Ch5 機器學習的基礎

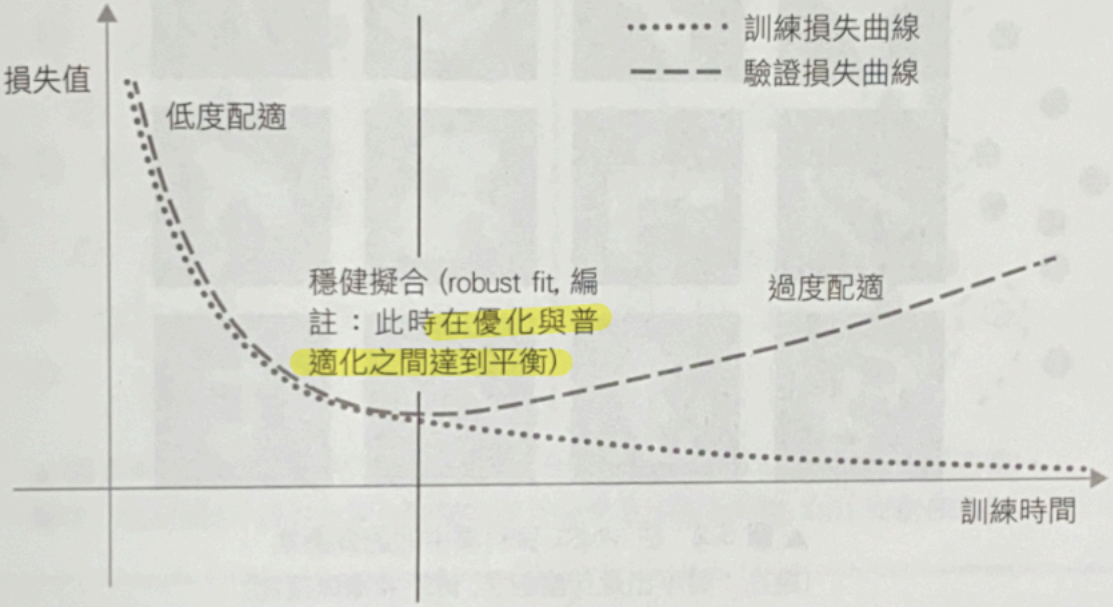
目錄

- 機器學習的終極目標
- 評估機器學習模型
- 提升模型的擬合(fitting)表現
- 提高普適化能力

機器學習的終極目標

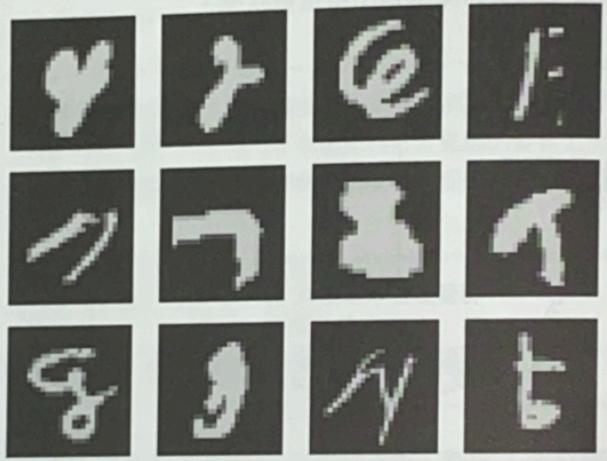
- 資料->訓練集、驗證集、測試集
- 模型在訓練資料表現優異->過度配適
- 優化v.s普適化
- 終極目標：取得普適化良好的模型

低度配適與過度配適

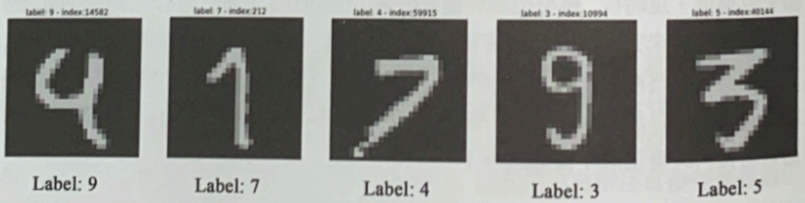


▲ 圖 5.1　典型的過度配適現象

雜訊(noisy)

