深度學習

第 6 章 神經網路實作篇

講師：紀俊男

本章大綱

- 迴歸問題
- 分類問題（二選一）
- 分類問題（多選一）

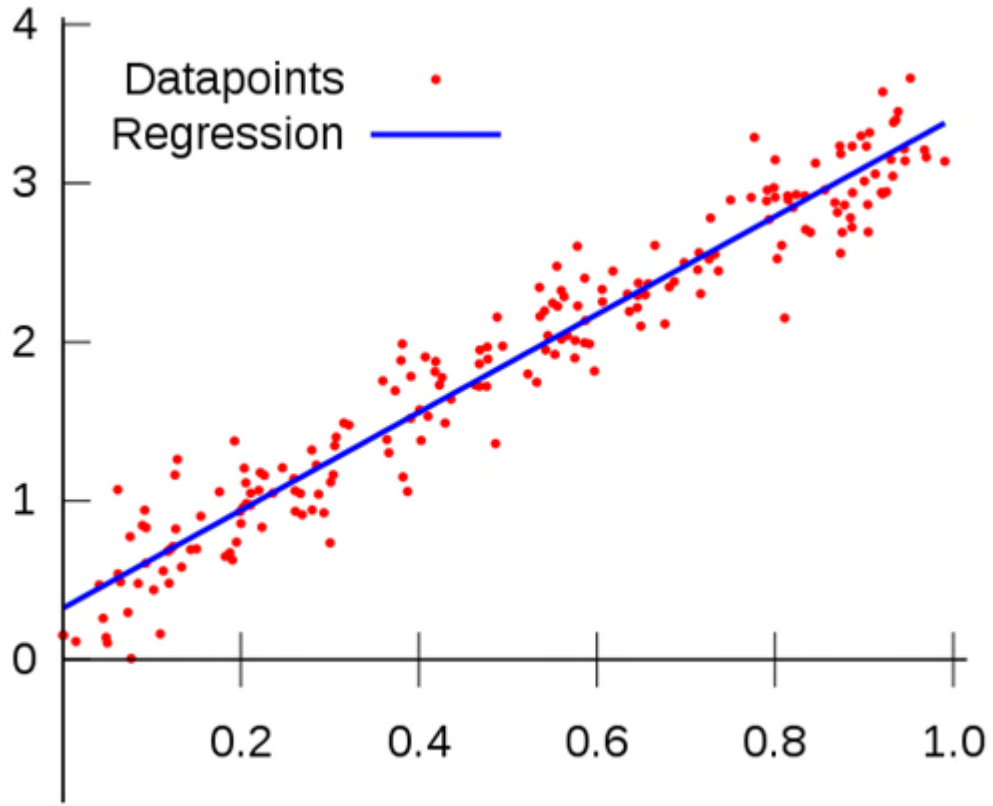
AI

迴歸問題
Regression

範例完整原始碼：
https://lurl.cc/vyi88r

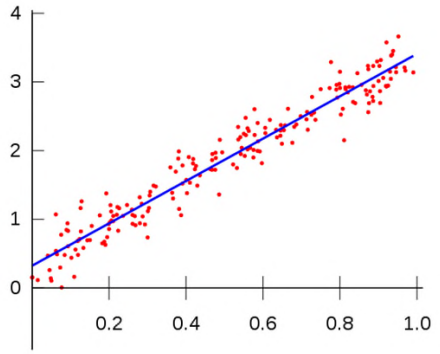
何謂「迴歸」（Regression）

- 利用「過往資料」，計算「近似線型」，並用於「預測未來」的方法



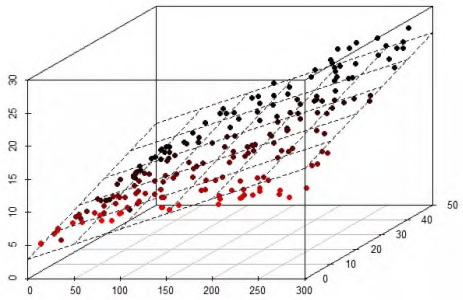
迴歸的種類

$f(x) = c + mx$



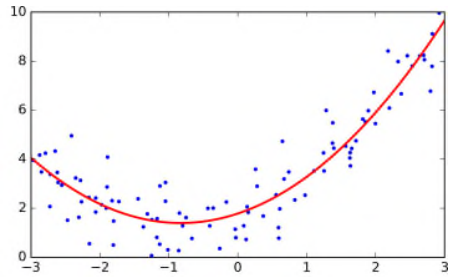
簡單線性迴歸
(Simple Linear Regression)

$f(x, y, z, \ldots) = c_0 + c_1X + \cdots c_nZ$



多元線性迴歸
(Multiple Linear Regression)

$f(x) = c_0 + c_1X^1 + \cdots c_nX^n$



多項式迴歸
(Polynomial Regression)

神經網路都可以用「多層神經元」架構，一視同仁地處理

問題簡介：加州房價預測

- 利用收入中位數、屋齡中位數...，預測類似地區之房價

20640筆

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	Price
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422
...	...	...	...	...	...	...	...	...	...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09	0.781
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21	0.771
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22	0.923
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32	0.847
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24	0.894

自變數 X

應變數 Y

自變數說明：

MedInc：社區收入中位數
HouseAge：社區房齡中位數
AveRooms：每戶平均房間數
AveBedrms：每戶平均臥室數
Population：社區人口
AveOccup：家庭平均成員數
Latitude：社區所在緯度
Longitude：社區所在經度

應變數說明：

Price：社區房價中位數
（百萬美元）

Colab 環境準備

環境準備

安裝快樂版函式庫

```
[2]  1  # 先檢查是否存在 HappyML 這個資料夾，若沒有，則下載
     2  import os
     3
     4  if not os.path.isdir("HappyML"):
     5      os.system("git clone https://github.com/cnchi/HappyML.git")
```

從 **GitHub** 下載安裝 **HappyML** 函式庫

隨堂練習

- 執行前，請先用「編輯 > 筆記本設定」，將硬體加速器選擇「GPU」，以加速神經網路的訓練速度。
- 請試著輸入前一頁的程式碼，並執行看看。
- 執行完畢後，請到「檔案」面板，確認 HappyML 資料夾是否已經下載下來了？

載入資料集

- 加州房價資料庫

```
1  # Load Dataset
2  from sklearn.datasets import fetch_california_housing
3  dataset = fetch_california_housing()
```

Key	Type	Size	Value
data	Array of float64	(20640, 8)	[[8.3252 41. 6.9841269 37.8 ...
DESCR	str	1776	.. _california_housing_dataset:
feature_names	list	8	['MedInc', 'HouseAge', 'AveRooms', 'AveBedr 'Population', 'AveOccu ...
frame	NoneType	1	NoneType object
target	Array of float64	(20640,)	[4.526 3.585 3.521 ... 0.923 0.847 0.894]
target_names	list	1	['MedHouseVal']

您可用下列指令，了解資料集的大概樣貌：

print(dataset)：列出 dataset 有哪些東西
print(dataset.keys())：可以列出 dataset 有哪些欄位
print(dataset["DESCR"])：漂亮列出 dataset 的說明
print(dataset["data"])：列出自變數 X 資料的大致情況
print(dataset["target"])：列出應變數 Y 資料的大致情況
print(dataset["feature_names"])：印出自變數 X 所有欄位名稱

隨堂練習

- 請試著輸入前一頁的程式碼，並執行看看。
- 請將前一頁的那些指令，都在 Scratch Code Cell 輸入看看。藉以了解資料集的大概樣貌。
- 對照講義前幾頁提供的欄位解釋，您是否能對資料集有一定認識了呢？

資料集前處理

- Step 1：切分自變數 X 與應變數 Y：

```
1  import HappyML.preprocessor as pp
2
3  # Decomposition dataset as X and Y
4  import pandas as pd
5  X = pd.DataFrame(dataset.data, columns=dataset.feature_names)
6  Y = pd.DataFrame(dataset.target, columns=["Target"])
```

您可用下列指令，更了解資料集的細節：

print(X)：可以看到包裝好的自變數 X 樣貌
print(Y)：可以看到包裝好的應變數 Y 樣貌
print(X.info())：觀看 X 的欄位名稱、數量、是否缺值、型態
print(X.describe())：觀看 X 各欄位的平均、標準差、極大極小、分位數

隨堂練習

- 請試著輸入前一頁的程式碼，並執行看看。
- 請將前一頁的那些指令，都在 Scratch Code Cell 輸入看看。藉以更了解資料集的細節。
- 尤其是 print(X.info()) 可以了解到是否要執行「缺值補遺」，print(X.describe()) 可以了解到是否要執行「特徵縮放」

資料集前處理

- Step 2：切分訓練集、測試集 與特徵縮放

```
1  # Split Training vs. Testing Set
2  X_train, X_test, Y_train, Y_test = pp.split_train_test(X, Y, train_size=0.8)
3
4  # Feature Scaling (optional)
5  X_train, X_test = pp.feature_scaling(fit_ary=X_train, transform_arys=(X_train, X_test))
```

您可用下列指令，了解
程式碼是否執行成功：

```
print(X_train)
print(Y_train)
print(X_test)
print(Y_test)
```

隨堂練習

- 請試著輸入前一頁的程式碼，並執行看看。
- 請將前一頁的那些指令，都在 Scratch Code Cell 輸入看看。以確定程式碼執行成功。

建構神經網路各層級

- 程式碼

```
1 from tensorflow.keras.models import Sequential
2 from tensorflow.keras.layers import Dense
3
4 # Add layers with .add() function
5 model = Sequential()
6
7 model.add(Dense(input_dim=X_train.shape[1], units=18, kernel_initializer="glorot_normal", activation="relu"))
8 model.add(Dense(units=13, kernel_initializer="glorot_normal", activation="relu"))
9 model.add(Dense(units=1, kernel_initializer="glorot_normal", activation="linear"))
```

您可用下列指令，確定神經網路已經建立成功：

print(model.summary())：了解神經網路各層級架構。
print(model.inputs)：了解「輸入層」架構。
print(model.outputs)：了解「輸出層」架構。

建構神經網路各層級

- 也可以這麼寫：

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Add layers with a list (middle-click to uncomment multiple lines)
model = Sequential(
    [Dense(input_dim=X_train.shape[1], units=18, kernel_initializer="glorot_normal", activation="relu", name="input_hidden1"),
    Dense(units=13, kernel_initializer="glorot_normal", activation="relu", name="hidden2"),
    Dense(units=1, kernel_initializer="glorot_normal", activation="linear", name="output")])
```

隨堂練習

- 請試著輸入前一頁的程式碼，並執行看看。
- 請將前兩頁的那些指令，都在 Scratch Code Cell 輸入看看。以確定神經網路已經建置成功。

模型編譯

- **一般寫法**

```
1  # Compile the whole Neural Networks
2  model.compile(optimizer="adam", loss="mse", metrics=["mse"])
```

- 如果你想在每個**週期**，監督更多**效能指標（Metrics）**的話：

```
1  # Add more metrics
2  from tensorflow.keras import metrics
3  model.compile(optimizer="adam", loss="mse", metrics=["mse", metrics.RootMeanSquaredError()])
```

補充：迴歸效能公式解說

均方誤差
(Mean Squared Error, MSE)

真實值　　預測值

$$MSE = \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n}$$

說明：
- 殘差平方的平均

優點：
- 微分方便

缺點：
- 數量級變平方倍（一百 --> 一萬）

均方根誤差
(Root Mean Squared Error, RMSE)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n}}$$

說明：
- $\sqrt[2]{\text{殘差平方的平均}}$

優點：
- 數量級不變（一百 --> 一百）

缺點：
- 微分稍微不方便

隨堂練習

- 請試著輸入前一頁的一般程式碼，並執行看看。
- 額外的效能指標，我們會留待後面的練習再做。請現在使用一般的寫法即可。

模型訓練

- 程式碼

切出 20% 的訓練集，不參與訓練。
做為「驗證集」，驗證「超參數」正確與否之用。

```
1  # Fit (Training)
2  epochs_metrics = model.fit(x=X_train, y=Y_train, validation_split=0.2, batch_size=5, epochs=50)
```

超參數

您可用下列指令，看到每個回合（一期）的效能歷史：

