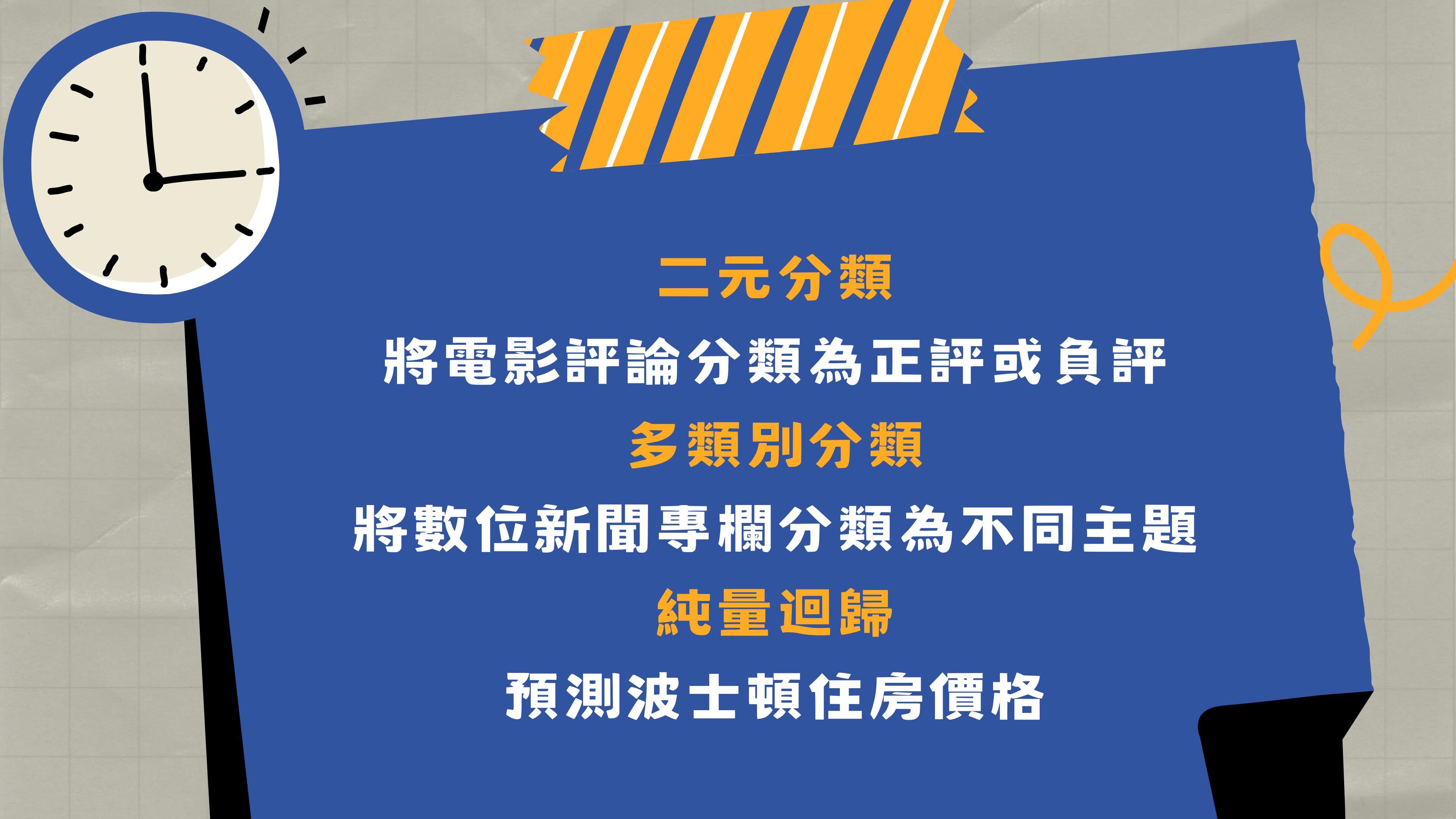


使用神經網路分類與回歸

第四章



二元分類

將電影評論分類為正評或負評

多類別分類

將數位新聞專欄分類為不同主題

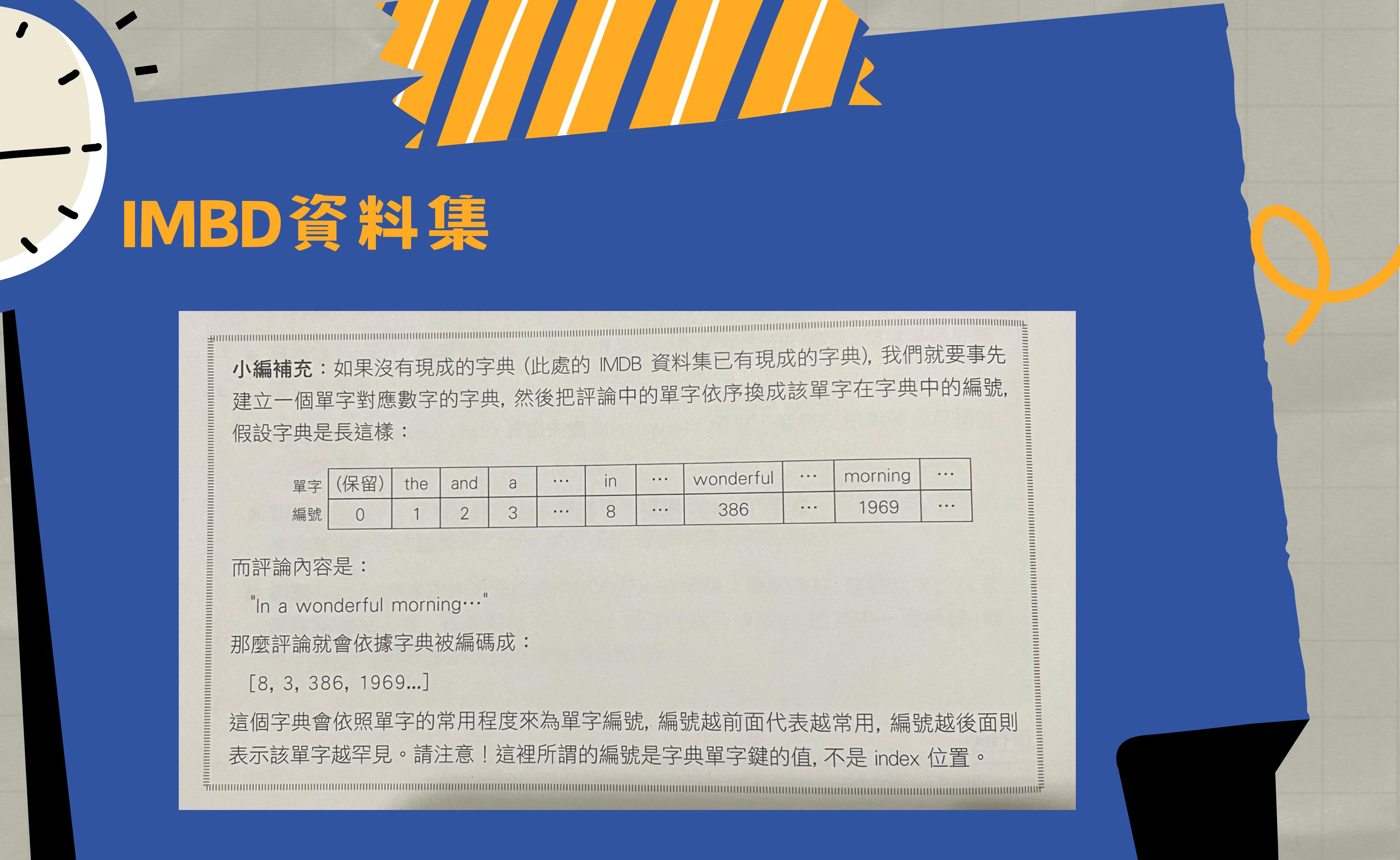
純量迴歸

預測波士頓住房價格



常見詞彙

- 樣本 / 輸入
- 預測值 / 輸出
- 目標值
- 預測誤差 / 損失值
- 類別
- 標籤
- 真實值 / 標註值
- 二元分類
- 多類別分類
- 多標籤分類
- 純量迴歸
- 向量迴歸
- 批次



IMBD資料集

小編補充：如果沒有現成的字典（此處的 IMDB 資料集已有現成的字典），我們就要事先建立一個單字對應數字的字典，然後把評論中的單字依序換成該單字在字典中的編號，假設字典是長這樣：