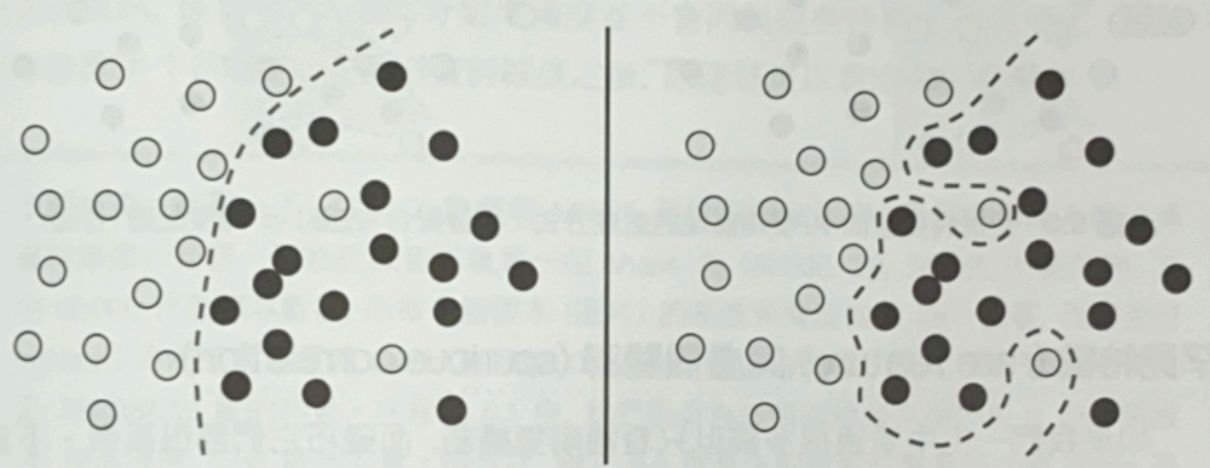


▲ 圖 5.2　在 MNIST 資料集中的詭異圖案
(編註：看不出是什麼數字，對訓練毫無幫助)



▲ 圖 5.3　標籤錯誤的 MNIST 訓練樣本

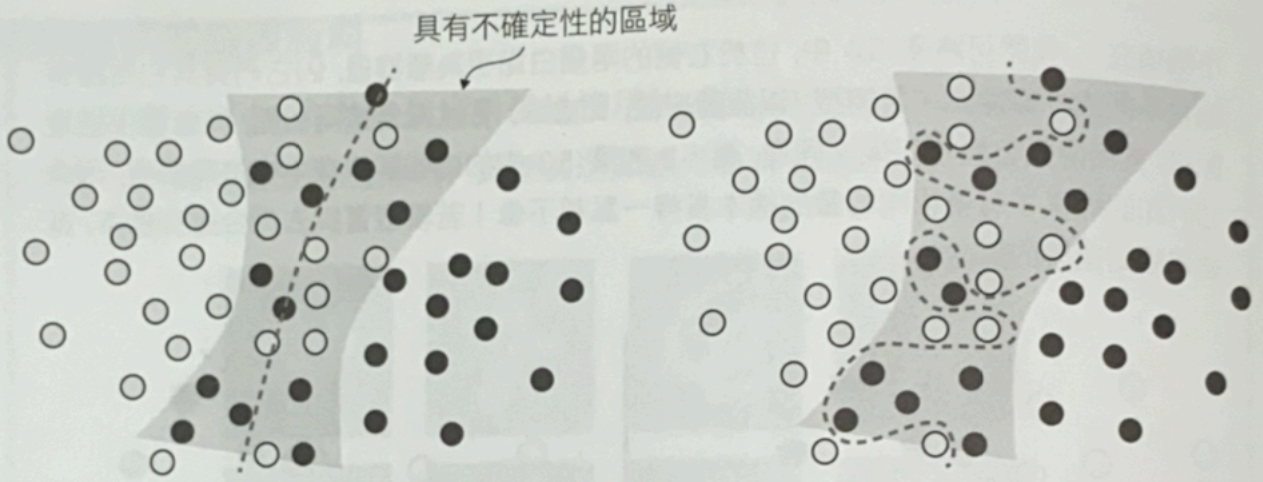
離群值(outlier)



▲ 圖 5.4　處理離群值的方式：穩健擬合 (robust fit, 左圖) vs. 過度配適 (右圖)。

編註：在左圖中, 模型只學到群體特徵而尚未學到離群特徵, 所以較能普適化。

模糊特徵



▲ 圖 5.5　對於特徵空間中, 模糊區域的處理方式：穩健擬合 (左圖) vs. 過度配適 (右圖)

罕見特徵(rare feature)與虛假關聯(spurious correlation)