```
print(epochs_metrics.history)
```

- **訓練集**：訓練模型之用。
- **驗證集**：調整模型超參數之用。
- **測試集**：測試訓練出來的模型好壞。

隨堂練習

- 請試著輸入前一頁的程式碼，並執行看看。
- 請將前一個「模型編譯」的程式碼，加上額外的效能指標，再跑一次看看。
- 您看到每一期新增的效能指標了嗎？
- 為了後面的練習，請再次改回模型編譯的「一般寫法」，重新執行看看。

模型校正

- 請先製作下列的圖形繪製副程式：

```python
import matplotlib.pyplot as plt

def plot(history_dict, keys=(), title=None, xyLabel=[], ylim=(), size=()):
    lineType = ("-", "--", ".", ":")
    if len(ylim)==2:
        plt.ylim(*ylim)
    if len(size)==2:
        plt.gcf().set_size_inches(*size)
    epochs = range(1, len(history_dict[keys[0]])+1)
    for i in range(len(keys)):
        plt.plot(epochs, history_dict[keys[i]], lineType[i])
    if title:
        plt.title(title)
    if len(xyLabel)==2:
        plt.xlabel(xyLabel[0])
        plt.ylabel(xyLabel[1])
    plt.legend(keys, loc="best")
    plt.show()
```

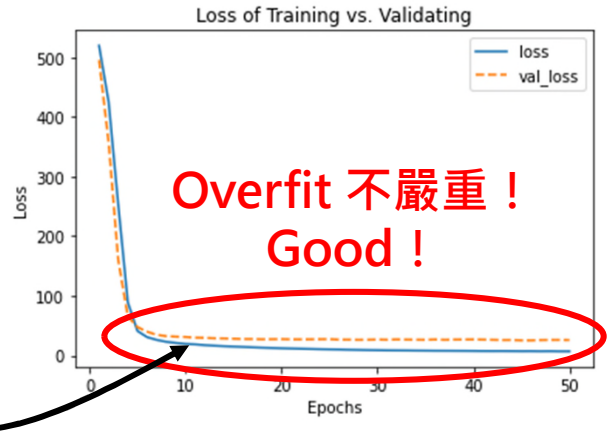
模型校正

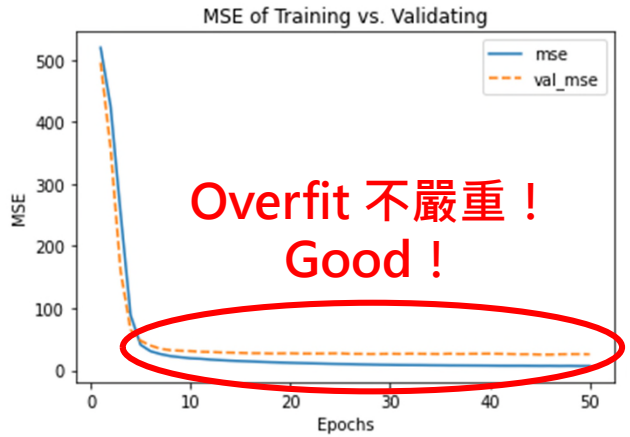
- 接著用下面程式碼，繪製出每一期的「損失值」與「MSE」值：

```
plot(epochs_metrics.history, keys=("loss", "val_loss"),
    title="Loss of Training vs. Validating", xyLabel=("Epochs", "Loss"))
plot(epochs_metrics.history, keys=("mse", "val_mse"),
    title="MSE of Training vs. Validating", xyLabel=("Epochs", "MSE"))
```

- 這是繪製出來的圖形：

約 Epochs = 10
即可停止訓練





隨堂練習

- 請試著輸入前一頁的程式碼，並執行看看。
- 您能從圖形中看出，是否有 Overfit 嗎？說說看您是如何判斷的？
- 依照繪製出來的圖形，請重新調整 Epochs 這個超參數，以便下次訓練時，能夠快速訓練完畢。
- 為了將來方便，上述程式碼已經納入 HappyML.model_drawer

中，函數名稱為 epochs_metrics_plot()。您可以試試看用下列方法呼叫：

```
1  import HappyML.model_drawer as md
2  md.epochs_metrics_plot(epochs_metrics.history, keys=("loss", "val_loss"), title="Loss of Training vs. Validating", xyLabel=("Epochs", "Loss"))
3  md.epochs_metrics_plot(epochs_metrics.history, keys=("mse", "val_mse"), title="MSE of Training vs. Validating", xyLabel=("Epochs", "MSE"))
```

預測答案

- 程式碼

```
1  # Predict
2  import pandas as pd
3  Y_pred = pd.DataFrame(model.predict(x=X_test), index=Y_test.index, columns=Y_test.columns)
4
5  # Show the Predict Result
6  df = pd.concat([Y_test, Y_pred], axis=1)
7  print(df)
```

- 執行結果

真實值

	Target	Target
111	22.8	25.584444
223	30.1	27.337389
458	14.9	14.523902
92	22.9	24.935705
495	23.1	15.739724
..	...	...
193	31.1	32.687866
109	19.4	17.989935
210	21.7	20.566086
454	14.9	12.947841
469	20.1	14.491833

[102 rows x 2 columns]

預測值

模型評估

- 程式碼

```
# Evaluation
test_loss, test_mse = model.evaluate(X_test, Y_test)
print("Loss of Testing Set:", test_loss)
print("MSE of Testing Set:", test_mse)
```

- 執行結果

```
4/4 [==============================] - 0s 2ms/step - loss: 24.1148 - mse: 24.1148
Loss of Testing Set: 24.114789962768555
MSE of Testing Set: 24.114789962768555
```

隨堂練習

- 請試著輸入前一頁的程式碼，並執行看看。
- 您的模型有多好呢？

課後作業： TOYOTA 二手車價格預測

- **英國 TOYOTA 二手車商，收集了 6738 筆二手車交易記錄。欄位包括：**
  - **model**：車種。如：Yaris、Corolla...等。
  - **year**：註冊日期。如：2017、2018...。
  - **transmission**：排檔模式。如 Manual（手排）、Automatic（自排）、Semi-Auto（手自排）。
  - **mileage**：里程數。單位是「英哩（mile）」。
  - **fuelType**：適用油品。如 Petrol（汽油）、 Diesel（柴油）、 Hybrid（油電混合）。
  - **tax**：每年稅金。單位為英鎊。
  - **mpg**：油耗表現（Miles Per Gallon）。單位為「加侖/每英哩」。
  - **engineSize**：引擎升數。如 2.0、1.8...等。
  - **price**：車價。此為「**應變數**」。單位為「英鎊」。

- **請用神經網路撰寫一個「迴歸模型」。能做到：**
  - 訓練時，必須繪製出每個週期「**訓練集**」與「**驗證集**」的 **Loss** 與 **MSE** 差異，藉以了解是否**過擬合**。
  - 印出測試集中的「**真實值**」與「**預測值**」，好讓人感受此模型的準度。
  - 印出此模型的 **MSE**，以客觀說明該模型的**效能**。

課後作業： TOYOTA 二手車價格預測

- 提示（１）：用下列程式碼，安裝 HappyML 函式庫與資料集。

```
# Install HappyML Library
import os

if not os.path.isdir("HappyML"):
  os.system("git clone https://github.com/cnchi/HappyML.git")
```

```
# Upload Dataset from GitHub Repository
Dataset_File = 'ToyotaUsedCars.csv'
if not os.path.isfile(Dataset_File):
  os.system('wget https://raw.githubusercontent.com/cnchi/datasets/master/' + Dataset_File)

```

課後作業： TOYOTA 二手車價格預測

- 提示（2）：使用 HappyML 做資料前處理參考程式碼

▾ 資料集載入

```
[5]  1  import HappyML.preprocessor as pp
     2  dataset = pp.dataset(file=Dataset_File)
```

▾ 切分自變數 X 與應變數 Y

```
[6]  1  X, Y = pp.decomposition(dataset=dataset, x_columns=[i for i in range(8)], y_columns=[8])
```

▾ 類別資料數位化

```
[7]  1  X = pp.onehot_encoder(X, columns=[0, 2, 4], remove_trap=True)
```

▾ 切分訓練集、測試集

```
[8]  1  X_train, X_test, Y_train, Y_test = pp.split_train_test(X, Y, train_size=0.8)
```

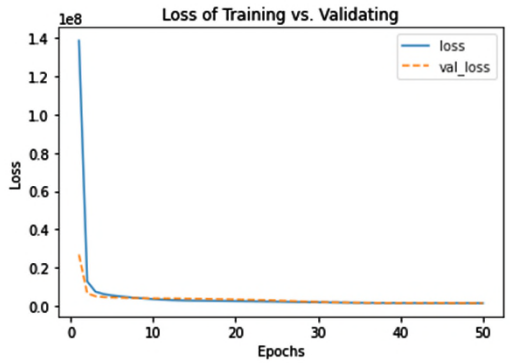
▾ 特徵縮放

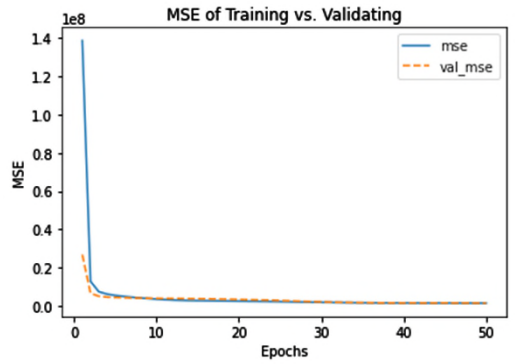
```
[9]  1  X_train, X_test = pp.feature_scaling(fit_ary=X_train, transform_arys=(X_train, X_test))
```

課後作業： TOYOTA 二手車價格預測

- 參考輸出

```
Epoch 1/50
863/863 [==============================] - 1s 2ms/step - loss: 138599744.0000 - mse: 138599744.0000 - val_loss: 26860952.0000 - val_mse: 26860952.0000
Epoch 2/50
863/863 [==============================] - 1s 2ms/step - loss: 12874949.0000 - mse: 12874949.0000 - val_loss: 6606613.5000 - val_mse: 6606613.5000
Epoch 3/50
863/863 [==============================] - 1s 2ms/step - loss: 7473052.5000 - mse: 7473052.5000 - val_loss: 5130436.5000 - val_mse: 5130436.5000
...
Epoch 48/50
863/863 [==============================] - 1s 1ms/step - loss: 1531635.2500 - mse: 1531635.2500 - val_loss: 1557402.0000 - val_mse: 1557402.0000
Epoch 49/50
863/863 [==============================] - 1s 1ms/step - loss: 1520778.6250 - mse: 1520778.6250 - val_loss: 1512498.5000 - val_mse: 1512498.2500
Epoch 50/50
863/863 [==============================] - 1s 1ms/step - loss: 1509236.1250 - mse: 1509235.8750 - val_loss: 1517111.6250 - val_mse: 1517111.6250
```





	price	price
3354	9990	10529.969727
3404	1599	2419.139160
5865	15110	17875.341797
4894	6950	6335.562500
2980	8498	8309.142578
...	...	...
2587	11975	12205.828125
4473	6799	7927.747559
2483	8099	8728.291016
5917	23000	20318.523438
1555	12498	13183.396484

[1348 rows x 2 columns]