單字	(保留)	the	and	a	…	in	…	wonderful	…	morning	…
編號	0	1	2	3	…	8	…	386	…	1969	…

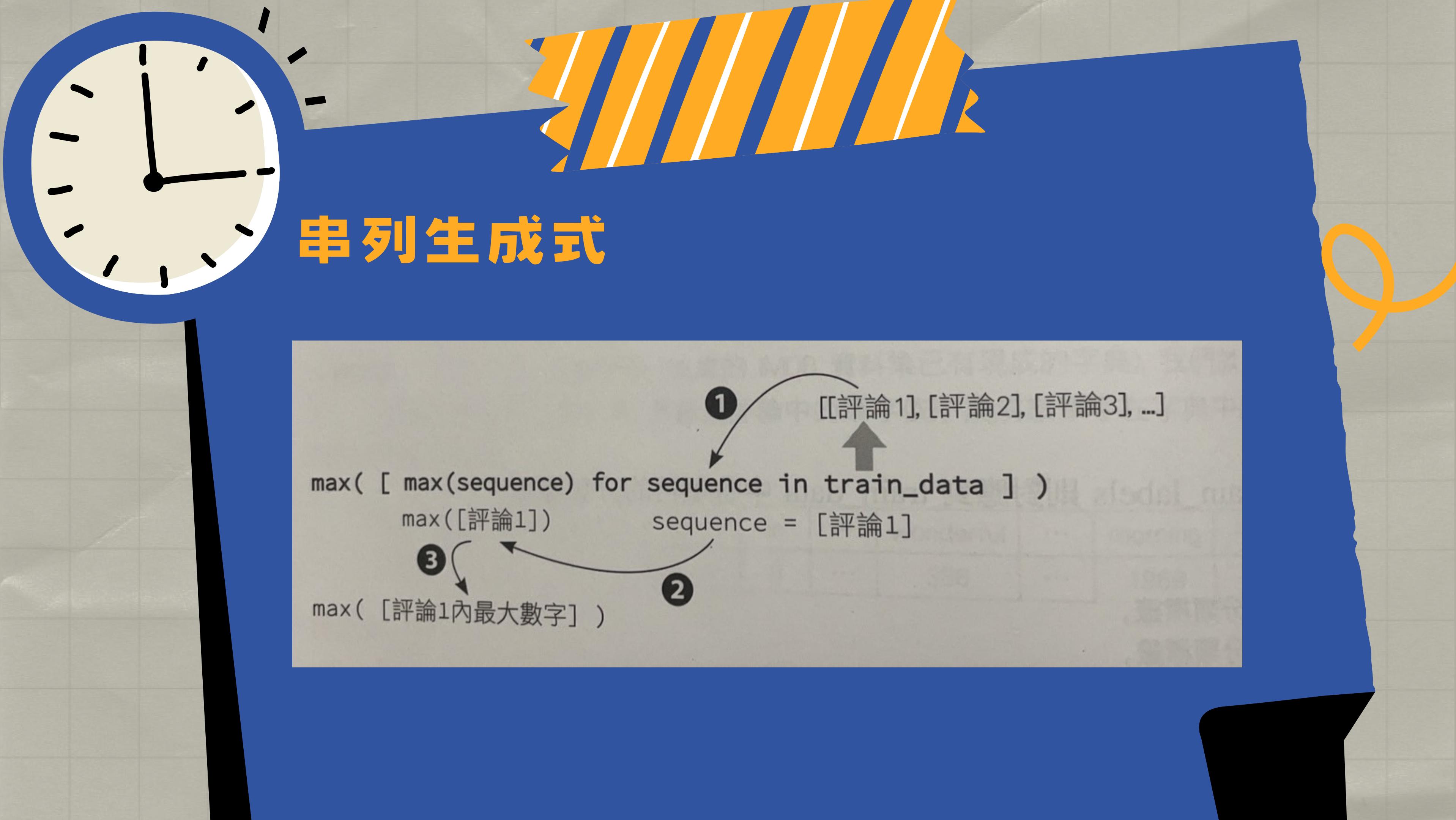
而評論內容是：

"In a wonderful morning..."

那麼評論就會依據字典被編碼成：

[8, 3, 386, 1969...]

這個字典會依照單字的常用程度來為單字編號，編號越前面代表越常用，編號越後面則表示該單字越罕見。請注意！這裡所謂的編號是字典單字鍵的值，不是 index 位置。



串列生成式

```
max( [ max(sequence) for sequence in train_data ] )  
      max([評論1])  
      sequence = [評論1]  
      1  
      2  
      3  
max( [評論1內最大數字] )
```

將數字還原成英文單字

```
dict( [ (value, key) for (key, value) in word_index.items() ] )  
      (5340, 'marshall') ('marshall', 5340)  
      ③ ②  
dict( [(5340, 'marshall'), (9095, 'honeymoon'), (3231, 'shoots'), ...] )  
      ↓ ④  
{  
  ...  
  5340: 'marshall',  
  9095: 'honeymoon',  
  3231: 'shoots',  
  ...  
}
```

The diagram illustrates the process of converting a dictionary where keys are integers back into a dictionary where keys are strings. It shows four steps:

- Original dictionary with integer keys:

```
dict( [ (value, key) for (key, value) in word_index.items() ] )  
      (5340, 'marshall') ('marshall', 5340)
```

- Keys are converted to strings:

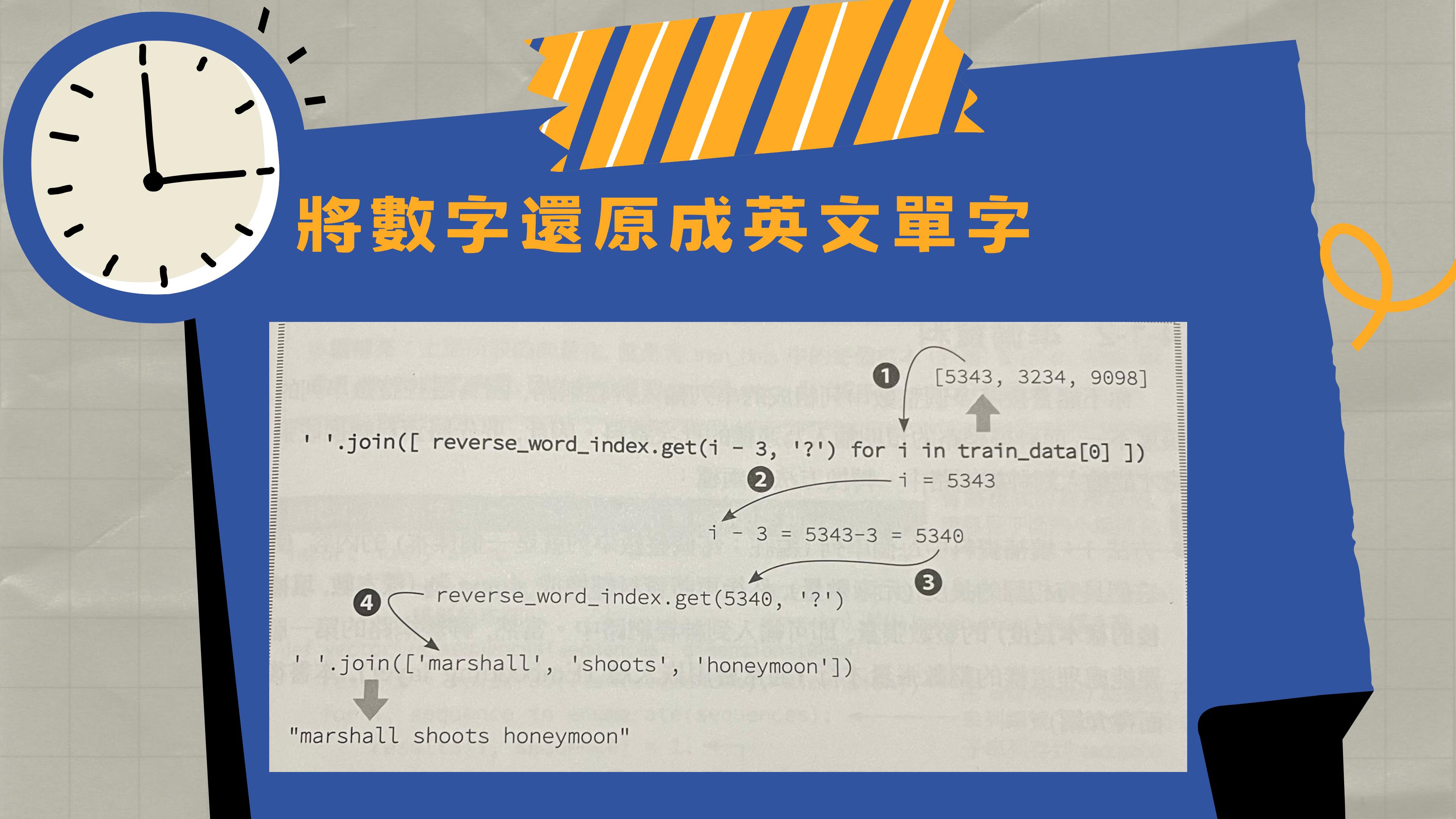
```
dict( [(5340, 'marshall'), (9095, 'honeymoon'), (3231, 'shoots'), ...] )
```

- Integer value is converted to its string representation:

```
5340: 'marshall',
```

- Final dictionary with both keys and values as strings:

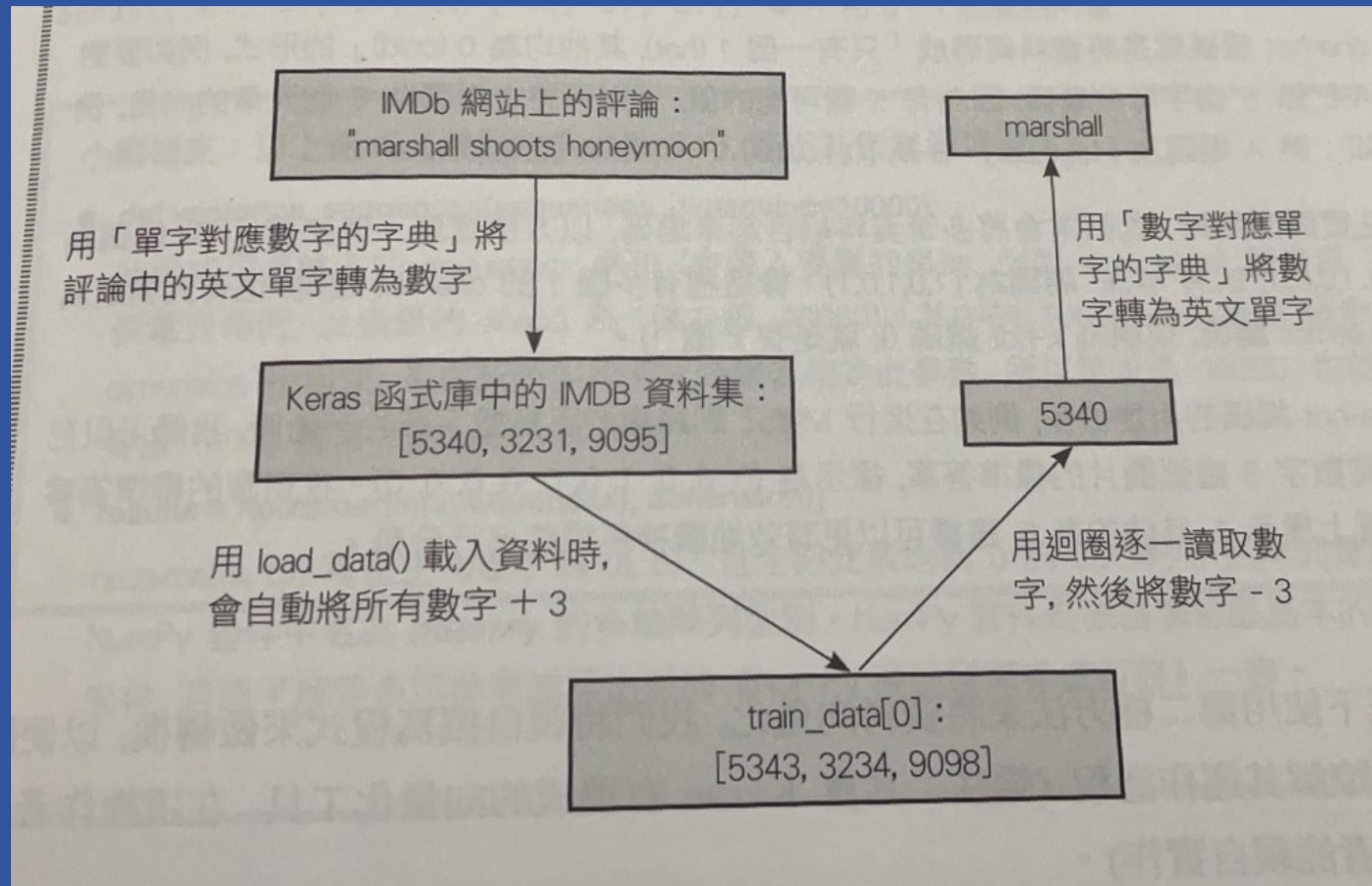
```
{  
  ...  
  5340: 'marshall',  
  9095: 'honeymoon',  
  3231: 'shoots',  
  ...  
}
```