程式5.1-5.3

使用包含罕見特徵的資料集來訓練模型，也很可能出現過度配適

*並非只有罕見特徵會出現虛假關聯

人類的分類準確度不會因為這些特徵而有所變化

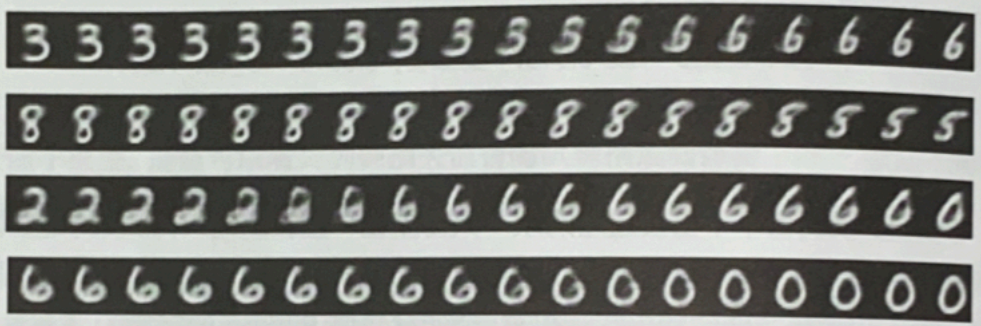
普適化在深度學習中的本質

程式5.4

設計良好的模型會盡力從訓練資料中學習規則，以降低損失率並提高準確度

只要模型參數夠多，就可以成功擬合這些隨機資料，會直接將輸入與標籤關係背起來，類似python字典

流形假說

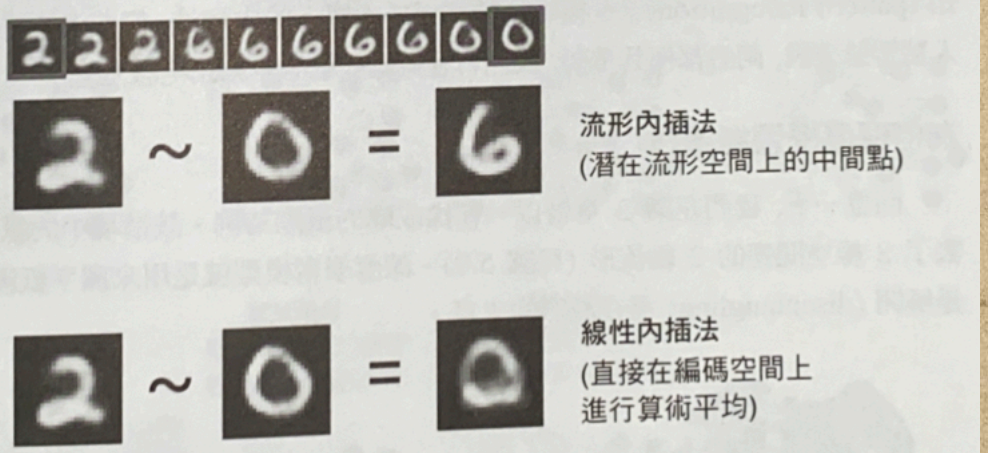


▲ 圖 5.7　不同 MNIST 數字之間可以平滑轉換，代表手寫數字的空間構成了流形。本圖是透過第 12 章的程式來生成

機器學習只需擬合(學習)其輸入空間中的潛在流形即可

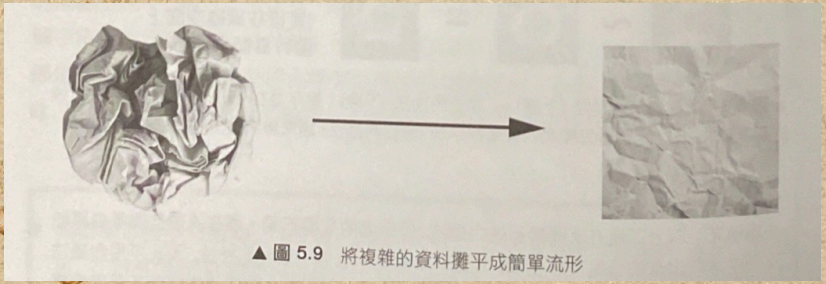
在流形中，可進行內插

內插法



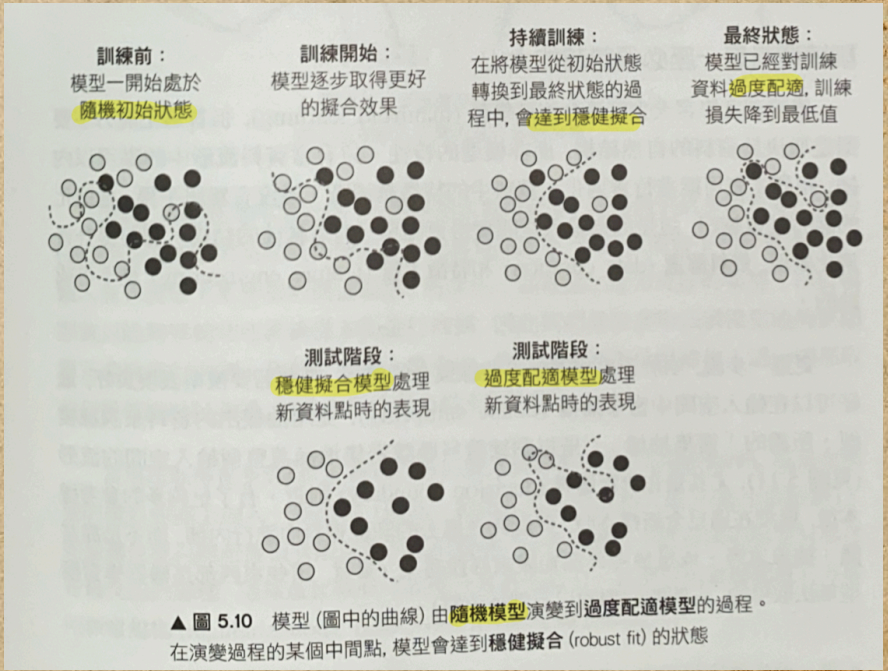