```
43/43 [==============================] - 0s 1ms/step - loss: 1857699.7500 - mse: 1857699.7500
Loss of Testing Set: 1857699.75
MSE of Testing Set: 1857699.75
```

分類問題（二選一）
Binary Classification

範例完整原始碼：
https://lurl.cc/mqJbF2

何謂「分類問題」

- 迴歸問題

年齡	月薪	購買額
19	19000	0
32	150000	15638
25	33000	0
47	25000	4752
45	26000	1244
46	28000	3677
32	18000	0
18	82000	0
47	49000	6221
48	41000	4576
45	22000	598
35	65000	0

連續數字

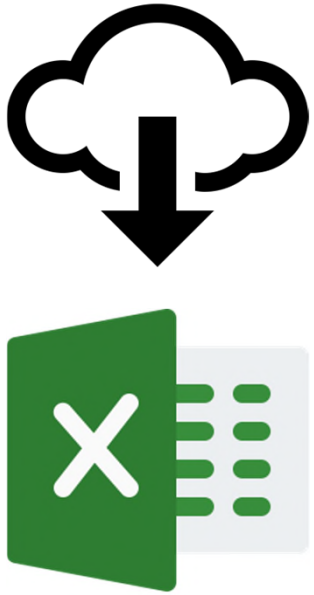
- 分類問題

年齡	月薪	是否購買
19	19000	0
32	150000	1
25	33000	0
47	25000	1
45	26000	1
46	28000	1
32	18000	0
18	82000	0
47	49000	1
48	41000	1
45	22000	1
35	65000	0

離散數字

分類問題 = 應變數 Y 是離散數字的問題

問題簡介：蘑菇可吃嗎？



Mushroom.csv

	A	B	C	D
1	class	cap-shape	cap-surface	cap-color
2	p	x	s	n
3	e	x	s	y
4	e	b	s	w
5	p	x	y	w
6	e	x	s	g
7	e	x	y	y
8	e	b	s	w
9	e	b	y	w
10	p	x	y	w

目的：利用「傘帽、表皮、傘色...」
--> 推算「香菇是否有毒」

- class：p-有毒、e-可吃
- cap-shape：傘帽
  b-鐘形、c-錐形、x-圓形、
  f-扁形、k-把手形、s-凹陷
- cap-surface：傘皮
  f-纖維感、g-凹洞、s-鱗片
  s-光滑
- cap-color：傘色
  n-棕、b-淺黃、c-肉桂、
  g-灰、r-綠、p-粉紅、u-紫
  e-紅、w-白、y-黃
- bruises：瘀傷（t-有/f-無）
- odor：氣味。a-杏仁、l-茴香
  c-雜酚油味、y-魚腥味、f-臭
  m-霉、n-無、p-辛、s-辣
- .....
- 完整列表：
  https://is.gd/F6sEBA

Colab 環境設定

安裝 HappyML 函式庫

```
[ ]  1  # Install HappyML Library
     2  import os
     3
     4  if not os.path.isdir("HappyML"):
     5    os.system("git clone https://github.com/cnchi/HappyML.git")
```

從 **GitHub** 下載安裝 **HappyML** 函式庫

下載資料集

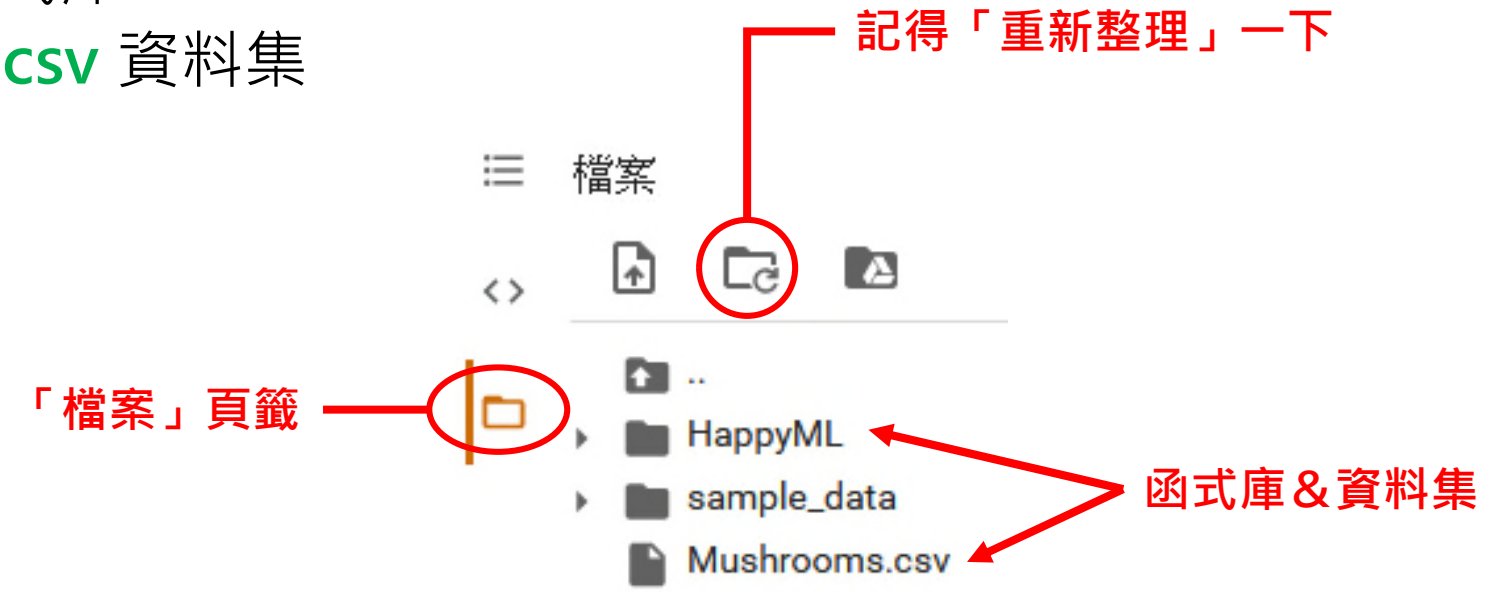
```
[ ]  1  # Upload Dataset from GitHub Repository
     2  Dataset_File = 'Mushrooms.csv'
     3  if not os.path.isfile(Dataset_File):
     4    os.system('wget https://raw.githubusercontent.com/cnchi/datasets/master/' + Dataset_File)
```

從 **GitHub** 下載 **Mushrooms** 資料集

隨堂練習：環境設定

- 請先撰寫好前一頁投影片的原始碼。

- 執行看看，是否能看到左側欄「檔案」頁籤，出現下列檔案：
  - HappyML 函式庫
  - Mushrooms.csv 資料集



資料前處理：載入資料集

載入資料集

```
[ ]  1  import HappyML.preprocessor as pp
     2
     3  # Load Data
     4  dataset = pp.dataset(file="Mushrooms.csv")
     5
```

dataset 變數的內容

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	stalk-root	stalk-surface-above-ring	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population	habitat
0	p	x	s	n	t	p	f	c	n	k	e	e	s	s	w	w	p	w	o	p	k	s	u
1	e	x	s	y	t	a	f	c	b	k	e	c	s	s	w	w	p	w	o	p	n	n	g
2	e	b	s	w	t	l	f	c	b	n	e	c	s	s	w	w	p	w	o	p	n	n	m
3	p	x	y	w	t	p	f	c	n	n	e	e	s	s	w	w	p	w	o	p	k	s	u
4	e	x	s	g	f	n	f	w	b	k	t	e	s	s	w	w	p	w	o	e	n	a	g
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8119	e	k	s	n	f	n	a	c	b	y	e	?	s	s	o	o	p	o	o	p	b	c	l
8120	e	x	s	n	f	n	a	c	b	y	e	?	s	s	o	o	p	n	o	p	b	v	l
8121	e	f	s	n	f	n	a	c	b	n	e	?	s	s	o	o	p	o	o	p	b	c	l
8122	p	k	y	n	f	y	f	c	n	b	t	?	s	k	w	w	p	w	o	e	w	v	l
8123	e	x	s	n	f	n	a	c	b	y	e	?	s	s	o	o	p	o	o	p	o	c	l

8124 rows × 23 columns

隨堂練習：載入資料集

- 請撰寫好、並執行前一頁投影片中的程式碼。
- 用 Insert > Scratch Cell，插入一個「草稿儲存格」。
- 輸入 dataset + Ctrl-Enter，觀看 dataset 變數，是否跟您想像的內容一樣？
- 使用下列指令，了解 dataset 是否缺值？是否需要類別資料數位化？是否需要特徵縮放？
  - dataset.info()
  - dataset.describe()

資料前處理：切分自變數、應變數

切分自變數與應變數

```
[ ]  1  # Decomposition
     2  X, Y = pp.decomposition(dataset, x_columns=[i for i in range(1, 23)], y_columns=[0])
     3
```

自變數 X

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color
0	x	s	n	t	p	f	c	n	k
1	x	s	y	t	a	f	c	b	k
2	b	s	w	t	l	f	c	b	n
3	x	y	w	t	p	f	c	n	n
4	x	s	g	f	n	f	w	b	k
...	...	...	...	...	...	...	...	...	...
8119	k	s	n	f	n	a	c	b	y
8120	x	s	n	f	n	a	c	b	y
8121	f	s	n	f	n	a	c	b	n
8122	k	y	n	f	y	f	c	n	b
8123	x	s	n	f	n	a	c	b	y

8124 rows × 22 columns

應變數 Y

	class
0	p
1	e
2	e
3	p
4	e
8119	e
8120	e
8121	e
8122	p
8123	e

8124 rows × 1 columns

隨堂練習：切分自變數、應變數

- 請撰寫、並執行前一頁投影片中的程式碼。

- 用「草稿儲存格」觀看變數 X 與 Y，是否跟您想像的內容一樣？

資料前處理：類別資料數位化

類別資料數位化

```
1  # Categorical Data
2  X = pp.onehot_encoder(X, columns=[i for i in range(22)], remove_trap=True)
3  Y = pp.label_encoder(Y)
4
```

光「shape」就
分裂成 (6-1) 種
（含移除「虛擬變數陷阱」）

	cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	cap-shape_x
0	0	0	0	0	1
1	0	0	0	0	1
2	0	0	0	0	0
3	0	0	0	0	1

應變數 Y 也從
「e/p」變成「1/0」

	class
0	1
1	0
2	0
3	1

隨堂練習：類別資料數位化

- 請撰寫、並執行前一頁投影片中的程式碼。

- 用「草稿儲存格」觀看變數 X 與 Y，是否跟您想像的內容一樣？

資料前處理：切分訓練集、測試集

切分訓練集、測試集

```
[ ]  1  # Split Training / Testing Set
     2  X_train, X_test, Y_train, Y_test = pp.split_train_test(x_ary=X, y_ary=Y)
     3
```

X_train

	cap-shape_c	cap-shape_k	cap-shape_s
2704	0	0	0
3443	0	0	0
2719	0	0	0
2353	0	0	0
4602	0	0	0
...	...	...	...

Y_train

	class
2704	0
3443	1
2719	0
2353	0
4602	1
...	...

X_test

	cap-shape_c	cap-shape_k	cap-shape_s
1815	0	0	0
625	0	0	0
4710	0	0	0
5519	0	0	0
6367	0	0	0
...	...	...	...

Y_test

	class
1815	0
625	0
4710	1
5519	1
6367	1
...	...

隨堂練習：類別資料數位化

- 請撰寫、並執行前一頁投影片中的程式碼。

- 用「草稿儲存格」觀看變數 X_train、 X_test、 Y_train、與 Y_test，是否跟您想像的內容一樣？

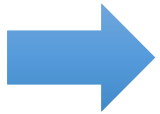
資料前處理：特徵縮放

特徵縮放

```
[ ]  1  # Feature Scaling
     2  X_train, X_test = pp.feature_scaling(fit_ary=X_train, transform_arys=(X_train, X_test))
     3
```

	cap-shape_c	cap-shape_k	cap-shape_s	cap-surface_g
2704	0	0	0	0
3443	0	0	0	0
2719	0	0	0	0
2353	0	0	0	0
4602	0	0	0	0
...	...	...	...	...
6988	0	0	0	0
6789	0	0	0	0
1073	0	0	0	0
2218	0	0	0	0
5859	0	1	0	0

6093 rows × 90 columns



	cap-shape_c	cap-shape_k	cap-shape_s	cap-surface_g
2704	-0.02563	-0.339590	-0.062885	-0.022195
3443	-0.02563	-0.339590	-0.062885	-0.022195
2719	-0.02563	-0.339590	-0.062885	-0.022195
2353	-0.02563	-0.339590	-0.062885	-0.022195
4602	-0.02563	-0.339590	-0.062885	-0.022195
...	...	...	...	...
6988	-0.02563	-0.339590	-0.062885	-0.022195
6789	-0.02563	-0.339590	-0.062885	-0.022195
1073	-0.02563	-0.339590	-0.062885	-0.022195
2218	-0.02563	-0.339590	-0.062885	-0.022195
5859	-0.02563	2.944729	-0.062885	-0.022195

6093 rows × 90 columns

隨堂練習：類別資料數位化

- 請撰寫、並執行前一頁投影片中的程式碼。
- 用「草稿儲存格」觀看變數 X_train、 X_test，是否跟您想像的內容一樣？
- 想想看：原始特徵每一欄的數量級都一樣，您覺得這種狀況需要特徵縮放嗎？

模型建置：標準寫法

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Initialize the whole Neural Networks
model = Sequential()

# Add the Input & First Hidden Layer
model.add(Dense(input_dim=X_train.shape[1], units=45, kernel_initializer="glorot_normal", activation="relu"))

# Add the Second Hidden Layer
model.add(Dense(units=23, kernel_initializer="glorot_normal", activation="relu"))

# Add the Output Layer
model.add(Dense(units=1, kernel_initializer="glorot_normal", activation="sigmoid"))

# Compile the whole Neural Networks
model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["acc"])
```