將數字還原成英文單字

```
' '.join([ reverse_word_index.get(i - 3, '?') for i in train_data[0] ])  
  
① [5343, 3234, 9098]  
    ↑  
    ↓ i = 5343  
    ↓ i - 3 = 5343-3 = 5340  
    ↓ ③  
    ↓ ④ reverse_word_index.get(5340, '?')  
  
' '.join(['marshall', 'shoots', 'honeymoon'])  
  
↓  
"marshall shoots honeymoon"
```

將數字還原成英文單字





準備資料-兩個方法

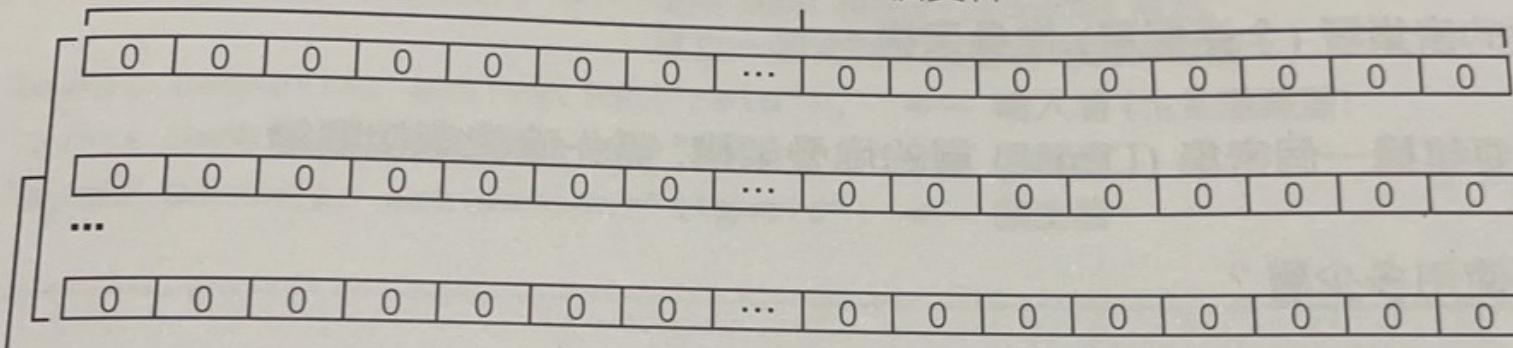
- 填補資料中每個串列
- 對資料中每個串列做multi-hot

RESULTS[i, J] = 1.

- `results[i, sequence] = 1.`

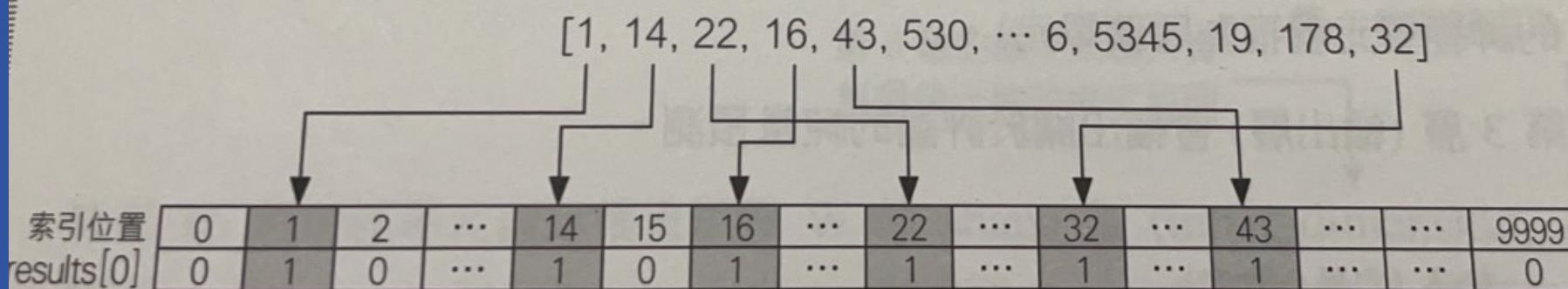
`results` 是先前所建立, 值均為 0、shape 為 (樣本數, 10000) 的 ndarray 陣列：