▲ 圖 5.8　潛在流形內插法 (上圖) vs. 線性內插法 (下圖)：數字潛在流形上的每個點都是有效的數字樣本，但直接對兩個樣本的像素取平均 (線性內插法) 通常無法得出有效的數字樣本。

深度機器學習



▲ 圖 5.9　將複雜的資料攤平成簡單流形

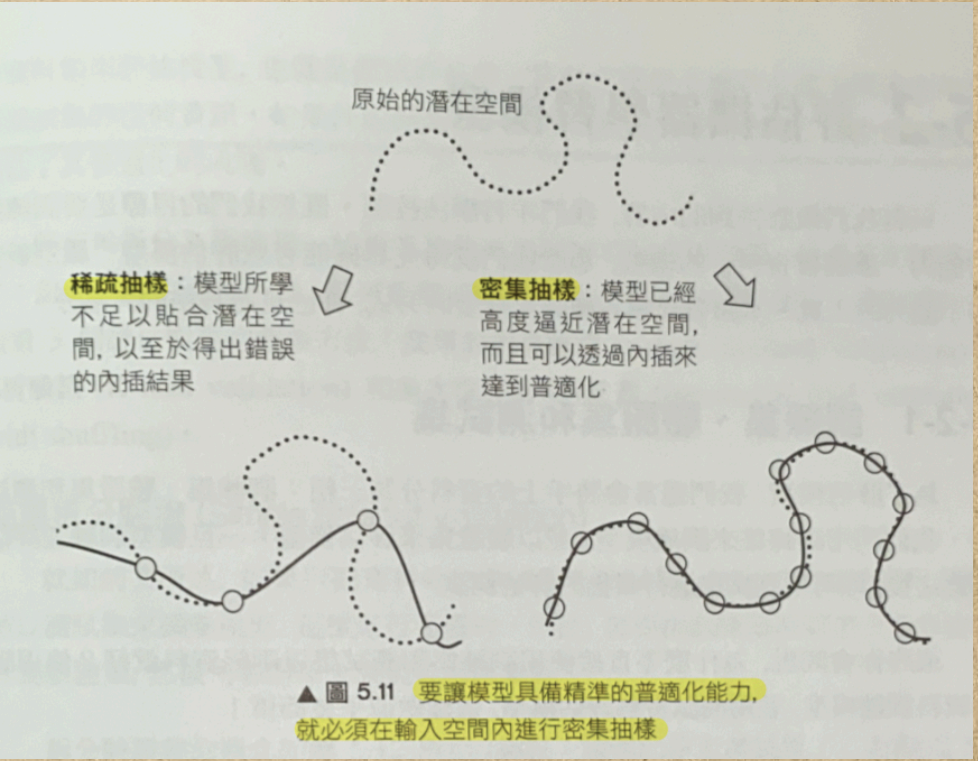
連續且平滑的曲線



▲ 圖 5.10　模型（圖中的曲線）由**隨機模型**演變到**過度配適模型**的過程。在演變過程的某個中間點，模型會達到**穩健擬合**（robust fit）的狀態

曲線移動會與真實的資料流形移動的狀況接近

密集抽樣



▲ 圖 5.11 要讓模型具備精準的普適化能力，就必須在輸入空間內進行密集抽樣

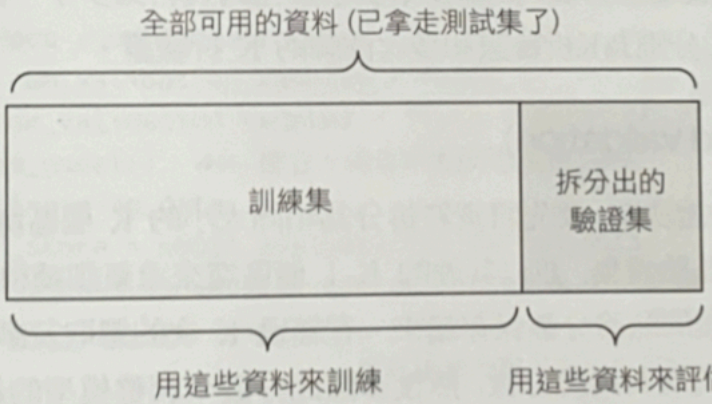
無法取得更多資料時->調降模型能容納的資訊、在模型的擬合曲線加入限制->可提升普適化
以上對抗過度配適之方法->常規化