1 節點數計算

$$\frac{8124}{2 \times (90+1)} \cong 45$$

2 權重初始化
（→ 有更好的選擇嗎？）

3 二選一激活函數
（一定要用這個）

4 優化器
（→ 有更好的選擇嗎？）

5 二選一損失函數
（一定要用這個）

6 評估標準
（可自由添加多個標準）

模型建置：HappyML 寫法

- 函數定義

```
1  def create_seq_model(nodes=[], weight_init="glorot_normal",
2                       hidden_activation="relu", opt_name="adam", metric_list=["acc"],
3                       output_activation="softmax", loss_name="categorical_crossentropy"):
4      # Create Sequential Model
5      model = Sequential()
6
7      if nodes != []:
8          # Create Input Layer
9          model.add(layers.InputLayer(input_shape=nodes[0], name="input"))
10
11         # Create Hidden Layers
12         for i in range(1, len(nodes)-1):
13             hidden_name = "hidden_{}".format(i)
14             model.add(layers.Dense(units=nodes[i], kernel_initializer=weight_init, activation=hidden_activation, name=hidden_name))
15
16         # Create Output Layers
17         model.add(layers.Dense(units=nodes[-1], kernel_initializer=weight_init, activation=output_activation, name="output"))
18
19         # Compile Neural Network
20         model.compile(optimizer=opt_name, loss=loss_name, metrics=metric_list)
21
22     # Return built model
23     return model
```

1 預設參數（以「多選一」為標準）

2 建造輸入層

3 用「迴圈」建造隱藏層

4 建造輸出層

5 編譯&回傳

模型建置：HappyML 寫法

- 呼叫範例

指定各層節點數目

輸入層　　隱藏層　輸入層

```
1  import HappyML.neural_networks as nn
2
3  model = nn.create_seq_model(nodes=[X_train.shape[1], 45, 23, 1],
4                              output_activation="sigmoid", loss_name="binary_crossentropy")
```

「二選一」激活函數

「二選一」損失函數

隨堂練習：模型建置

- 請撰寫好前一頁投影片的程式碼。

- 用「草稿儲存格」輸入下列指令，看看建構出來的神經網路模型，是否跟你指定的一樣？
  - model.summary()
  - model.inputs

- 課後實驗：找找看下列超參數有沒有更好的選擇？
  - kernel_initializer= “glorot_normal” （有收斂更快的選擇嗎？）
  - optimizer= “adam” （有收斂更快的選擇嗎？）

模型校正：問題說明

- 原始碼

到底要訓練幾期，
才最適合這類問題？

```
1  # Fit
2  epochs_metrics = model.fit(x=X_train, y=Y_train, validation_split=0.2, batch_size=10, epochs=100)
```

```
Epoch 1/100
488/488 [==============================] - 1s 2ms/step - loss: 0.1227 - acc: 0.9581 - val_loss: 0.0095 - val_acc: 1.0000
Epoch 2/100
488/488 [==============================] - 1s 2ms/step - loss: 0.0047 - acc: 0.9994 - val_loss: 0.0018 - val_acc: 1.0000
Epoch 3/100
488/488 [==============================] - 1s 2ms/step - loss: 0.0011 - acc: 1.0000 - val_loss: 6.0447e-04 - val_acc: 1.0000
Epoch 4/100
488/488 [==============================] - 1s 2ms/step - loss: 4.1227e-04 - acc: 1.0000 - val_loss: 2.9228e-04 - val_acc: 1.0000
Epoch 5/100
488/488 [==============================] - 1s 2ms/step - loss: 2.0861e-04 - acc: 1.0000 - val_loss: 1.7977e-04 - val_acc: 1.0000
Epoch 6/100
....
Epoch 99/100
488/488 [==============================] - 1s 2ms/step - loss: 5.6015e-10 - acc: 1.0000 - val_loss: 7.6610e-10 - val_acc: 1.0000
Epoch 100/100
488/488 [==============================] - 1s 2ms/step - loss: 5.5456e-10 - acc: 1.0000 - val_loss: 7.3535e-10 - val_acc: 1.0000
```

很早就收斂了！

模型校正：TensorBoard 工具簡介

- 使用 TensorBoard，了解訓練出來的神經網路參數！

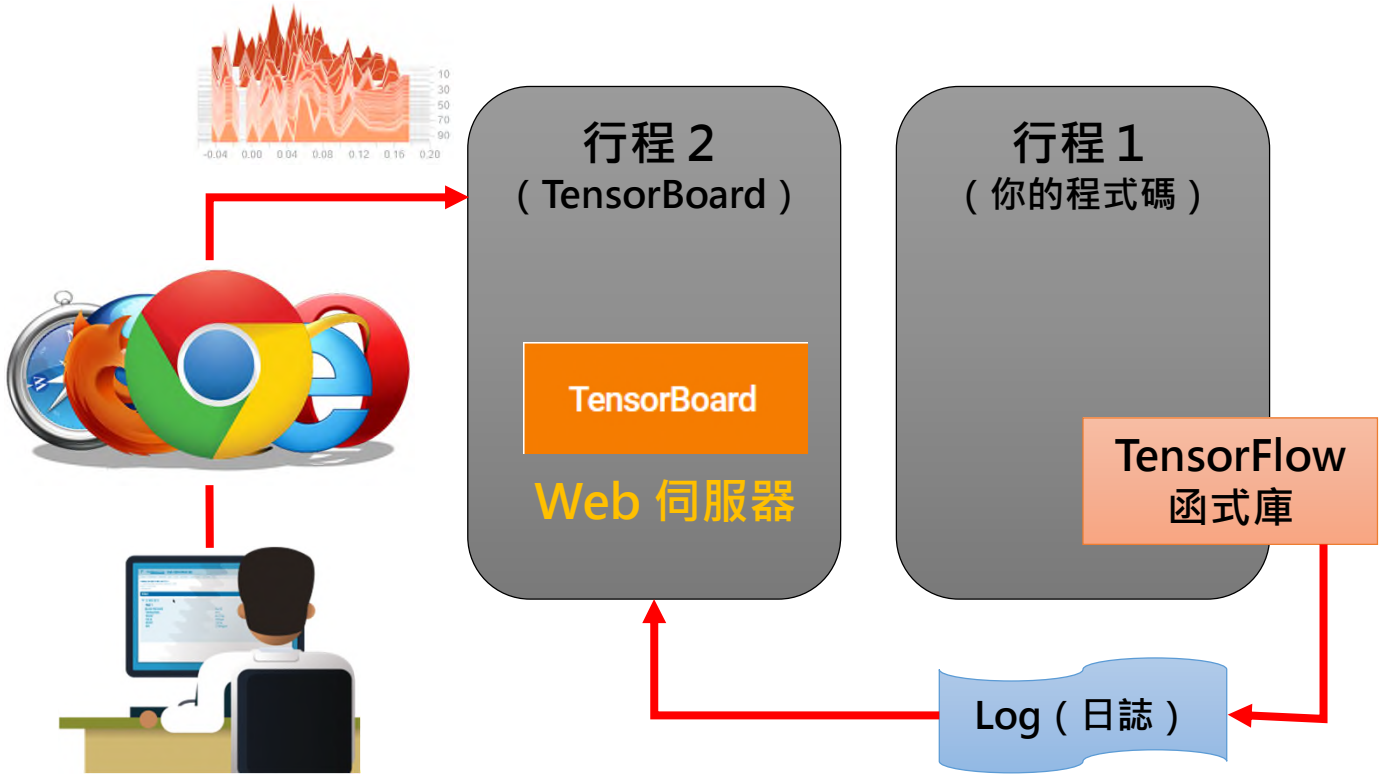
模型校正：TensorBoard 工具簡介

- 何謂「TensorBoard」？

TensorBoard

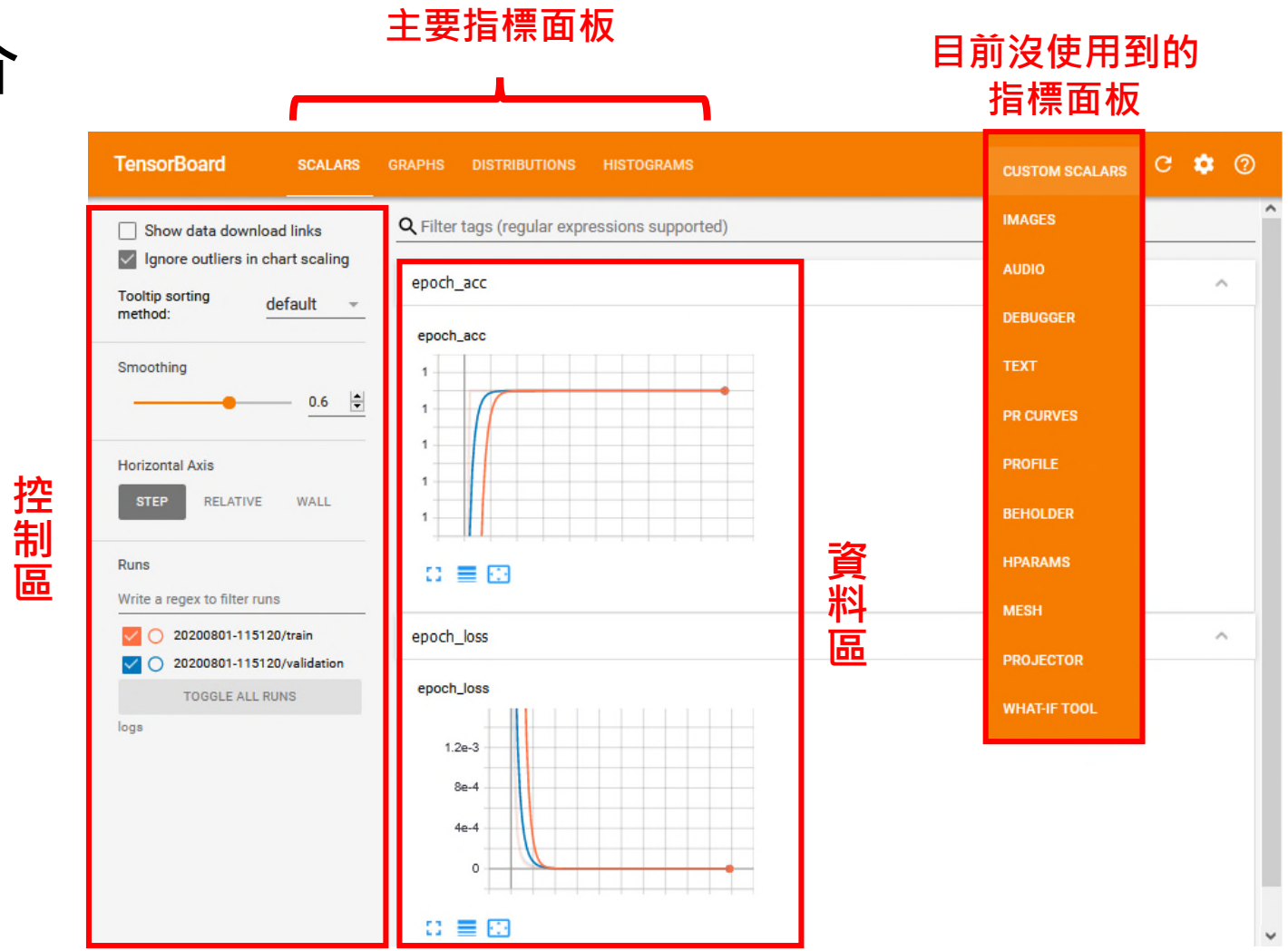
將 TensorFlow 執行期日誌
視覺化的工具

- 「TensorBoard」如何運作？



模型校正：TensorBoard 工具簡介

- 使用者介面簡介



模型校正：TensorBoard 工具簡介

- 常見的「**指標面板**」

特定純量值

神經網路結構

各期權重分佈狀況
（折線圖形式）

各期權重分佈狀況
（直方圖形式）

- 各期圖片輸入值
- 各期聲音輸入值
- 除錯器
- 各期文字輸入值
- 精度-廣度曲線
  （不同模型比較用）
- 效能瓶頸分析器
- 以影片呈現訓練成果
- 超參數組合比較
- 3D 模型視覺化
- 高維資料投射器
- 不同模型比較工具