每一筆有 10000 個資料

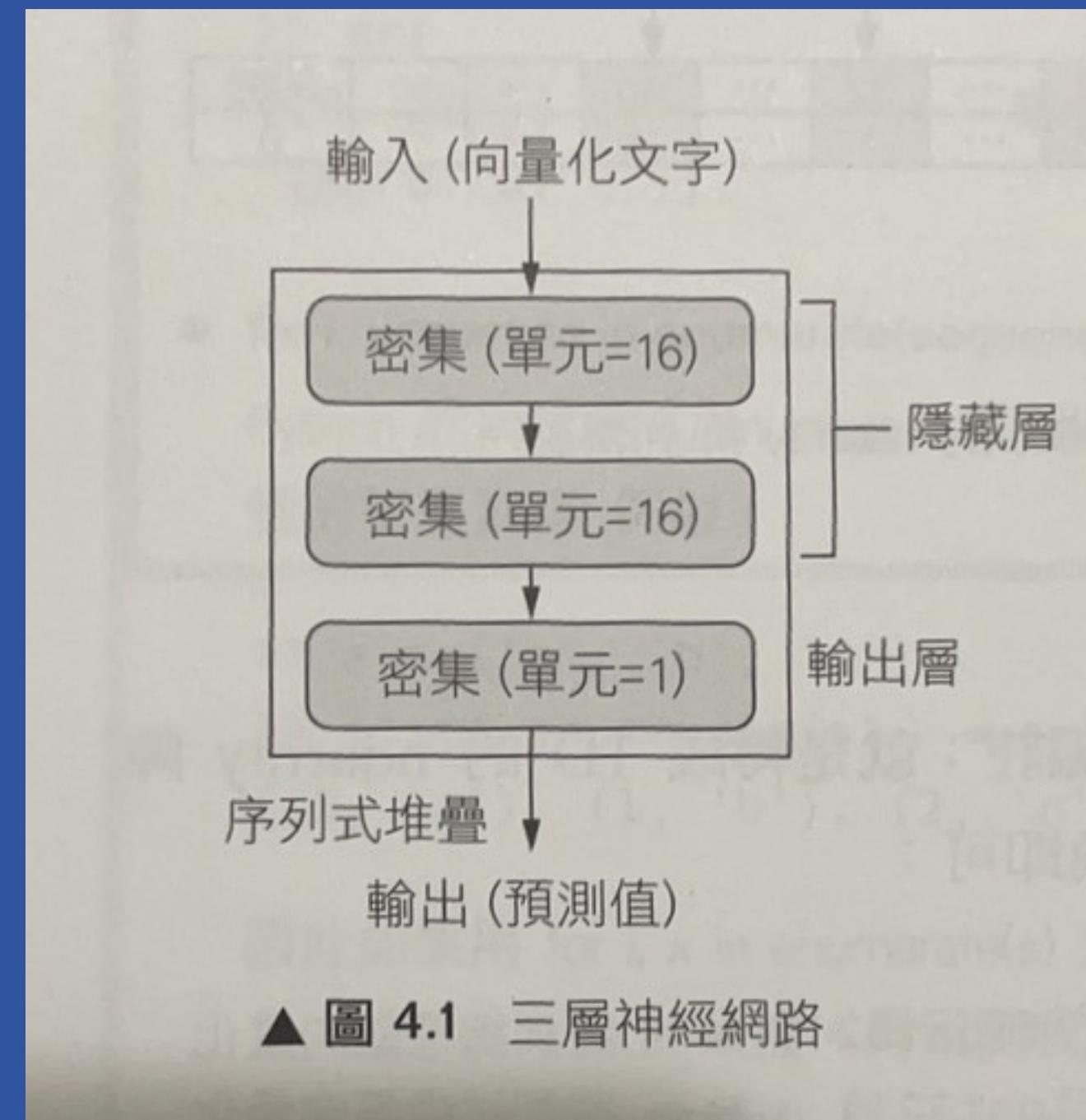


總共有 25000 筆資料

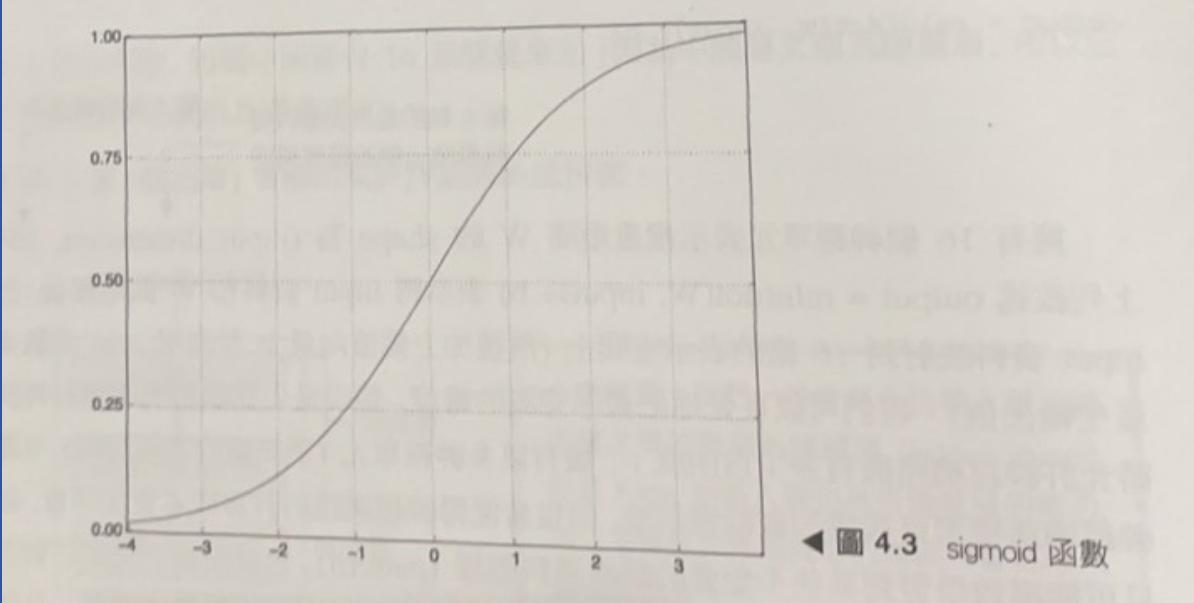
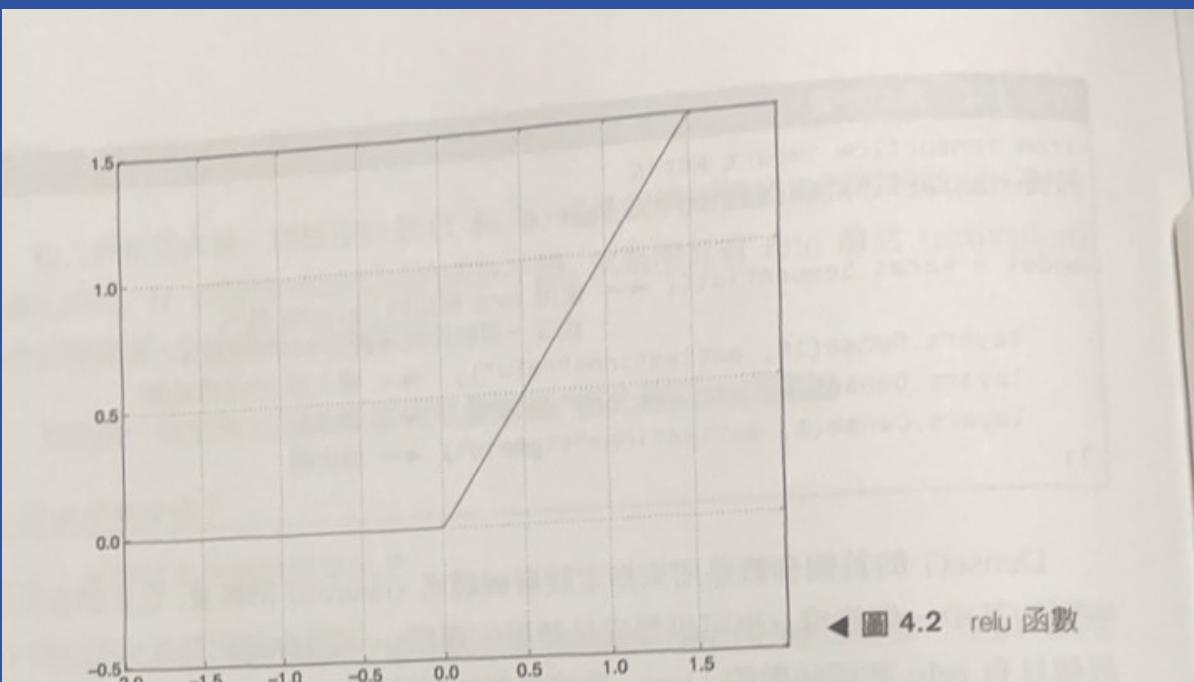
`results[i, sequence] = 1.` 會將 `results[i]` 中的多個元素 (以 `sequence` 串列的每個元素為索引) 都設為 1.0, 例如在第 0 迴圈中 `i` 為 0, `sequence` 的值為 [1, 14, 22, 16, ..., 178, 32], 因此會將 `results[0]` 的第 1, 14, 22, 16, ..., 178, 32 個元素都設為 1.0 :