評估機器學習模型

- 資料->訓練集(訓練模型)、驗證集(評估模型)、測試集(最後的評估測試)
- 資料洩漏
- 優化模型
- 簡單拆分驗證、K折驗證、 多次洗牌K折驗證

簡單拆分驗證 程式5.5

由訓練集切出一部分資料作為驗證集，用該驗證集來調整模型超參數



▲ 圖 5.12 簡單拆分驗證

程式 5.5 簡單拆分驗證 (為了方便處理，此處省略了處理標籤的流程)

```
num_validation_samples = 10000   ← 設定驗證集樣本數為 10000 筆
np.random.shuffle(data)          ← 打亂樣本的順序
validation_data = data[:num_validation_samples]   ← 抽取前 10000 筆樣本為驗證集
```

▶接下頁

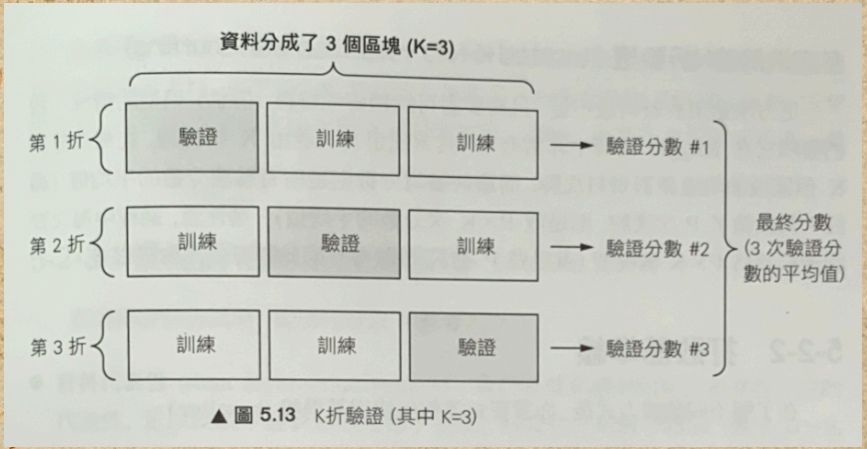
```
training_data = data[num_validation_samples:]   ← 剩餘的樣本為訓練集
model = get_model()   ← 建構模型 (請參考程式 5.2)
model.fit(training_data, ...)   ← 以訓練集資料來訓練模型
validation_score = model.evaluate(validation_data, ...)   ← 以驗證集來評估模型

...   ← 調整模型 (編註：即調整程式 5.2 的 get_model() 函式中的模型超參數)，
        再進行訓練與評估，然後再調整模型，不斷重複直到對驗證分數滿意

model = get_model()   ← 重新建構模型，此時 get_model() 函式中的
                        模型超參數都已手工調整到最佳值
model.fit(np.concatenate([training_data, validation_data]), ...)   ← 當完成超參數的調整後，通常會用測試資料以外的資料 (訓練集+驗證集) 再重新訓練一次，此處使用 np.concatenate() 將訓練集和驗證集合併在一起

test_score = model.evaluate(test_data, ...)   ← 測試模型
```

如果重新洗牌後所訓練出的模型差異很大，以[K折驗證]、[多次洗牌的K折驗證]來解決

K折驗證

程式5.6

將資料拆分為相同大小之K個區塊



▲ 圖 5.13 K折驗證 (其中 K=3)

當模型因為資料的隨機拆分產生明顯差異時->利用此方法解決

程式 5.6 K 折交叉驗證 (此處 K=3)

```
k = 3
num_validation_samples = len(data) // k ← 算出每個區塊內的資料樣本數
np.random.shuffle(data)
validation_scores = []
for fold in range(k):                                          選擇驗證資料區塊
    validation_data = data[num_validation_samples * fold:
                           num_validation_samples * (fold + 1)]
    training_data = np.concatenate(                            使用剩餘的資料
        data[:num_validation_samples * fold],                  做為訓練資料
        data[num_validation_samples * (fold + 1):])
    model = get_model()  ← 建立一個全新的模型(未經訓練)
    model.fit(training_data, ...)                              取得驗證分數
    validation_score = model.evaluate(validation_data, ...)
    validation_scores.append(validation_score) ← 將每一折的驗證
                                                 分數存進串列
validation_score = np.average(validation_scores)
                                 最終驗證分數是每一折驗證分數的平均值
```