更詳細說明：請參考 TensorBoard 官方指南

模型校正：TensorBoard 程式碼

- 為我們的程式碼，加入「TensorBoard」

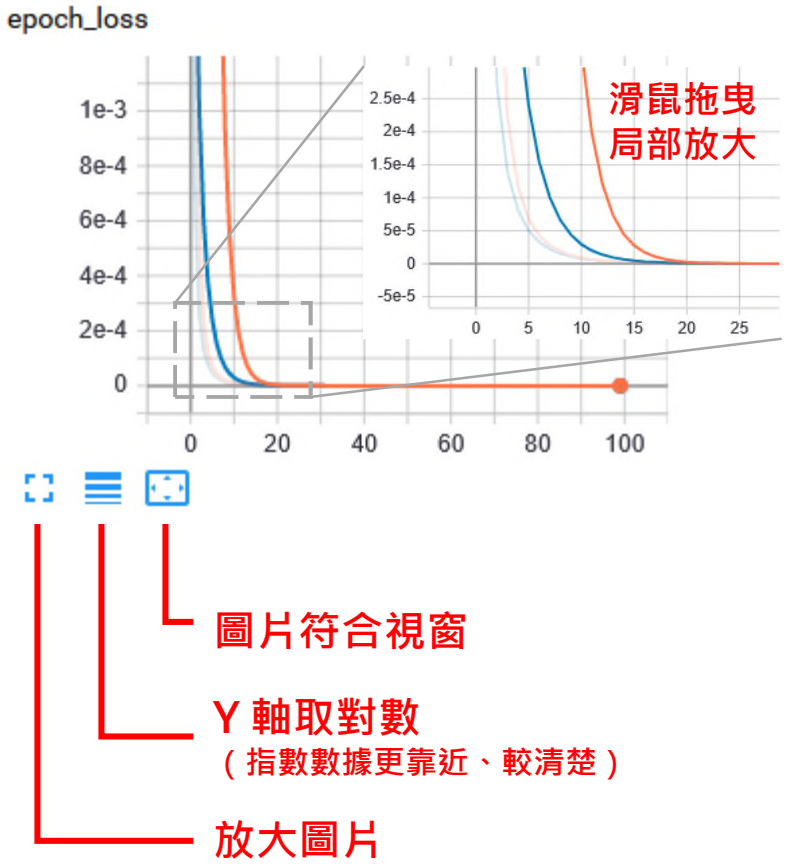
```
1  # Load the TensorBoard notebook extension
2  %load_ext tensorboard
3
4  # Create TensorBoard log directory
5  import os
6  from datetime import datetime
7  from tensorflow.keras.callbacks import TensorBoard
8                          資料放在類似：logs/20200801-175346 這樣的資料夾下
9  logdir = os.path.join("logs", datetime.now().strftime("%Y%m%d-%H%M%S"))
10 tensorboard_callback = TensorBoard(logdir, histogram_freq=1)
11                                                  多少 epochs 要
12 # Start the TensorBoard                          更新一次直方圖數據
13 %tensorboard --logdir logs
14
15 # Fit
16 epochs_metrics = model.fit(x=X_train, y=Y_train, validation_split=0.2,
17     batch_size=10, epochs=100, callbacks=[tensorboard_callback])
```

logs 資料夾下的資料全抓

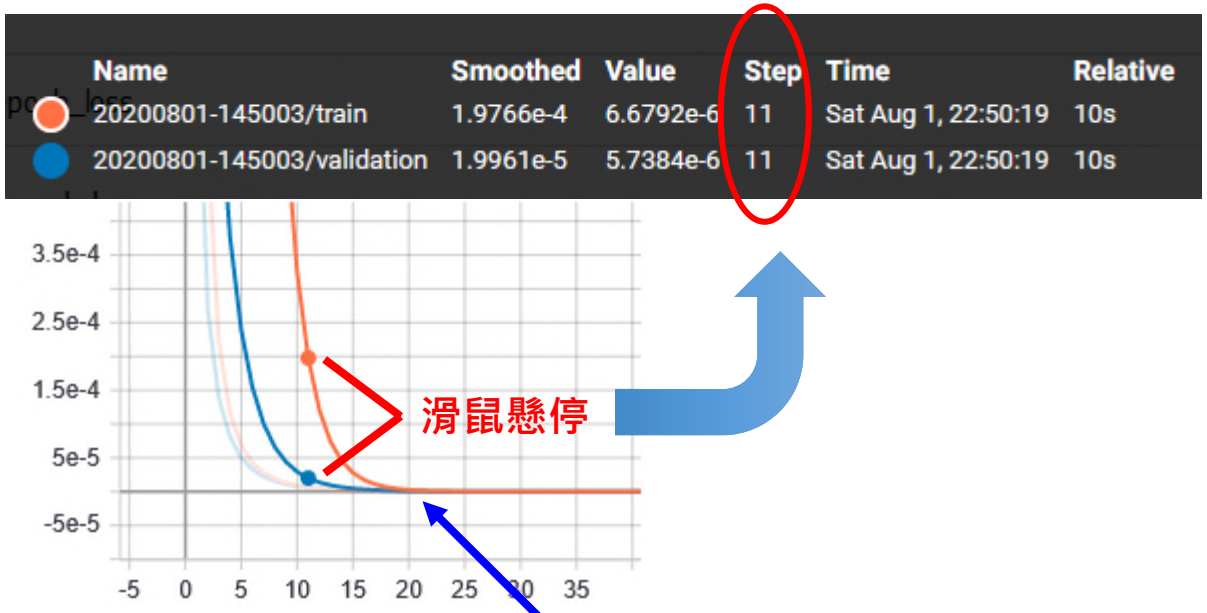
1. 用 Colab 特殊指令 %load_ext 載入 TensorBoard 函式庫外掛
2. 指定這次執行出來的資料，要丟到哪個資料夾。並產生 TensorBoard 物件。
3. 啟動 TensorBoard 行程，內建 Web 伺服器開始運轉
4. 將 TensorBoard 機制「插」到每一個訓練週期裡

模型校正：TensorBoard 校正結果

- 觀看「**執行結果**」



第 11 期的訓練狀況

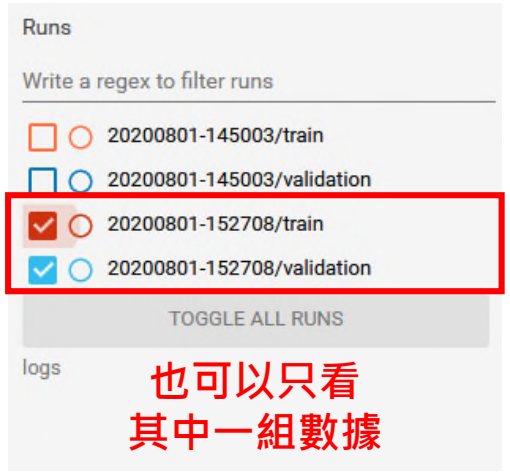


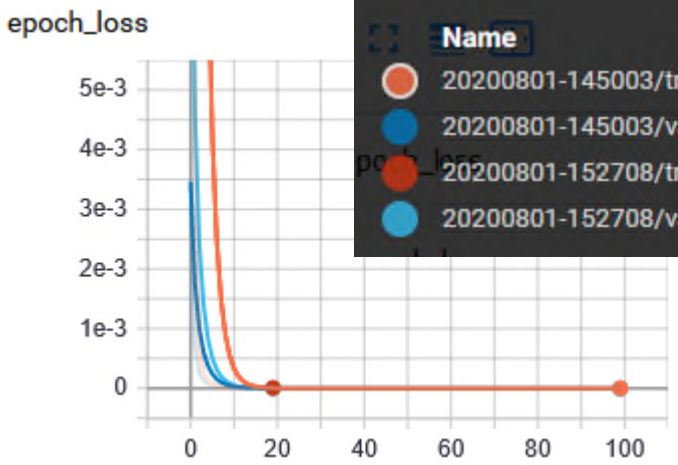
結論：

- 由圖可知，訓練到第 20 期就夠了！

隨堂練習：模型校正

- 請先寫好 TensorBoard 引入、啟動，以及模型訓練的程式碼。
- 執行看看，是否能從 TensorBoard 看到模型所顯示的數據？
- 將 .fit() 的 epochs 參數，從 100 修正成 20，重新執行一次看看。
- 現在你有兩組數據（每組另含訓練集、驗證集數據）。用滑鼠懸停，找出兩組數據的執行時間，比較看看。你節省了多少時間？





Name	Smoothed	Value	Step	Time	Relative
20200801-145003/train	6.0991e-1	6.1314e-1	99	Sat Aug 1, 22:51:47	1m 38s
20200801-145003/validation	7.1393e-1	7.0754e-1	99	Sat Aug 1, 22:51:47	1m 38s
20200801-152708/train	4.8797e-6	8.8351e-7	19	Sat Aug 1, 23:27:30	19s
20200801-152708/validation	1.9565e-6	8.411e-7	19	Sat Aug 1, 23:27:30	19s

模型訓練（利用校正過後的數字）

- 程式碼

```
# Model Training

epochs_metrics = model.fit(x=X_train, y=Y_train, validation_split=0.2, batch_size=10, epochs=20)
```

```
Epoch 1/20
488/488 [==============================] - 2s 4ms/step - loss: 6.4426e-10 - acc: 1.0000 - val_loss: 2.2082e-09 - val_acc: 1.0000
Epoch 2/20
488/488 [==============================] - 2s 4ms/step - loss: 6.4838e-10 - acc: 1.0000 - val_loss: 2.2416e-09 - val_acc: 1.0000
Epoch 3/20
488/488 [==============================] - 2s 4ms/step - loss: 6.5217e-10 - acc: 1.0000 - val_loss: 2.2738e-09 - val_acc: 1.0000
...
Epoch 19/20
488/488 [==============================] - 2s 4ms/step - loss: 6.4779e-10 - acc: 1.0000 - val_loss: 2.5189e-09 - val_acc: 1.0000
Epoch 20/20
488/488 [==============================] - 2s 4ms/step - loss: 6.2957e-10 - acc: 1.0000 - val_loss: 2.4386e-09 - val_acc: 1.0000
```

預測答案

- 程式碼

```
# Predict
import numpy as np
import pandas as pd

Y_pred = np.rint(model.predict(x=X_test))
Y_pred = pd.DataFrame(Y_pred, index=Y_test.index, columns=Y_test.columns)

# Show the Predict Result
df = pd.concat([Y_test, Y_pred], axis=1)
print(df)
```

	class	class
2076	0	0
1405	1	1
5293	1	1
1687	0	0
3303	0	0
...	...	...
2387	0	0
5226	1	1
7771	1	1
4229	1	1
2758	0	0

[2031 rows x 2 columns]

真實值
Y_test

預測值
Y_pred

模型評估

- 程式碼

```
1  test_loss, test_acc = model.evaluate(X_test, Y_test)
2  print("Loss of Testing Set:", test_loss)
3  print("Accuracy of Testing Set:", test_acc)
```

```
64/64 [==============================] - 0s 999us/step - loss: 5.3373e-07 - acc: 1.0000
Loss of Testing Set: 5.3373179298432666e-07
Accuracy of Testing Set: 1.0
```