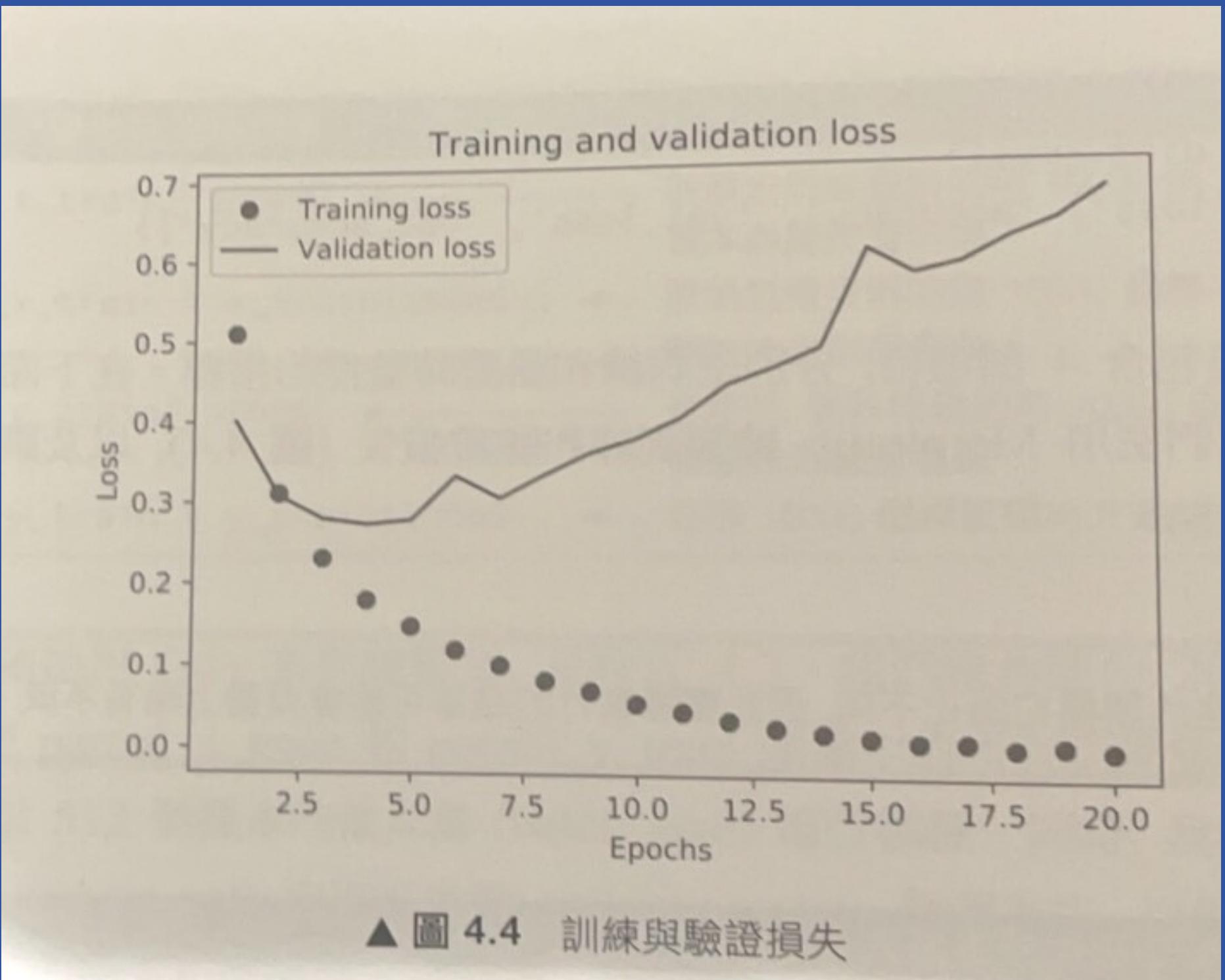
建立神經網路



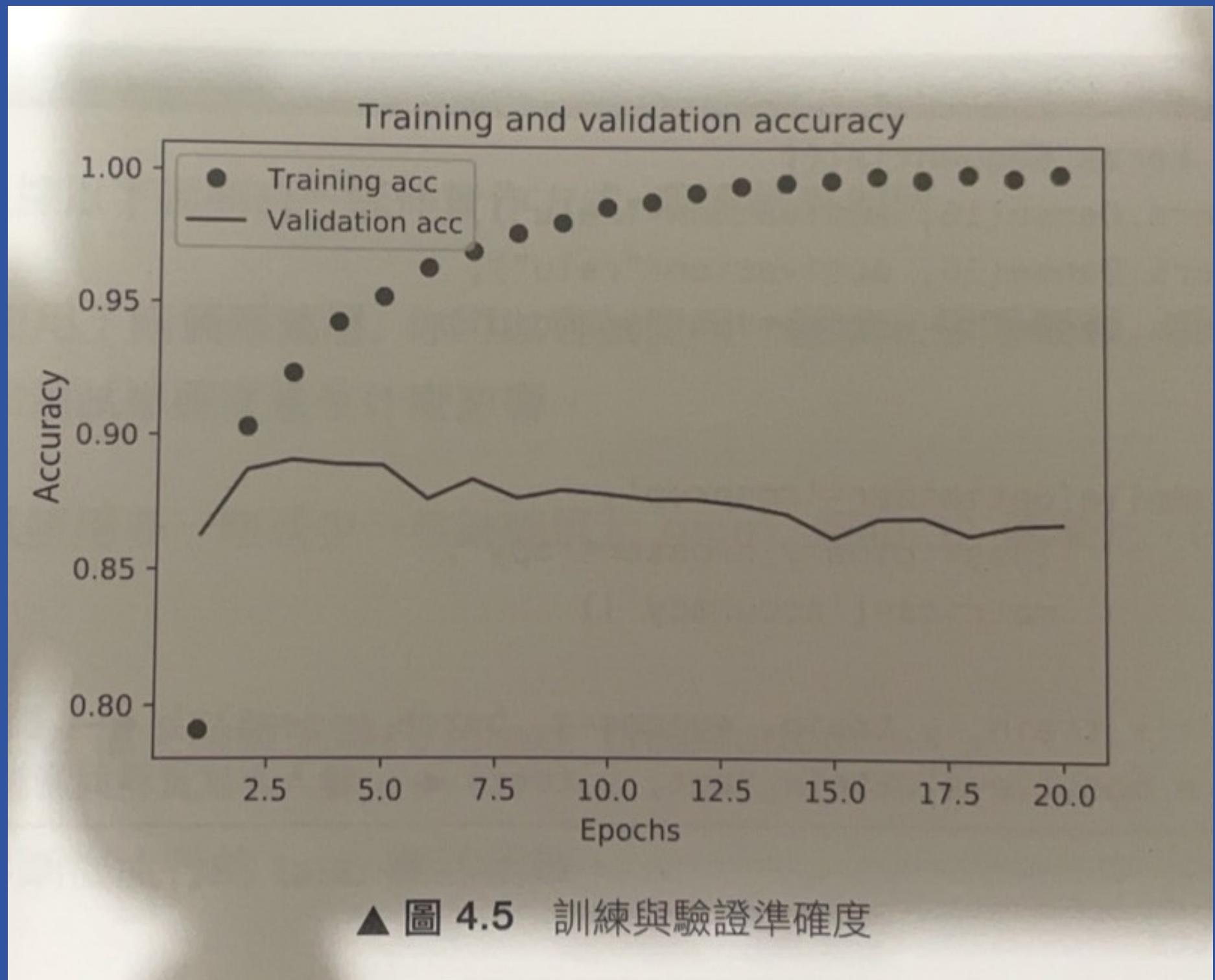
建立神經網路



繪製訓練與驗證損失



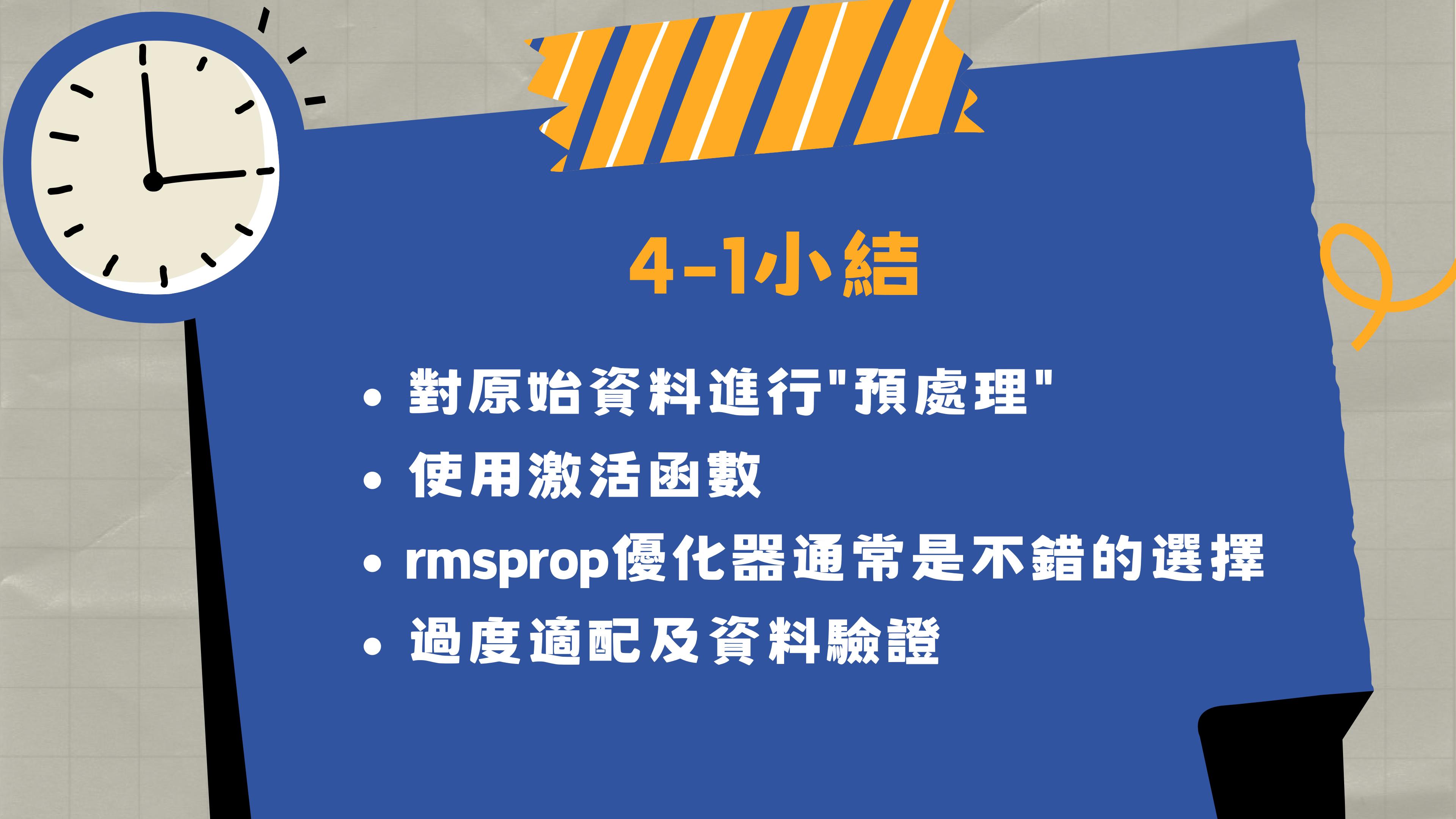
繪製訓練與驗證損失





過度適配(OVERFITTING)

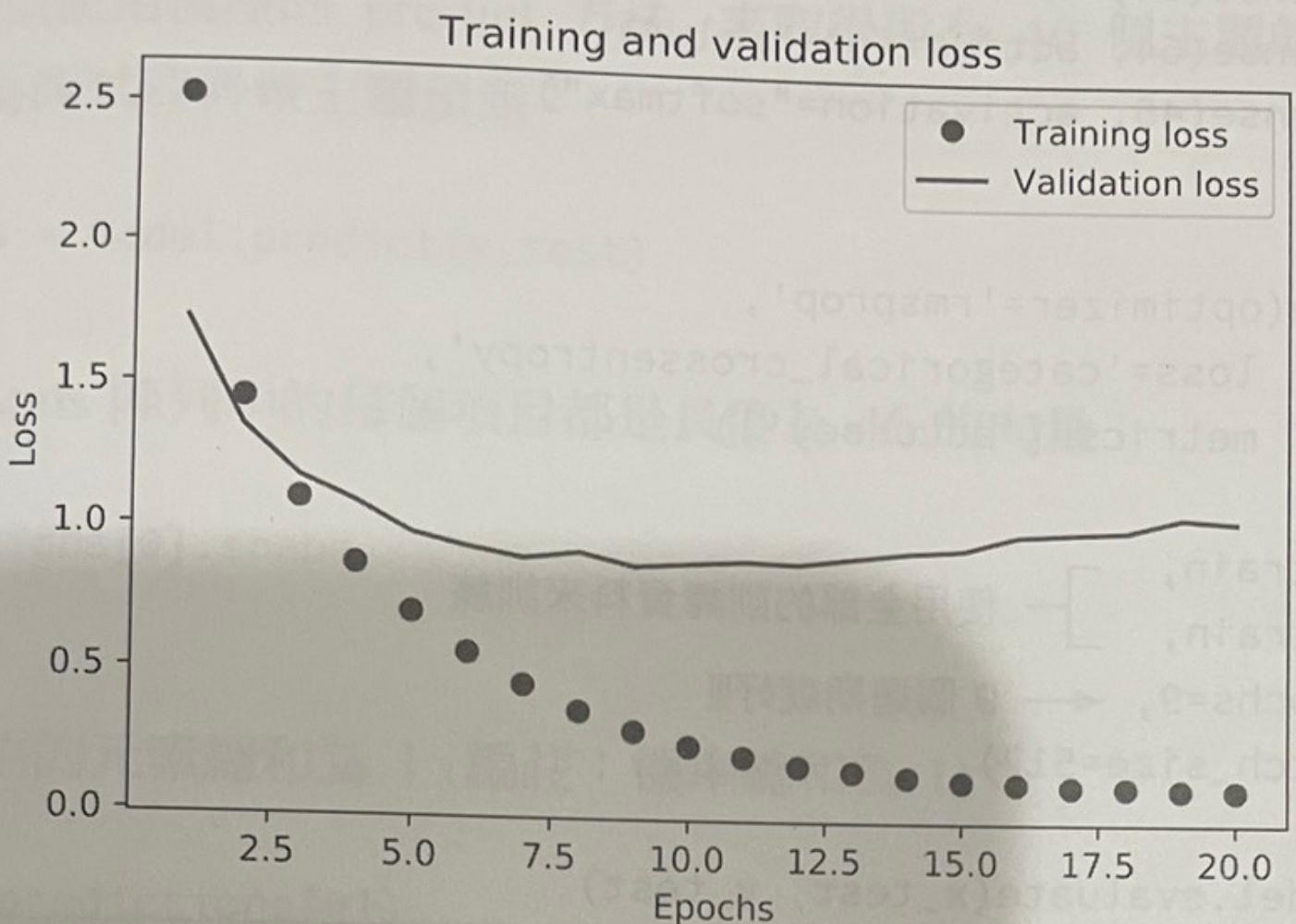
- 在訓練資料上成效不錯的模型，在面對從未見過的資料上不一定有同樣的成效
- 當我們對訓練資料過度最佳化，最終反而學習到特別針對這些訓練資料的表示法，而無法普遍適用於訓練資料以外的其他資料。