編註：以上程式可不斷重複進行，並依照上一行的**最終驗證分數**來調校模型，直到滿意為止。然後再進行以下程式，用調好參數的最佳模型重新進行完整訓練，並用測試集評估成效：

```
model = get_model()  ← 重建模型 (其超參數已人工調整到最好了)
model.fit(data, ...) ← 使用所有的非測試資料進行訓練
test_score = model.evaluate(test_data, ...) ← 使用測試集做最後的評估
```

多次洗牌的K折驗證

多次應用K折驗證，在每次分割K個區塊前重新對資料洗牌，最終驗證分數為所有驗證分數之平均值

運算成本過高

基準線

模型表現超過基準線->正確可繼續往下走

評估模型之注意事項

- 資料代表性

  ->對資料隨機洗牌，使訓練集跟測試集有一定代表性
- 時間方向性

  ->確保測試資料的發生時間在訓練資料後
- 資料中的重複現象

  ->確保訓練集跟驗證集之間無交集

提升模型的擬合（fitting）表現

- 訓練出一個能展現基本的普適化能力，且會發生過度配適之模型

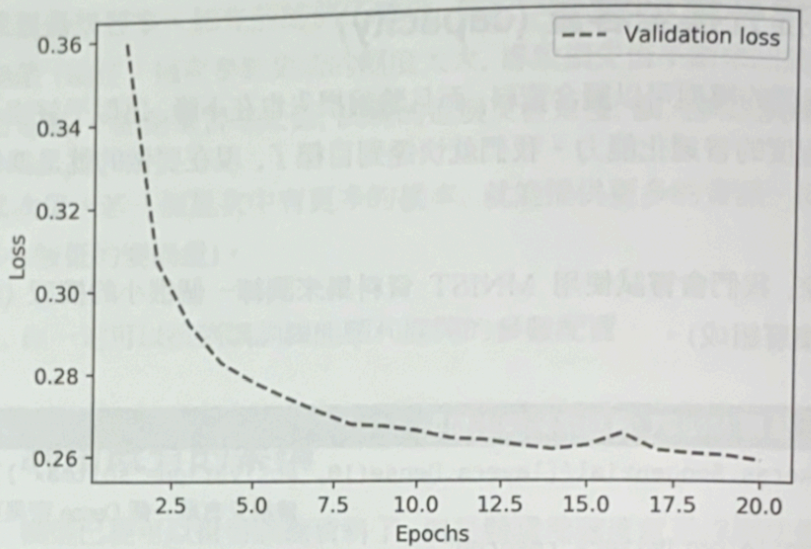
- 遇到之問題

1. 訓練無成效
2. 訓練成效尚可
3. 無法達到過度配適

