練(交叉驗證)(報告主題加入交叉驗證)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import joblib
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from keras.models import Sequential
from keras.layers import Dense
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.naive_bayes import
MultinomialNB,ComplementNB,GaussianNB,CategoricalNB,BernoulliNB
from sklearn.linear_model import LogisticRegression
import seaborn as sns

from keras.layers import Dropout
df = pd.read_csv("C:/Users/Nasser/Desktop/大三上/AI report/hotel_booking.csv") #建立訓練模型
#建立 model

#scaled
'''from sklearn.preprocessing import StandardScaler
features_to_scale = df.columns[0:1]+df.columns[2:29]
std=StandardScaler()
df[features_to_scale] = std.fit_transform(df[features_to_scale])
df=pd.DataFrame(df,columns=df.columns)'''
```

```

#df.to_csv("C:/Users/Shen/Downloads/123.csv")
#print(df)
#columns=df.columns 是將原始 DataFrame df 的列名稱傳遞給新建立的經過,確保轉換後的資料擁有原始資料相同的列名稱

selected_features = [
    'adr', 'lead_time', 'country', 'previous_cancellations','total_of_special_requests',
    'stays_in_week_nights','market_segment', 'distribution_channel',
    'is_canceled'
]
df_selected = df[selected_features]

print(df_selected.info())
print(df_selected.shape)
print(df_selected.describe())
print(df_selected.columns)

features_with_missing_values = df_selected.columns[df_selected.isnull().any()]

# 顯示有缺失值的特徵以及缺失值的總數
for feature in features_with_missing_values:
    missing_count = df_selected[feature].isnull().sum()
    print(f"特徵 {feature} 具有 {missing_count} 個缺失值")

df_selected = df_selected.dropna()

# 找出所有 object 型態的欄位
object_columns = df_selected.select_dtypes(include=['object']).columns

# 對每個 object 型態的欄位進行 Label Encoding
label_encoder = LabelEncoder()
for column in object_columns:
    df_selected[column] = label_encoder.fit_transform(df_selected[column])

print(df_selected.info())

print(df_selected.shape)
df_selected.to_csv("C:/Users/Nasser/Desktop/大三上/AI report/hotel_booking_new.csv", index=0)

#df_selected.describe()

```

```

#df_selected.columns

df_feature = df_selected.drop('is_canceled',axis=1) #捨去 label 特徵
df_label = df_selected['is_canceled']

print(df_feature)

print(df_feature.shape)
print(df_label.shape)

X = df_feature
y = df_label
X=X.to_numpy()
y=y.to_numpy()

train_feature, test_feature, train_label, test_label =
train_test_split(df_feature,df_label,test_size=0.2)

DecisionTree_model = tree.DecisionTreeClassifier()#criterion='entropy',max_depth=3
DecisionTree_model.fit(train_feature, train_label)
# print(DecisionTree_model.get_depth())

#將原本 train_data 資料， 分成訓練集與測試集合， 並丟入訓練好的模型測試準確度
print ("決策樹(Decision Trees)模型準確度(訓練集): ",DecisionTree_model.score(train_feature,
train_label))
predictions = DecisionTree_model.predict(train_feature)
print("輸出混亂矩陣,顯示準確率:使用訓練資料")
print(classification_report (train_label, predictions))

print ("決策樹(Decision Trees)模型準確度(測試集): ",DecisionTree_model.score(test_feature,
test_label))
predictions = DecisionTree_model.predict(test_feature)
print("輸出混亂矩陣,顯示準確率:使用測試資料")
print(classification_report (test_label, predictions))
print('\n' + 'confusion matrix')
print(confusion_matrix(test_label,predictions))
print('\n')

#DecisionTree_model_acc = DecisionTree_model.score(train_feature, train_label)
#Compute ROC curve and ROC area for each class

```

```

fpr, tpr, threshold =roc_curve (test_label, predictions) #計算真正率&假正率
roc_auc = auc (fpr, tpr) #計算 auc 的值
plt.figure()
lw = 2
#plt.figure(figsize=(10,10))
plt.plot(fpr, tpr, color='darkorange',
          lw=lw, label='ROC curve (area = %0.4f)' % roc_auc)
#假正率為橫坐標，真正率為縱座標做曲線
plt.plot([0,1], [0,1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('DTree ROC Curve')