隨堂練習：預測&評估

- 請將前三頁的程式碼寫好，並執行看看。
- 您的模型效能如何？夠好嗎？

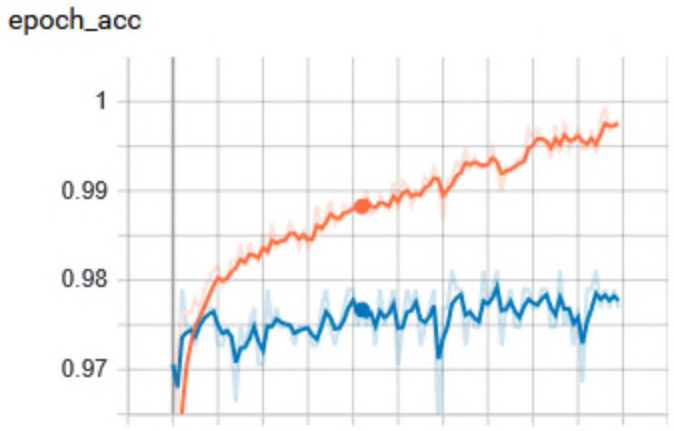
課後作業：辨識「男聲」vs.「女聲」

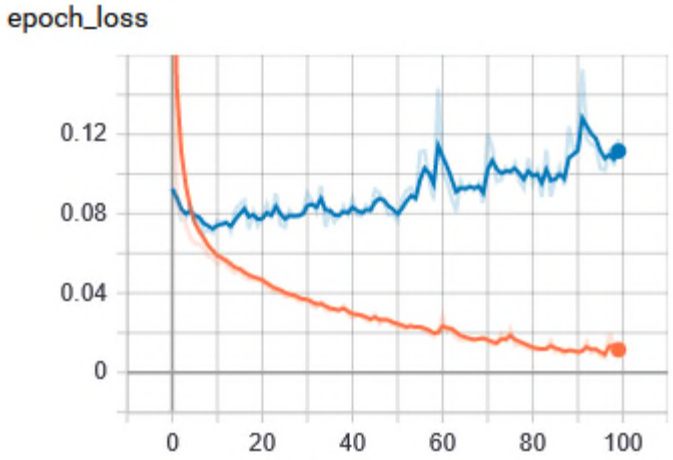
- **資料集介紹：有一聲音分析資料集，包含下列欄位：**
  - 前 20 欄（[0]~[19]），是各種音質分析參數：
    - meanfreq：平均音頻
    - sd：最低音~最高音標準差
    - median：音頻中位數
    - Q25, IQR, Q75：音頻的第一、第二、第三、四分位數。
    - ……（以下略）
  - 第 21 欄（[20]），是判別結果（male=男性、female=女性）

- **要求**
  - 請用**神經網路**，建造一個**分類器**模型，能辨別「**男聲**」或是「**女聲**」。
  - 用 **TensorBoard** 觀看您的模型，調整**超參數**。
  - 印出您的模型**效能**有多好。

課後作業：辨識「男聲」vs.「女聲」

- 提示
  - 如果「**訓練集**」與「**驗證集**」的「**損失值（Loss）**」、或「**確度（Accuracy）**」，發生「**分岔**」現象，代表已經 **Overfit** 了。
  - 請調整**訓練期數**（epochs），讓模型在 **Overfit** 之前停下來。



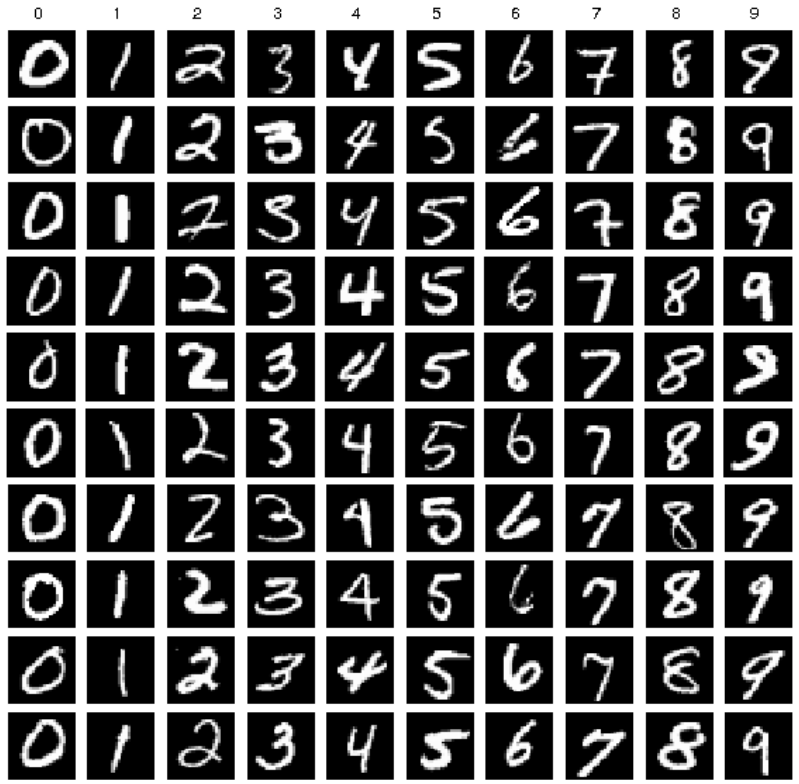


分類問題（多選一）
Multi-labels Classification

範例完整原始碼：
https://lurl.cc/aR5adn

問題描述：MNIST 手寫數字資料集

- MNIST = Modified National Institute of Standards and Technology



- 美國國家科技標準局
- 美國高中生手寫數字的圖片
- 60000 訓練集 + 10000 測試集
- 28 x 28 灰階圖片
- 多選一分類問題（10 選 1）

載入資料集

下載 HappyML 函式庫

```
# Install HappyML Library
import os

if not os.path.isdir("HappyML"):
  os.system("git clone https://github.com/cnchi/HappyML.git")
```

載入 MNIST 資料集

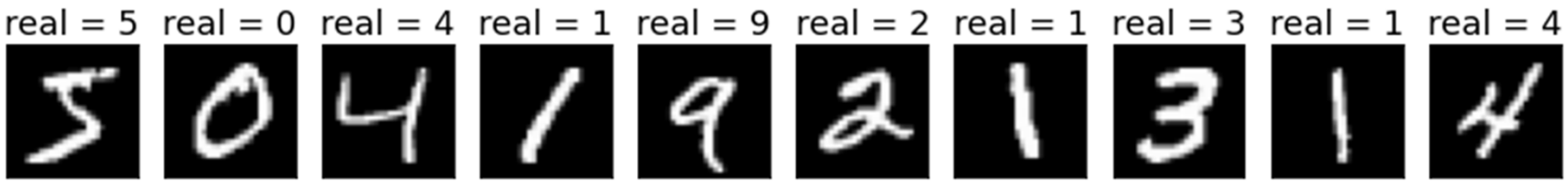
```
from tensorflow.keras.datasets import mnist

# Load Database
(X_train, Y_train), (X_test, Y_test) = mnist.load_data()
```

載入資料集

驗證資料集載入成功

```
1  import HappyML.model_drawer as md
2
3  md.show_first_n_images(x_ary=X_train, y_real=Y_train, first_n=10)
```

執行結果



載入資料集

- show_first_n_images() 程式碼解說

```python
def show_first_n_images(x_ary, y_real=[], y_pred=[], first_n=5, font_size=18, color_scheme="gray"):
    # Get Current Figure (GCF) & Set Height 15 inches, Width 4 inches
    plt.gcf().set_size_inches(15, 4)

    # Iterate the first N images
    for i in range(first_n):
        # each row has first_n sub-images
        ax = plt.subplot(1, first_n, i+1)

        # "gray": black background, "binary": white background
        ax.imshow(x_ary[i], cmap=color_scheme)

        # set sub-image title
        if y_pred == []:
            img_title = "real = {}".format(y_real[i])
        else:
            img_title = "real = {}\npred = {}".format(y_real[i], y_pred[i])
        ax.set_title(img_title, fontsize=font_size)

        # Make X-axis, Y-axis without ticks
        ax.set_xticks([])
        ax.set_yticks([])

    # Show all images
    plt.show()
```

1. gcf(): get current figure，取得目前繪圖區域
   set_size_inches()：設定繪圖區的高 x 寬（英吋）
2. 設定繪圖區域有幾張「子圖」
3. 繪製「子圖」，並設定圖片「色調」。
   “gray”：黑底白字、” binary”：白底黑字
4. 若沒有 Y_pred：只秀圖片的「真實答案」
   若有 Y_pred：包含「真實」與「預測」答案
5. 去掉 X 與 Y 軸的刻度

隨堂練習：載入資料集

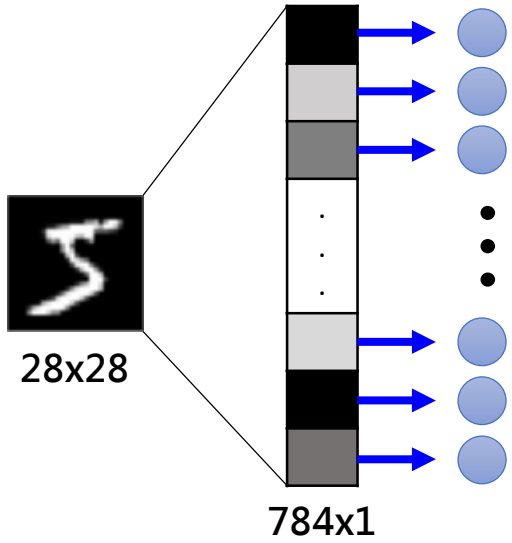
- **請依序撰寫下列原始碼**
  - 「下載 HappyML 函式庫」的原始碼
  - 「載入 MNIST 資料集」的原始碼
  - 「驗證資料集載入成功」的原始碼

- **執行看看，看是否能看到前幾張已載入的圖片？**

重新整理自變數 X & 應變數 Y

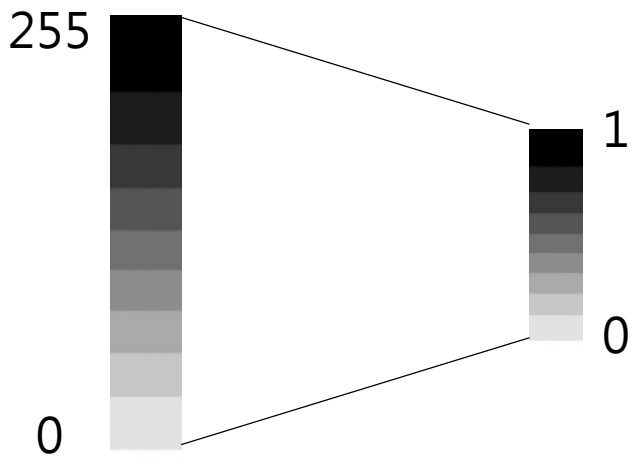
- **自變數 X**：展平圖片

```
1  # Re-shape X from 28x28 to 784x1 (Flatten)
2  X_train = X_train.reshape((60000, 28*28))
3  X_test = X_test.reshape((10000, 28*28))
```



- **自變數 X**：縮放灰階範圍

```
1  # Change Gray-scale from 0~255 to 0~1
2  X_train = X_train.astype("float32") / 255
3  X_test = X_test.astype("float32") / 255
```



重新整理自變數 X & 應變數 Y

- **應變數 Y**：使用 One-Hot Encoding

```
1  # One-Hot Encoding for Y
2  from keras.utils import to_categorical
3
4  Y_train = to_categorical(Y_train)
5  Y_test = to_categorical(Y_test)
```

[0] → [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]

輸出層

	0	1	2	3	4	5	6	7	8	9
0 →	1	0	0	0	0	0	0	0	0	0
1 →	0	1	0	0	0	0	0	0	0	0
2 →	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
⋮										
	0	0	0	0	0	0	1	0	0	0
7 →	0	0	0	0	0	0	0	1	0	0
8 →	0	0	0	0	0	0	0	0	1	0
9 →	0	0	0	0	0	0	0	0	0	1

隨堂練習：整理自變數 X & 應變數 Y

- **請依序撰寫下列原始碼**
  - 「將自變數 X 展平」的原始碼
  - 「將自變數 X 灰階調整成 0～1」的原始碼
  - 「對應變數 Y 執行 One-Hot Encoding」的原始碼

- **執行看看，並且用下列幾個指令，檢查結果是否如您所預期？**
  - print(X_train)
  - print(Y_train)
  - print(X_test)
  - print(Y_test)

模型建立 & 校正

- 建立模型

指定各層節點數目

```
1  from HappyML.neural_networks import create_seq_model
2
3  # Model create
4  model = create_seq_model(nodes=[784, 38, 10])
```