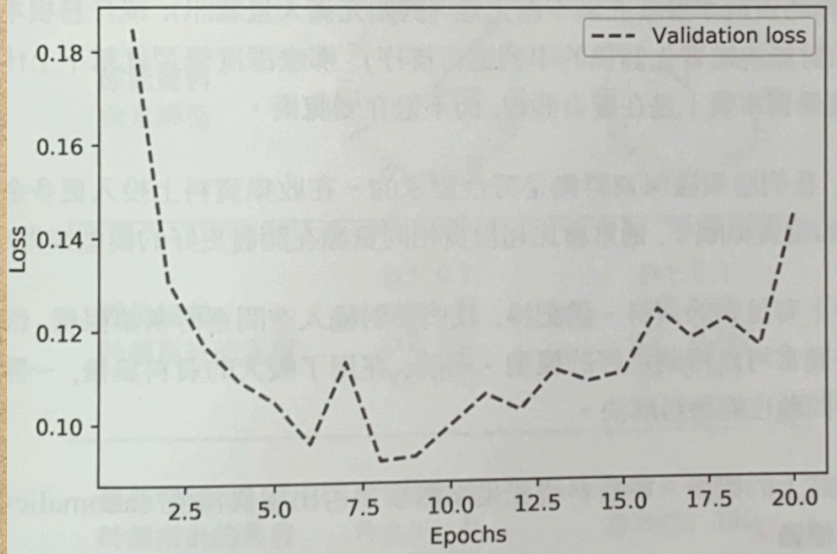
調整梯度下降的關鍵參數 程式5.7

降低或提高學習率、增加批次量

提升模型容量 程式5.9



▲ 圖 5.14　模型容量不足對損失曲線的影響



▲ 圖 5.15　當模型具有適當的容量時, 驗證損失的變化曲線

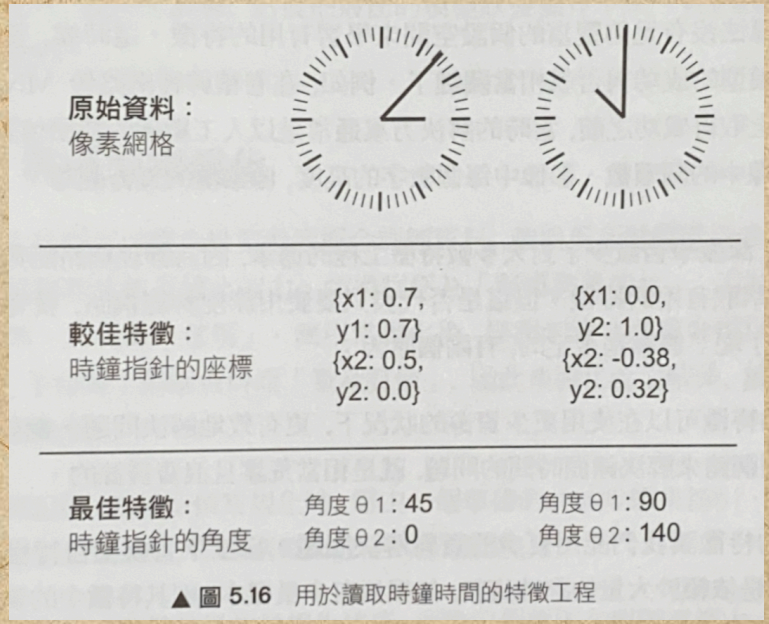
提高普適化能力

- 資料集篩選
- 特徵工程
- 使用早期停止
- 將模型常規化

資料集篩選

- 確保有足夠資料
- 減少標註上的錯誤
- 清理資料並處理缺失值
- 如果手上有很多特徵，但不知道那些有用，那麼先做特徵選擇

特徵工程



▲ 圖 5.16 用於讀取時鐘時間的特徵工程

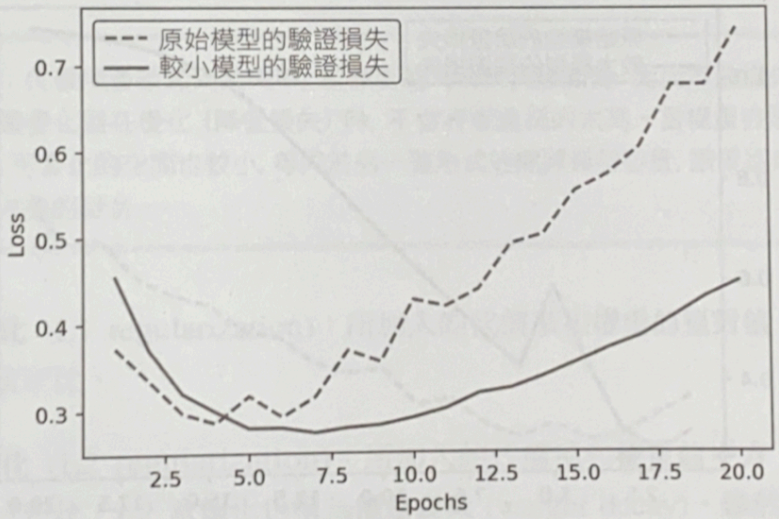
透過更簡單的方式來表示問題，使問題更容易處理。同時，使潛在流形更平滑、更簡單、更有組織性，但通常需要深入了解問題，才能進行特徵工程

使用早期停止

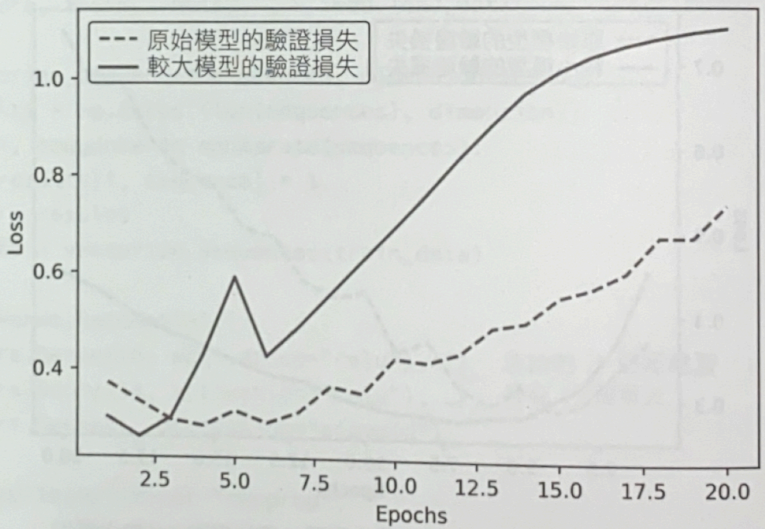
在訓練階段找出達到最佳普適化的時間點，並在該點停止訓練

將模型常規化

- 縮減神經網路的規模(程式5.10 11 12)