#plt.legend(loc="lower right") #標籤位置
plt.legend()
plt.show()
print("\nROC_auc area-%.4f" % (roc_auc))

#lr_probs = model.predict_proba(X_test)
#print(1r_probs)
#keep probabilities for the positive outcome only
#lr_probs = lr_probs [:, 1]
# predict class values
#yhat = model.predict(X_test)

#lr_precision, lr_recall, _ = precision_recall_curve (y_test, lr_probs)
lr_precision, lr_recall, _ = precision_recall_curve (test_label, predictions)

lr_f1, lr_auc = f1_score (test_label, predictions), auc(lr_recall, lr_precision)
# summarize scores
#print("MLP (ANN): f1-%.3f PRC_auc area=%3f' % (lr_f1, lr_auc)) # f1 乃是 label=1 的 f1

print("PRC_auc area=%4f" % (lr_auc))

# plot the precision-recall curves
no_skill= len(test_label[test_label==1]) / len(test_label)

#plt.figure(figsize=(10,10))

```

```

# plt.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill')
plt.plot([0, 1], [1, 0], color='navy', lw=lw, linestyle='--')
plt.plot(lr_recall, lr_precision, color= 'darkorange',
         lw=lw, label='PRC curve (area = %0.4f) % lr_auc')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])

# axis labels
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('DTree PRC Curve')

# show the legend
plt.legend()
#show the plot
plt.show()

#交叉驗證開始
from sklearn import tree
import pandas as pd
#model tree. DecisionTreeClassifier()
#調整參數設定
# model = tree.DecisionTreeClassifier(criterion="entropy", random_state = 30, splitter=
#"random")
#Dataframe to ndarray
from sklearn.model_selection import KFold
#紀錄開始時間
import datetime
starttime = datetime.datetime.now()
n_splits = 10 #*****
kf = KFold(n_splits=n_splits, shuffle=True)

list_CM_N=list()
list_CM_Y=list()
list_reoprt=list()

i=1
for train_index, val_index in kf.split(X):

```

```
DecisionTree_model.fit(X[train_index], y[train_index])
pred = DecisionTree_model.predict(X[val_index])
print("\nCross validation #",i)
print(confusion_matrix(y[val_index], pred))
print(classification_report(y[val_index], pred))
```

```
#confusion_matrix 取值
CM=confusion_matrix(y[val_index], pred)
```

```
#預測 N
```

```
TP=CM[0][0]
```

```
FN=CM[0][1]
```

```
FP=CM[1][0]
```

```
TN=CM[1][1]
```

```
TPR=TP/(TP+FN)
```

```
TNR=TN/(TN+FP)
```

```
FPR=1-TNR
```

```
FNR=1-TPR
```

```
Precision=TP/(TP+FP)
```

```
F1=(2*TPR*Precision)/(TPR+Precision)
```

```
Accuracy=(TP+TN)/(TP+FP+FN+TN)
```

```
tupleCM_N=[TP,FN, FP, TN, TPR, TNR, FPR, FNR, Precision, F1, Accuracy, (TP+FP+FN+TN)]
list_CM_N.append(tupleCM_N)
```

```
#預測 Y,僅修改(TP,TN)對調,(FP,FN)對調
```

```
TN=CM[0][0]
```

```
FP=CM[0][1]
```

```
FN=CM[1][0]
```

```
TP=CM[1][1]
```

```
TPR=TP/(TP+FN)
```

```
TNR=TN/(TN+FP)
```

```
FPR=1-TNR
```

```
FNR=1-TPR
```

```
Precision=TP/(TP+FP)
```

```
F1=(2*TPR*Precision)/(TPR+Precision)
```

```
Accuracy=(TP+TN)/(TP+FP+FN+TN)
```

```
tupleCM_Y=[TP, FN, FP, TN, TPR, TNR, FPR, FNR, Precision, F1, Accuracy, (TP+FP+FN+TN)]
```