模型建立 & 校正

- 建立、啟動 TensorBoard

```
# Load the TensorBoard notebook extension
%load_ext tensorboard

# Create TensorBoard log directory
import os
from datetime import datetime
from tensorflow.keras.callbacks import TensorBoard

logdir = os.path.join("logs", datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = TensorBoard(logdir, histogram_freq=1)

# Start the TensorBoard
%tensorboard --logdir logs
```

模型建立 & 校正

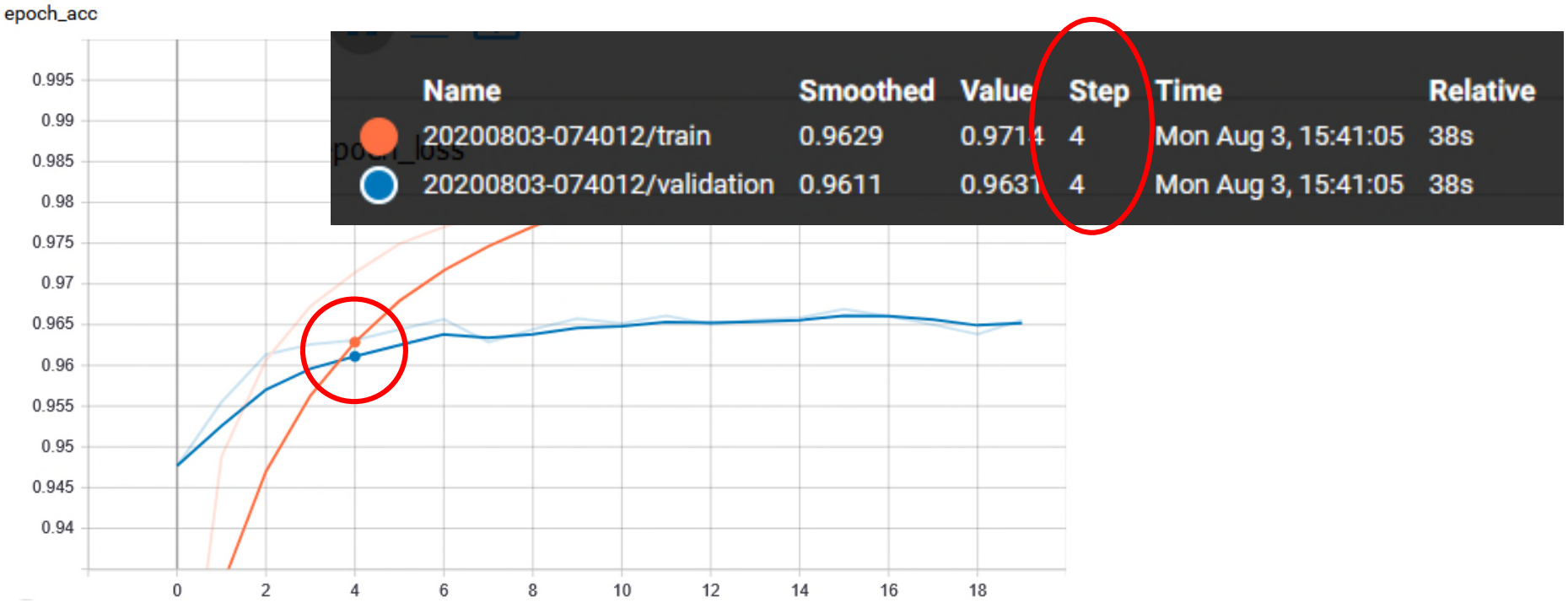
- 開始訓練模型

```
# Train the model
model.fit(x=X_train, y=Y_train, validation_split=0.2,
    batch_size=10, epochs=20, callbacks=[tensorboard_callback])
```

```
Epoch 1/20
4800/4800 [==============================] - 9s 2ms/step - loss: 0.3231 - acc: 0.9061 - val_loss: 0.1928 - val_acc: 0.9477
Epoch 2/20
4800/4800 [==============================] - 9s 2ms/step - loss: 0.1723 - acc: 0.9488 - val_loss: 0.1548 - val_acc: 0.9555
Epoch 3/20
4800/4800 [==============================] - 10s 2ms/step - loss: 0.1314 - acc: 0.9607 - val_loss: 0.1273 - val_acc: 0.9613
...
Epoch 19/20
4800/4800 [==============================] - 9s 2ms/step - loss: 0.0260 - acc: 0.9911 - val_loss: 0.1675 - val_acc: 0.9638
Epoch 20/20
4800/4800 [==============================] - 9s 2ms/step - loss: 0.0243 - acc: 0.9919 - val_loss: 0.1577 - val_acc: 0.9656
```

模型建立 & 校正

- 發現只要訓練約 5 期（0~4）就夠了



模型建立 & 校正

- 將期數（epochs）改為 5 期（避免「過擬合」）

```
1  # Train the model
2  model.fit(x=X_train, y=Y_train, validation_split=0.2,
3      batch_size=10, epochs=5, callbacks=[tensorboard_callback])
```

5 期就足夠了！

隨堂練習：模型建立 & 校正

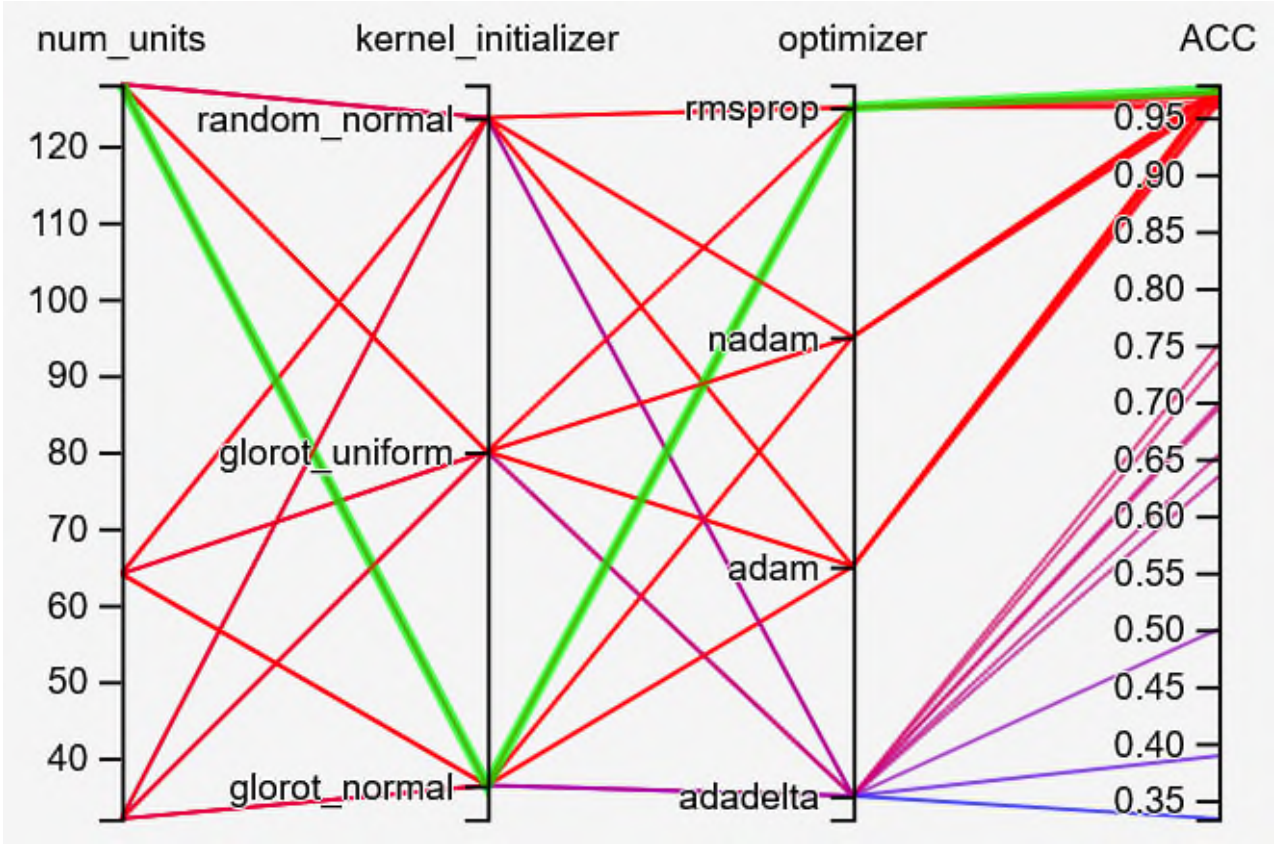
- 請依序撰寫下列原始碼
  - 「模型建立」的原始碼
  - 「建立、啟動 TensorBoard」的原始碼
  - 「訓練模型」的原始碼

- 執行看看，您覺得模型在多少期（epochs）訓練後，會開始「過擬合（Overfit）」？

- 修正您的期數，使您的模型不致於「過擬合」。

超參數調整

- 使用 TensorBoard 執行超參數調整



最佳組合

隱藏層節點數 = 128

x

權重初始化 = glorot_normal

x

優化器 = RMSProp

正確率 = 97.39%

超參數調整

- 程式碼（１）：製作「超參數（Hyper-Parameters）」組合

```
1  # Initializing Hyper-parameters
2  import tensorflow as tf
3  from tensorboard.plugins.hparams import api as hp
4
5  # Setup the HParams directory
6  hparams_dir = os.path.join(logdir, 'validation')
7
8  # Define Hyper-parameters
9  HP_NUM_UNITS = hp.HParam('num_units', hp.Discrete([32, 64, 128]))
10 HP_WEIGHT_INIT = hp.HParam('kernel_initializer',
11         hp.Discrete(["random_normal", "glorot_uniform", "glorot_normal"]))
12 HP_OPTIMIZER = hp.HParam('optimizer',
13         hp.Discrete(['adadelta', 'rmsprop', 'adam', 'nadam']))
14
15 HPARAMS = [HP_NUM_UNITS, HP_WEIGHT_INIT, HP_OPTIMIZER]
16
17 # Set the Standard of Evaluation
18 METRIC_ACC = "acc"
```

→ 路徑必須指到現在目錄之下的 validation 目錄

→製作「超參數」組合

→ 各組合優劣評估標準

超參數調整

- 程式碼（2）：設定 & 迭代

```
20  #Creating & configuring a log file writer
21  with tf.summary.create_file_writer(hparams_dir).as_default():
22      hp.hparams_config(
23          hparams=HPARAMS,
24          metrics=[hp.Metric(METRIC_ACC, display_name='ACC')]
25      )
26
27  # Count the total run
28  total_run = 1
29  for p in HPARAMS:
30    total_run *= len(p.domain.values)
31
32  curr_run = 1
33  for num_units in HP_NUM_UNITS.domain.values:
34      for weight_init in HP_WEIGHT_INIT.domain.values:
35          for optimizer in HP_OPTIMIZER.domain.values:
36              hparams = {
37                  HP_NUM_UNITS: num_units,
38                  HP_WEIGHT_INIT: weight_init,
39                  HP_OPTIMIZER: optimizer
40              }
```

1 of 36

curr_run　total_run

→把「超參數組合」+「評估標準」塞入系統

```
--- Starting trial: 1 of 36
{'num_units': 32, 'kernel_initializer': 'glorot_normal',
Epoch 1/5
4800/4800 [==============================] - 9s 2ms/step
Epoch 2/5
4800/4800 [==============================] - 8s 2ms/step
Epoch 3/5
4800/4800 [==============================] - 9s 2ms/step
```

→用迴圈挑出這次要評估的「超參數組合」

超參數調整

- 程式碼（3）：執行、評估、寫入日誌

```
--- Starting trial: 1 of 36
{'num_units': 32, 'kernel_initializer': 'glorot_normal'
Epoch 1/5
4800/4800 [==============================] - 9s 2ms/step
Epoch 2/5
```

```
42  # Show somethings on Standard Output for progress
    run_name = "{} of {}".format(curr_run, total_run)
    print('--- Starting trial: {}'.format(run_name))
    print({h.name: hparams[h] for h in hparams})
46
47  # Write information into log
48  with tf.summary.create_file_writer(hparams_dir + "/" + run_name).as_default():
        hp.hparams(hparams)
50
51      # Create & Train the model
52      theNodes = [784, hparams[HP_NUM_UNITS], 10]
53      model = create_seq_model(nodes=theNodes, weight_init=hparams[HP_WEIGHT_INIT],
                opt_name=hparams[HP_OPTIMIZER])
55      model.fit(x=X_train, y=Y_train, validation_split=0.2, batch_size=10, epochs=5,
56              callbacks=[tensorboard_callback, hp.KerasCallback(hparams_dir, hparams)])
57
58      # Evaluate the Accuracy & Write to log
59      test_loss, test_acc = model.evaluate(X_test, Y_test)
60      tf.summary.scalar(METRIC_ACC, test_acc, step=10)
61  curr_run += 1
```