▲ 圖 5.17 IMDB 影評分類的表現：原始模型 vs. 較小模型



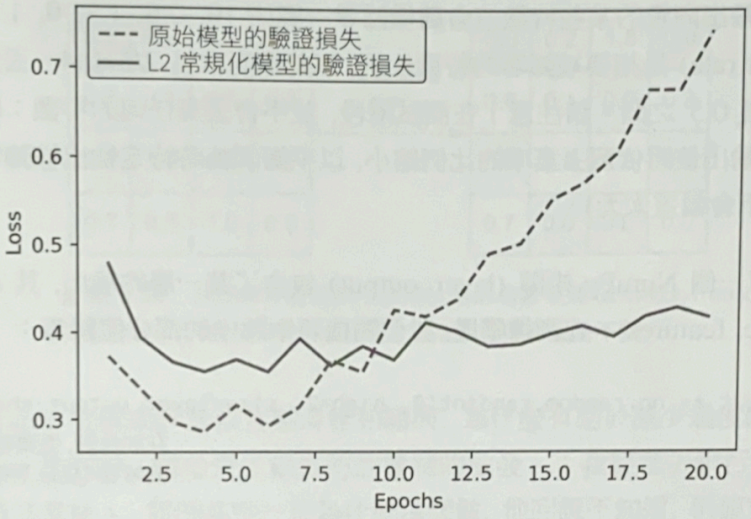
▲ 圖 5.18 IMDB 影評分類上的表現：原始模型 vs. 較大模型

將模型常規化

✢ 加入權重常規化(程式5.13 14)

L1常規化：所加入的代價項和權重的絕對值成正比
L2常規化：所加入的代價項和權重的平方成正比

* 有L2常規化的模型會變得比原始模型更能抵抗過度配適

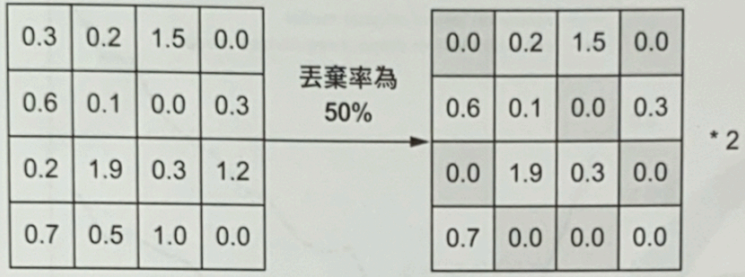


▲ 圖 5.19 L2 權重常規化對驗證損失的影響

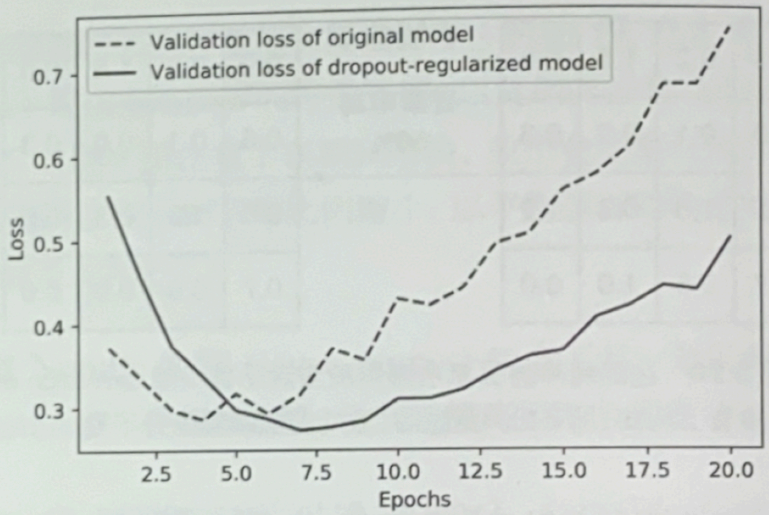
將模型常規化

✢ 加入丟棄法(大型)(程式5.15)

在訓練期間隨機丟棄神經網路層的一些輸出特徵(把特徵值設為0)



▲ 圖 5.20 在訓練時將丟棄法應用於激活函數矩陣 (activation matrix), 並緊接著 / = 0.5 (即×2)。測試時, 激活函數矩陣保持不變



▲ 圖 5.21 使用丟棄法對驗證損失的影響

THANK