4-1小結

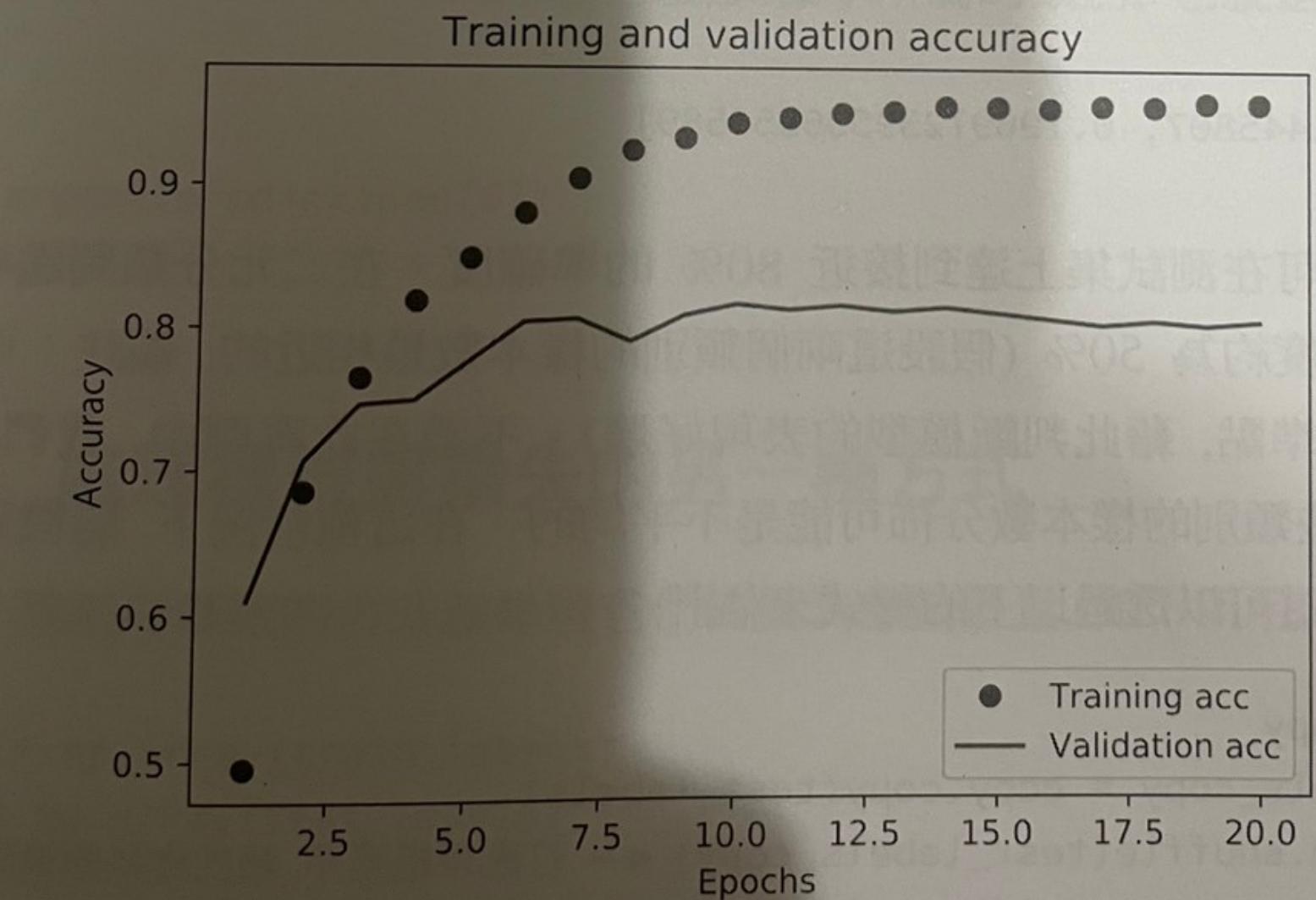
- 對原始資料進行"預處理"
- 使用激活函數
- rmsprop優化器通常是不錯的選擇
- 過度適配及資料驗證

繪製訓練與驗證損失



▲ 圖 4.6 訓練和驗證損失

繪製訓練與驗證損失



▲ 圖 4.7 訓練和驗證準確度



隨機猜測的準確率僅有19%

```
import copy  
test_labels_copy = copy.copy(test_labels)  
np.random.shuffle(test_labels_copy) #打亂標籤順序，藉此作為隨機猜測的結果  
hits_array = np.array(test_labels) == np.array(test_labels_copy) #與標準答案比對，猜對為1，猜錯為0  
float(np.sum(hits_array))/ len(test_labels) #計算準確度 (=隨機猜對的樣本數/樣本總數)
```

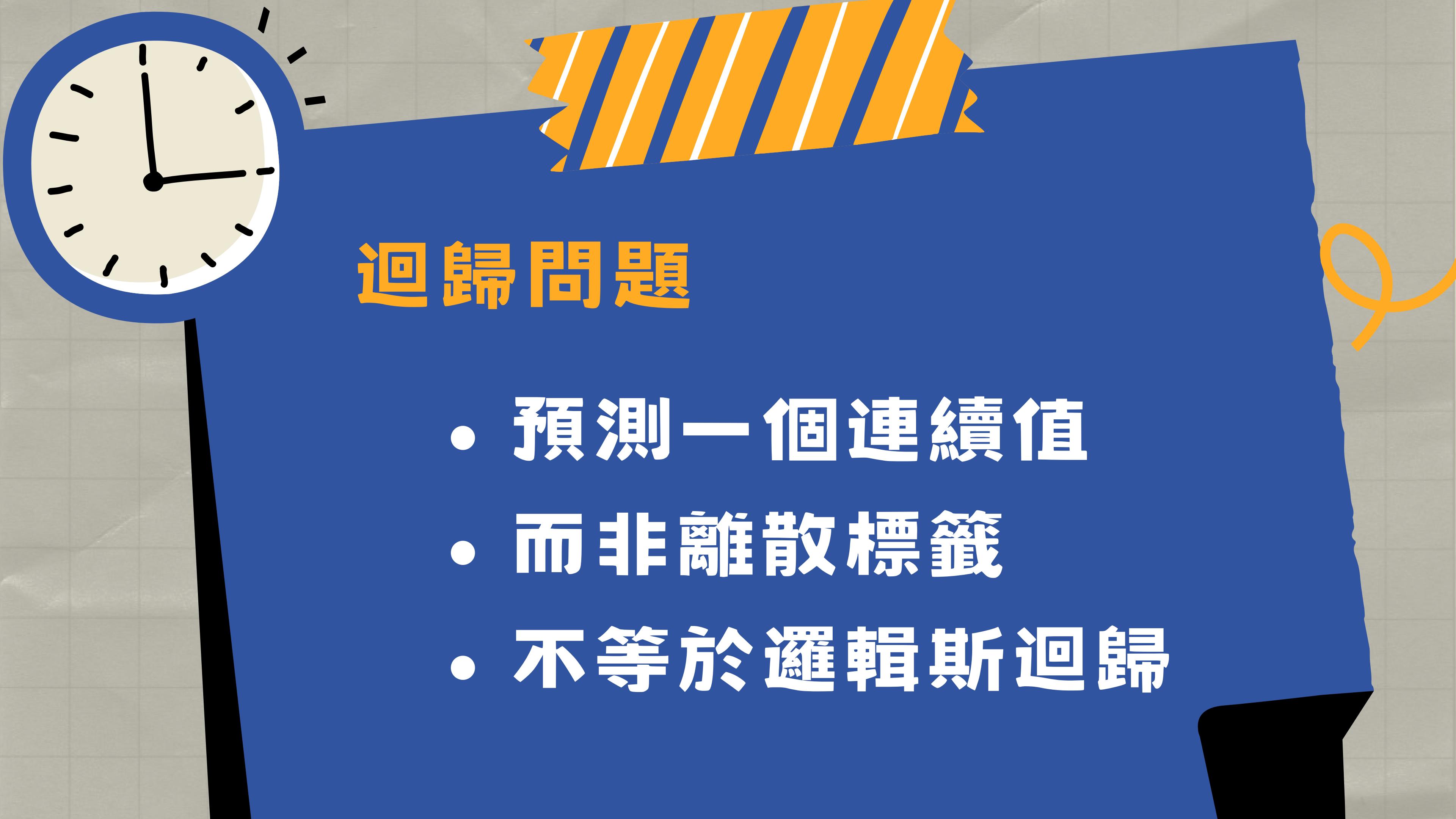
0.1892252894033838

SPARSE_CATEGORICAL_CROSSENTROPY

損失函數	categorical_crossentropy	sparse_categorical_crossentropy
輸入參數	[1, 0, 0]	0
的格式	[0, 1, 0] 分類標籤	1 整數標籤
	[0, 0, 1]	2