開啟寫入器 & 註冊這次的超參數 →

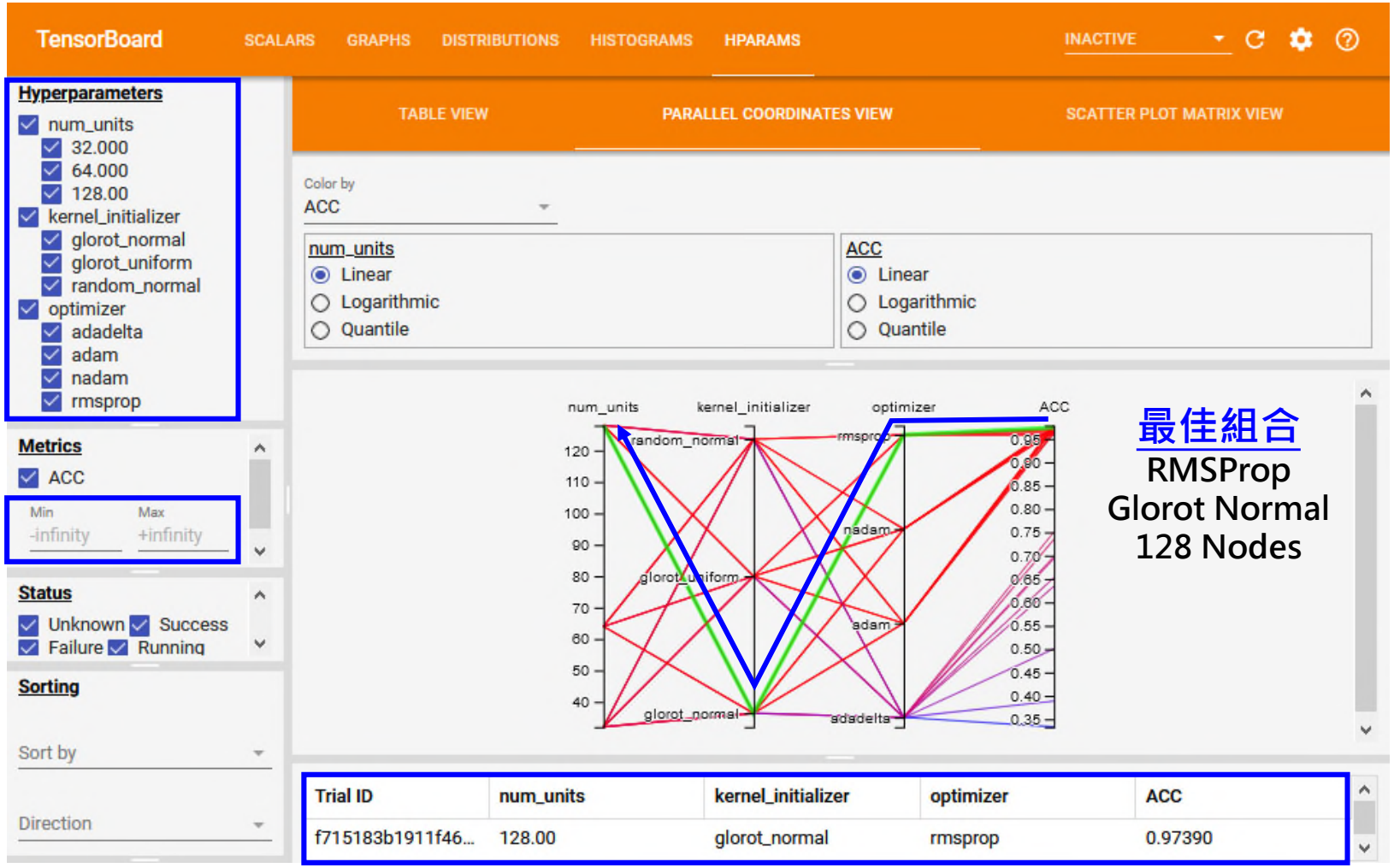
模型建造&訓練→

將這次的評估結果寫入→

超參數調整執行結果

顯示/隱藏
特定欄位

設定 ACC
上下限



最佳組合
RMSProp
Glorot Normal
128 Nodes

Trial ID	num_units	kernel_initializer	optimizer	ACC
f715183b1911f46...	128.00	glorot_normal	rmsprop	0.97390

選中那條線
的詳細數值

隨堂練習：調整超參數

- 請撰寫超參數調整的原始碼
- 執行看看，您覺得什麼組合，最適合解決現在的問題？

重新訓練模型、評估

- 重新訓練模型（使用調整出來的超參數）

```
## Model create
model = create_seq_model(nodes=[784, 128, 10], weight_init="glorot_normal", opt_name="rmsprop")
model.fit(x=X_train, y=Y_train, validation_split=0.2, batch_size=10, epochs=5)
```

- 評估模型效能

```
# Evaluate the Model
test_loss, test_acc = model.evaluate(X_test, Y_test)
print("Accuracy of Testing Set:", test_acc)
```

```
313/313 [==============================] - 0s 1ms/step - loss: 0.1256 - acc: 0.9718
Accuracy of Testing Set: 0.9718000292778015
```

RMSProp x Glorot Normal
x 128 Hidden Nodes x 5 Epochs

目前為止，最快速、精準、
且無過擬合的超參數組合！

隨堂練習：重新訓練模型、評估

- 請撰寫模型訓練、評估的原始碼
- 執行看看，您的模型效能目前有多好？
- 您是否已經知道如何取得一個神經網路的最佳超參數組合了？

預測結果

- 預測答案的程式碼

```
1  import numpy as np
2
3  # Predict the answer
4  Y_pred = np.argmax(model.predict(X_test), axis=1)
```

	0	1	2	3	4	5	6	7	8	9
axis=1 →	0.0	0.2	0.1	0.0	0.0	0.0	0.0	0.5	0.0	0.2

順著橫軸

argmax()

找到該軸最大數字，傳回索引值（=7）

0	1	2	3	4	5	6	7	8	9
0.0	0.2	0.1	0.0	0.0	0.0	0.0	0.5	0.0	0.2
0.0	0.0	0.8	**0.1**	0.0	0.0	0.0	0.1	0.0	0.0
0.0	0.7	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.0

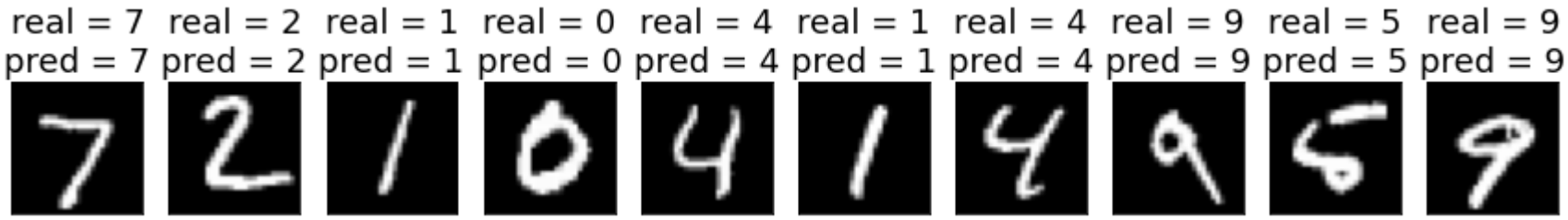
Y
7
2
1

預測結果

- 秀出預測結果前 10 筆

重新載回 28x28 灰階的照片

```
1  # Reload the pictures & show the answer vs. prediction
2  (x_train, y_train), (x_test, y_test) = mnist.load_data()
3  md.show_first_n_images(x_ary=x_test, y_real=y_test, y_pred=Y_pred, first_n=10)
```



前十筆預測皆正確！
太棒了！

隨堂練習：重新訓練模型、評估

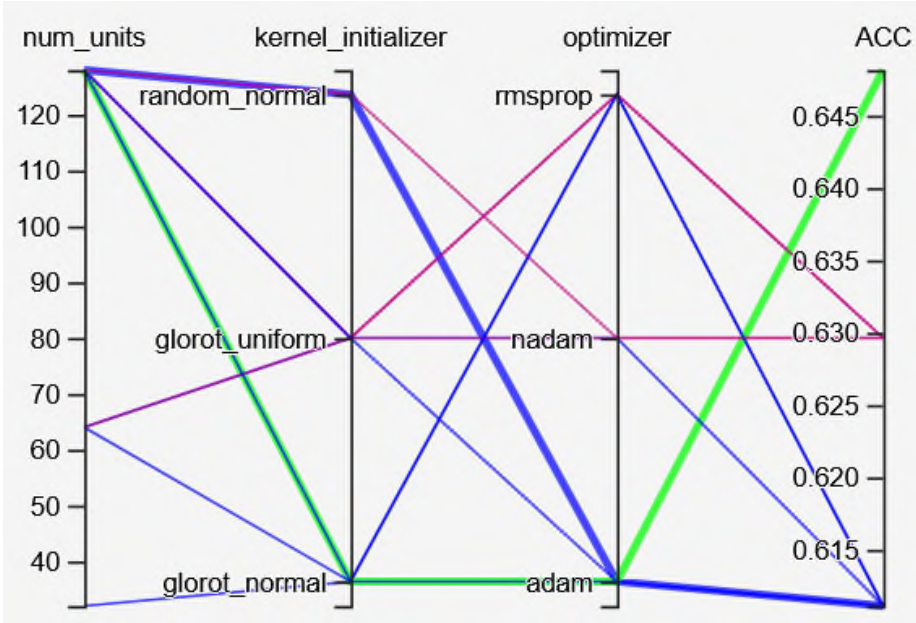
- 請撰寫預測答案、秀前十筆結果的原始碼
- 執行看看，感受一下，您模型的預測結果如何？
- 恭喜！您已經學會如何做出「多選一」的分類器，以及用 TensorBoard 選擇最佳超參數組合了。

課後作業：幫玻璃劃分等級

- **資料集介紹：玻璃會根據下列九個屬性，劃分成 1~7 等級**
  - RI：折射率（Refractive Index）
  - Na：鈉含量
  - Mg：鎂含量
  - Al：鋁含量
  - Si：矽含量
  - K：鉀含量
  - Ca：鈣含量
  - Ba：鋇含量
  - Fe：鐵含量
  - Type：等級，一共有 1~7 等
- **程式要求**
  - 請用**神經網路**，撰寫一個**「多選一」**的分類器，根據**屬性**自動幫玻璃**分級**。
  - 請記得去除顯著性不高的特徵（如：使用 HappyML.preprocessor.KBestSelector）。是否**特徵縮放**，可自行決定。
  - 請用 TensorBoard，調整出盡量好一點的**超參數組合**。
    （正確率高、訓練時間短、沒有過擬合）
  - 用 .evaluate() 計算您模型的**正確率**有多少？

課後作業：幫玻璃劃分等級

- 參考輸出




```
2/2 [==============================] - 0s 2ms/step - loss: 1.1141 - acc: 0.6481
Accuracy of Testing Set: 0.6481481194496155
```

本章總結

- **迴歸問題常用參數**
  - 輸出層激活函數："linear"
  - 損失函數："mse"
  - 效能指標："mse"
- **分類（二選一）問題常用參數**
  - 輸出層激活函數："sigmoid"
  - 損失函數："binary_crossentropy"
  - 效能指標："accuracy"
- **分類（多選一）問題常用參數**
  - 輸出層激活函數："softmax"
  - 損失函數："categorical_crossentropy"
  - 效能指標："accuracy"
- **TensorBoard 的用法**
  - 以「**訓練集**」與「**驗證集**」的**效能指標**（loss, acc），找出最佳**訓練期數**（防止**過擬合**）。
  - 用 **HParam** 找出最佳的**超參數組合**。