```

list_CM_Y.append(tupleCM_Y)

#classification_report 取值,
report=classification_report(y[val_index],pred,output_dict=True)
df=pd.DataFrame(report).transpose()
precision=np.array(df[4:5][["precision"]])
recall=np.array(df[4:5][["recall"]])
f1=np.array(df[4:5][["f1-score"]])
accuracy=np.array(df[2:3][["f1-score"]])
tupleOne=[round(precision[0],4), round(recall[0],4), round(f1[0],4), round(accuracy[0],4)]
list_reoprt.append(tupleOne)

i=i+1

#紀錄終止時間
endtime = datetime.datetime.now()
Algorithmtime = endtime - starttime
print('開始時間:',starttime)
print('演算時間:',Algorithmtime)
print('結束時間:',endtime)
print('花費時間:',endtime-starttime)

#print(list_reoprt)
#list 轉陣列,用於平均值計算
array_report=np.array(list_reoprt)
#print(array_report)
#print(array_report[:,0])
#計算總平均(Cross validation) 放置於最後一列(row)
avg_precision=round(np.average(array_report[:,0]),4)
avg_recall =round(np.average(array_report[:,1]),4)
avg_f1_score =round(np.average(array_report[:,2]),4)
avg_accuracy =round(np.average(array_report[:,3]),4)

tupleAvg=[avg_precision, avg_recall,avg_f1_score,avg_accuracy]
list_reoprt.append(tupleAvg)

from pandas import DataFrame
df_reoprt=DataFrame(list_reoprt)
df_reoprt.columns=("precision", "recall","f1-score", "accuracy")
print("計算個指標(Cross validation)的總平均,總平均放置於最後一列(row)")

```

```

print(df_reoprt)

filenameReport="C:/C_人工智慧/team1_final_project/Cross_validation/mid_Dtree_reoprt.csv"
#*****
print("輸出至檔案",filenameReport)
df_reoprt.to_csv(filenameReport, index=True)

#輸出預測的各種指標
df_CM_N=DataFrame(list_CM_N)
#tupleCM=[TP, FN, FP, TN, TPR, TNR, FPR, FNR, Precision, F1, Accuracy, (TP+FP+FN+TN)]
df_CM_N.columns=("TP", "FN", "FP", "TN", "TPR", "TNR", "FPR", "FNR", "Precision", "F1", "Accuracy",
"Total")
print(df_CM_N)

filenameCM_N="C:/Users/Nasser/Desktop/大三上/AI report/mid_Dtree_CM_N.csv"
print("confusion_matrix 輸出至檔案",filenameCM_N)
df_CM_N.to_csv(filenameCM_N, index=True)

#輸出預測 Y 的各種指標
df_CM_Y=DataFrame(list_CM_Y)
#tupleCM=[TP, FN, FP, TN, TPR, TNR, FPR, FNR, Precision, F1, Accuracy, (TP+FP+FN+TN)]
df_CM_Y.columns=("TP", "FN", "FP", "TN", "TPR", "TNR", "FPR", "FNR", "Precision", "F1", "Accuracy",
"Total")
print(df_CM_Y)