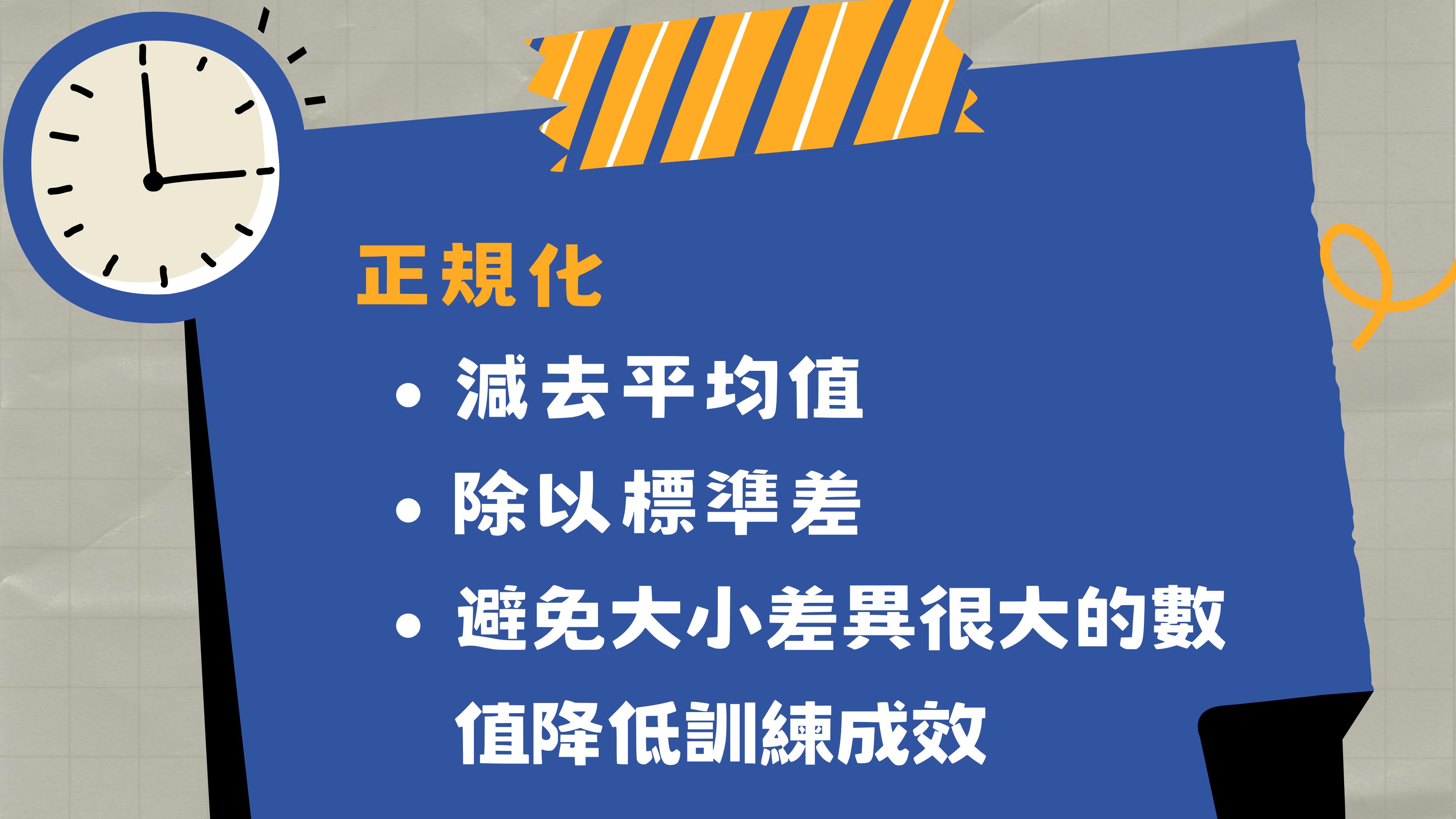
4-2小結

- N個類別就應該選擇N的密集層
- 選用softmax激活函數
- 分類交叉熵
- 分類編碼/整數編碼
- 資訊瓶頸(避免使用太小的中間層)