filenameCM_Y="C:/Users/Nasser/Desktop/大三上/AI report/mid_Dtree_CM_Y.csv"
print("confusion_matrix 輸出至檔案",filenameCM_Y)
df_CM_Y.to_csv(filenameCM_Y, index=True)

```

※小考

無小考

12/18 第十五週

● 重點整理 1

濾鏡的數量建議

- 濾鏡越多越能辨識複雜的特徵，但耗費更多的計算
- 越後面的卷積層用的濾鏡越多
 - 前面卷積層用於辨識簡單的特徵；後面的卷積層將簡單的特徵組合成複雜的特徵
- (模型越簡單越好) 計算複雜度越低越好，濾鏡能少則少
 - 若驗證模型的損失值，會因為濾鏡增加而降低，則濾鏡加倍($8 \rightarrow 16$)
 - 若驗證模型的損失值，不會因為濾鏡減少而降低，則濾鏡減半($16 \rightarrow 8$)

※ 卷積層：找重要特徵；池化層：特徵放大

CNN

CNN 資料及圖片檔名：前面數字 1 車頭 / 2 車尾

梯度消失：當 CNN 層數很多時，會發生

梯度爆炸：當在反向傳播(反向調整權重)過程中，神經網路的層數較多時，較靠近輸入層的梯度可能會變得非常小，甚至接近於零。這導致輸入層的權重幾乎不會更新(因為反向最後)，輸出端更新比較快，從而使得模型無法有效的學習。

梯度消失的情況通常是由於反向傳播過程中，多次乘積的梯度導致的。

● 重點整理 2

1. 梯度消失原因：在深度學習神經網絡中，隨著層數的增加，往往會遇到梯度不穩定的問題，值將會以指數形式越來越小，最後淺層(靠近 input 層)的權重變動會越來越少，這就是梯度消失原因：(1) 隱藏層的層數過多；(2) 採用了不合適的激活函數(更容易產生梯度消失，但是也有可能產生梯度爆炸)

2. 梯度爆炸原因：梯度爆炸則是剛好相反，每一項其值都大於 1，最後的值將隨著層數的增加以指數形式越變越大，越是靠近淺層(靠近 input 層)的權重變動會越來越大，當權值過大時，神經網絡中淺層比深層的梯度變化更快，就會引起梯度爆炸問題。

原因：(1) 隱藏層的層數過多；(2) 權重的初始化值過大

解決方式：

3. 梯度消失與梯度爆炸的解決方案

梯度消失和梯度爆炸問題都是因為網路太深，網路權值更新不穩定造成的。

對於較普遍的梯度消失問題，可以考慮三種方案解決：

- (1) 以 ReLU、Leaky-ReLU、P-ReLU、R-ReLU、Maxout 等取代 sigmoid 函數。
 - (2) 用 Batch Normalization。(正規化)可以加快梯度下降的求解速度，而且在一定程度緩解了深層網絡中梯度消失的問題，從而使得訓練深層網絡模型更加容易和穩定地進行梯度傳播。
 - (3) LSTM 的結構設計也可以改善 RNN 中的梯度消失問題。
4. 權重初始化：使用適當的權重初始化方法(例如:Xavier or He 初始方法)可以幫助防止梯度消失或爆炸
5. 激活函數的選擇：使用較少會導致梯度消失的激活函數，如 ReLu 或其變數
6. 批量標準化(batch normalization)：通過批量標準化層可以幫助維持梯度的大小，有助於防止梯度消失或爆炸
7. 梯度剪裁：將梯度限制在一定範圍內，防止梯度爆炸
8. 使用遞歸神經網路(RNN)或注意力機制：這些結構可以幫助減少梯度消失的影響

12/25 第十六週

●各組期末報告

期末報告：調整之參數中那些是最有必要調整的(影響大)

本組報告：應用機器學習模型於預測飯店訂房取消率及因素分析

●課堂重點

1. 隨機森林中決策樹數量奇數偶數之差別

A：因為從每棵決策樹的預測結果，進行多數決投票，決定最終預測的類別(label)(避免出現平票)

2. KNN 中近鄰點選擇奇數偶數之差別

A：KNN 根據群中大多數屬於哪一類即預測為那一類，但如果使用 distance 加權權重法則奇數偶數影響較小(當 k 為偶數，兩類資料點數量相等，一樣有平票問題)

3. 決策樹節點顏色差別

A：節點顏色越深，代表該節點(特徵)的 survived 機率越高(事件發生機率越高)，同顏色通常代表不同的類別或類別的混合。使用 plota tree 函數中的 class.names 參數，你可以為每個類別指定一個顏色。當繪製決策樹時，如果某個節點包含多個類別的樣本，則它可能會以混合顏色的方式表示這些類別的組合。

每個 Node 節點上的數值分別代表：

1. 預測類別(0, 1)
2. 預測目標類別的機率(1 的機率)
3. 節點中觀測資料個數佔比

2024/1/1 第十七週

元旦放假

2024/1/8 第十八週

總結及計算