迴歸問題

- 預測一個連續值
- 而非離散標籤
- 不等於邏輯斯迴歸

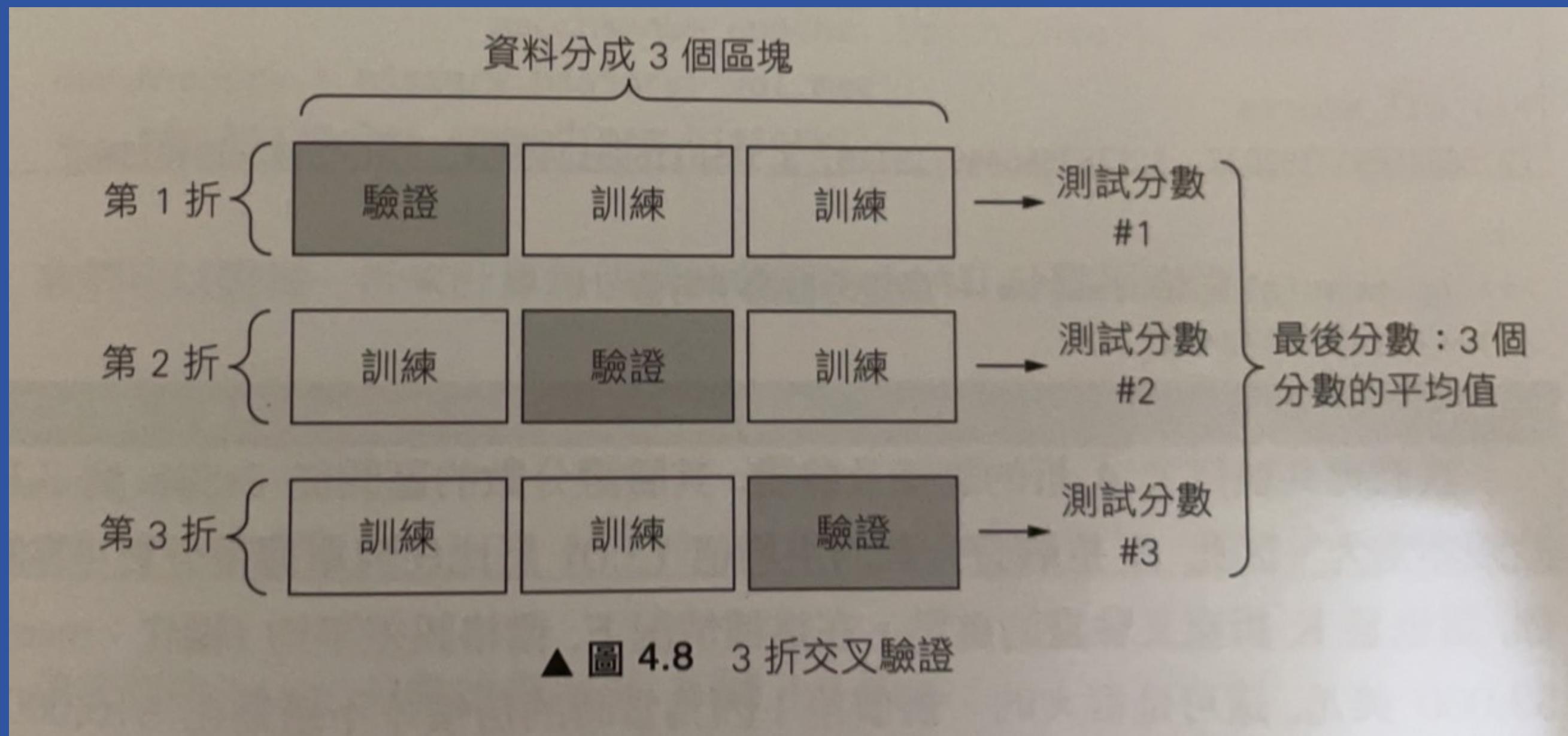


正規化

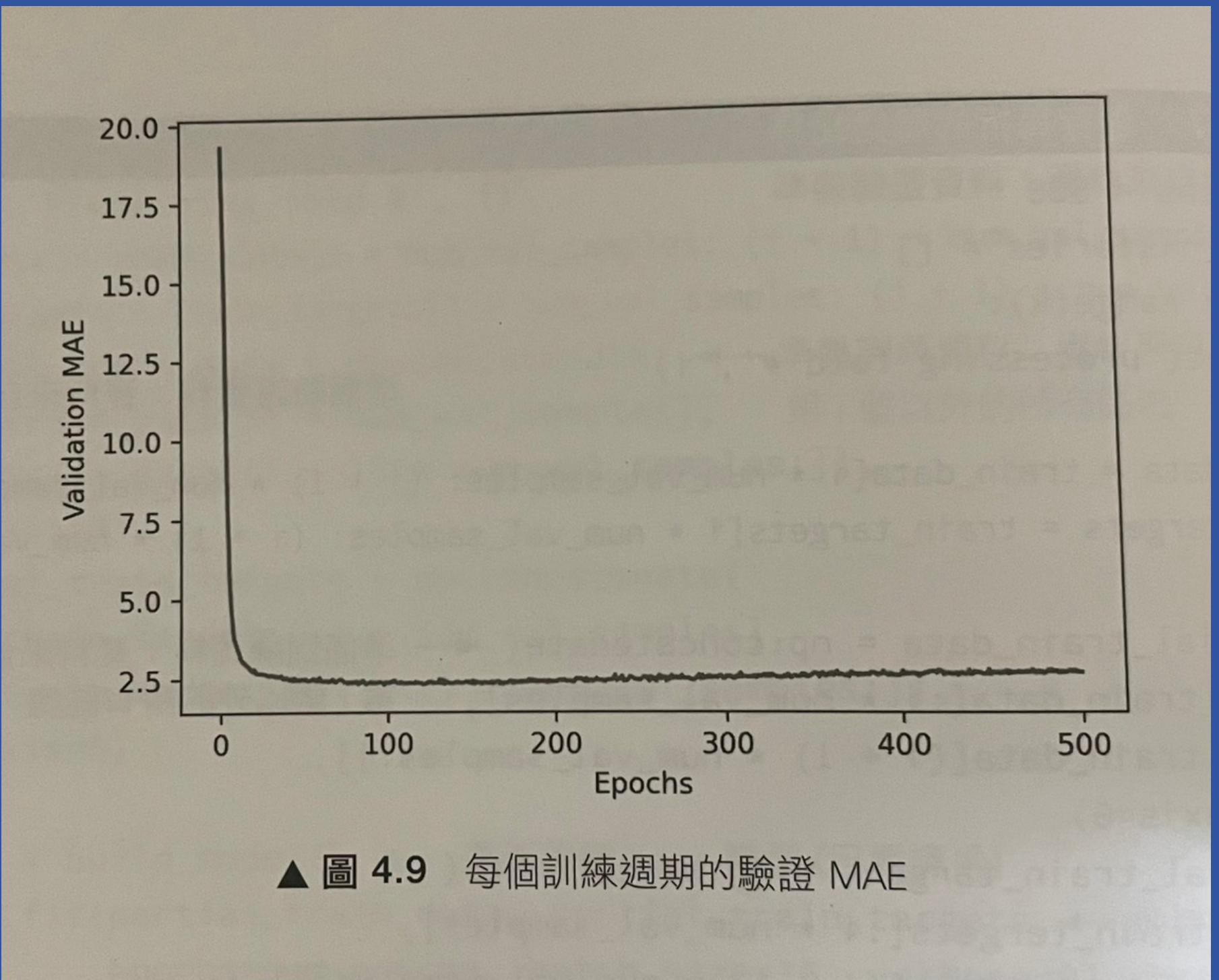
- 減去平均值
- 除以標準差
- 避免大小差異很大的數

值降低訓練成效

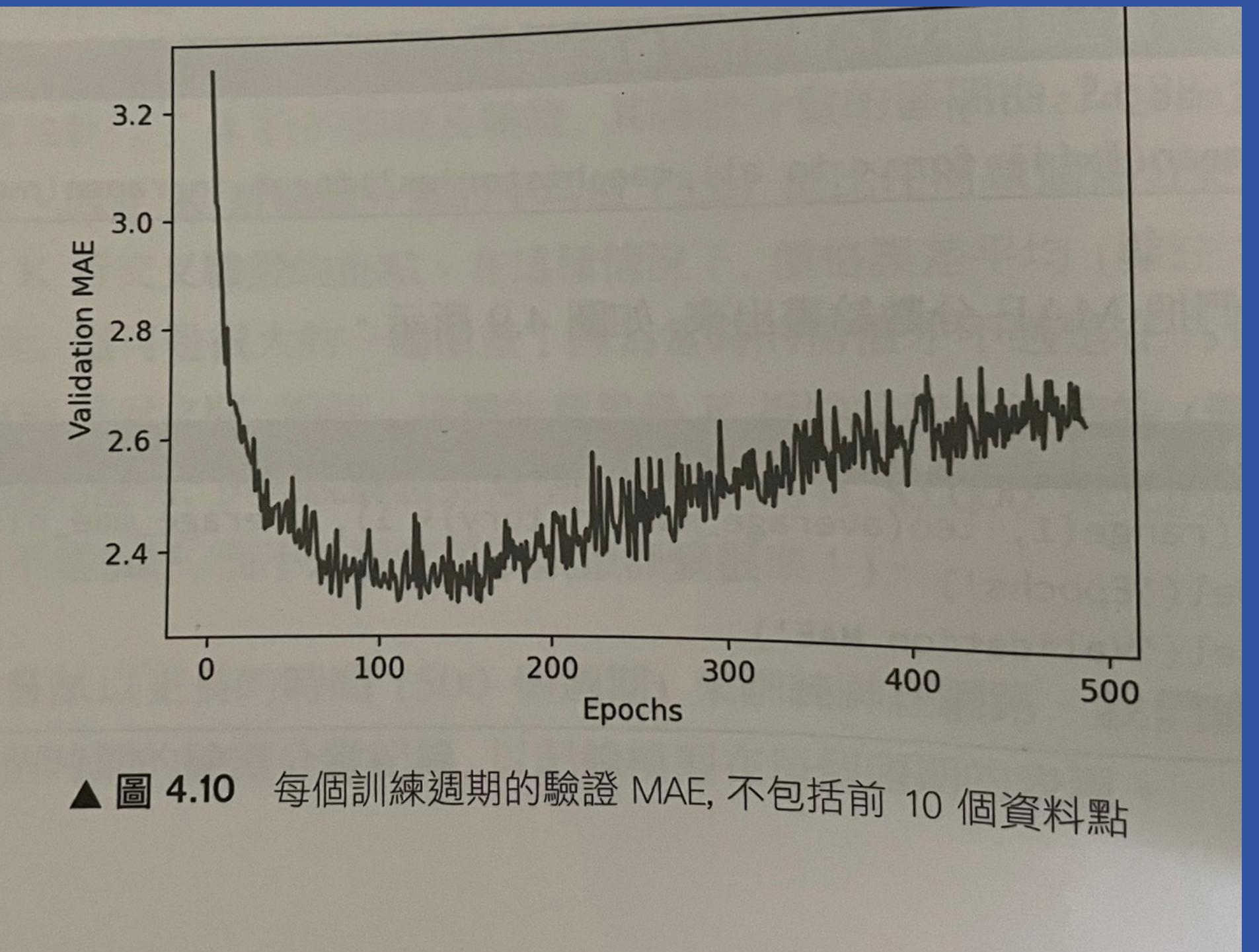
K折交叉驗證

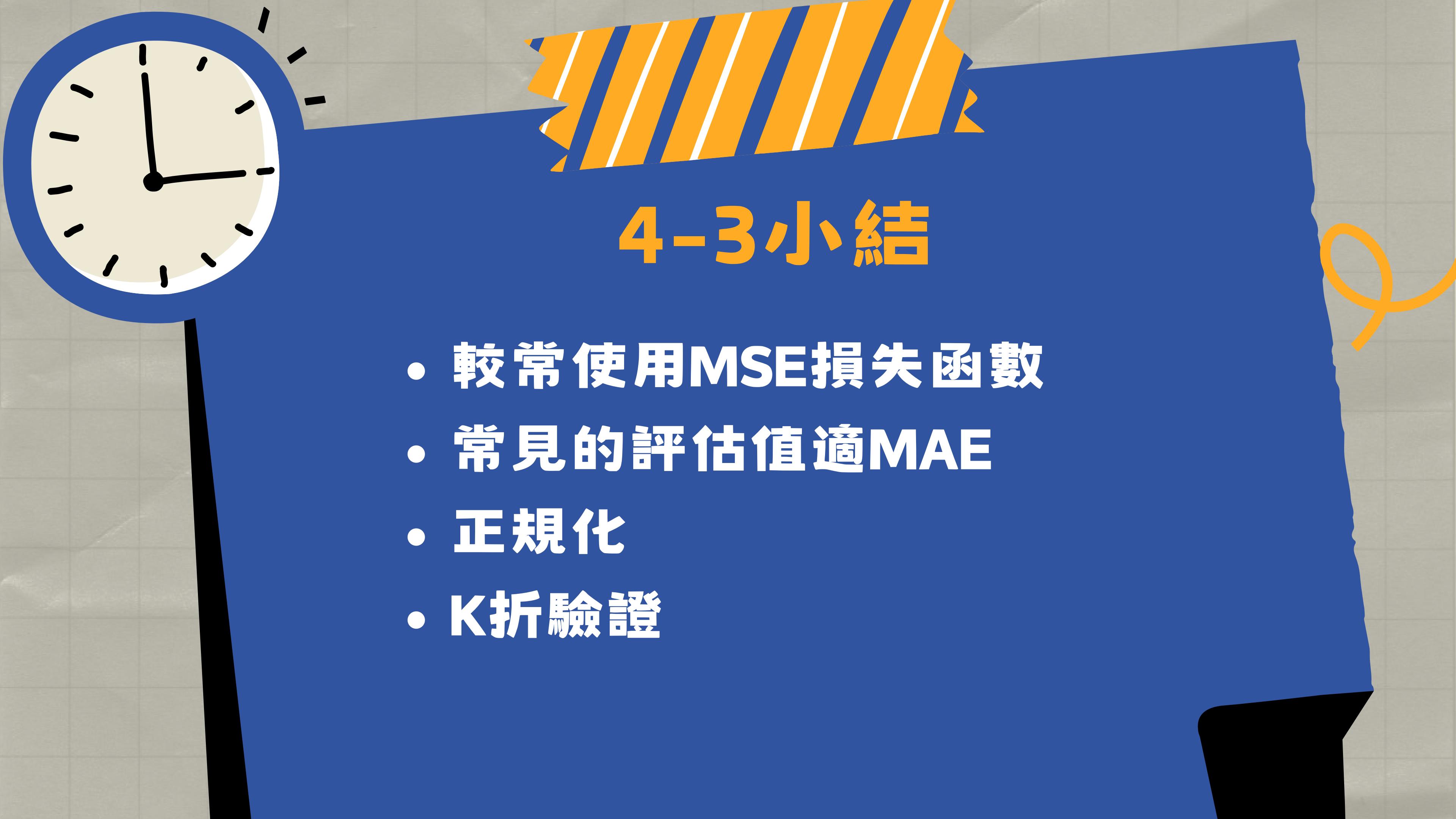


繪製驗證分數



排除前10個資料點





4-3小結

- 較常使用MSE損失函數
- 常見的評估值適MAE
- 正規化
- K折驗證

THE END

