



# 第六章

# 機器學習與工作流程

Hannah Morales

# 章節

6-1 定義任務

6-2 開發模型

6-3 部屬模型

# 6-1 定義任務

須對處理的任務有足夠的認識，才會有好的結果。

# 6-1-1 定義問題範圍

以下問題應擺在第一順位：

1. 輸入的資料是甚麼？想預測甚麼？
2. 面對的是甚麼類型的機器學習任務？
3. 現有解決方案是什麼樣子？
4. 是否需要考量特定的限制條件？  
@reallygreatsite

完成上述研究後，可掌握輸入為何、目標值為何，  
以及當前問題對應到的機器學習任務種類為何。

在這階段所做的假設：

1. 假設可以透過輸入來預測出目標值。
2. 假設現有(或收集到)的資料具備足夠資訊，可用來學習輸入與目標值間的關係。

# 6-1-2 建立資料集

1. 照片搜尋引擎專案
2. 偵測垃圾內容專案
3. 音樂推薦引擎
4. 異常餅乾偵測模型
5. 衛星影像專案

# 投資在資料標註工具

需先考慮以下選項：

1. 需要自己標註資料嗎？
2. 需要使用外包平台來收集標籤嗎？
3. 需要使用專業資料標註公司的服務嗎？

外包或許可以節省成本和時間，但同時也難控制標註品質。

若想找出最佳選項，若想找出最佳選項，  
可以考慮以下面向：

- 1.一定需要特定領域的專家，還是誰都可以？
- 2.如果需要特定知識才能標註資料，那可以順練別人來做嗎？如果無法，那如何找到相關領域專家？
- 3.了解專家怎麼標註嗎？如果不了解，只能把資料集當作黑盒子，而且無法手動進行特徵工程。

# 留意不具代表性的資料

訓練資料一定要有足以代表實際運作的資料

概念飄移:根據時機資料的特性不斷變動，導致模型準確度下降。

# 6-1-3 理解資料

應先探索和視覺化資料，已對資料有整體概念。

1. 資料包含影像或自然語言
2. 資料包含數值特徵
3. 資料包含位置訊息
4. 樣本是否缺少某些特徵值
5. 樣本數量是否相近
6. 是否存在目標值洩漏的問題

# 6-1-4 選擇測量成效的方法

若想控制某些事物，我們必須能夠觀察他。

何謂「成功」？準確度？精準度？  
故障召回率？客戶回流率？

平衡分類問題

## 6-2 開發模型

開發模型只是機器學習工作流程中的其中一步，還不是最難的部分。在機器學習中，最難的部分式定義問題範圍以及收集、標註並清理資料。

# 6-2-1 準備資料

深度學習通常無法接受原始資料，而需要先經過處理。

1. 向量化
2. 正規化
3. 處理缺失值

# 向量化

神經網路的所有輸入和目標值必須是浮點數張量。

# 數值正規化

一般來說，將相對大的數值或異值資料輸入神經網路  
並不安全。

為了更容易學習，需有以下特性：

1. 數值較小
2. 具備同質性
3. 單獨正規化，使平均為0
4. 單獨正規化，使標準差為1

- 單獨正規化每個特徵，使標準差為 1。

NumPy 陣列可以很容易執行正規化：

```
x -= x.mean(axis=0) ← 假設 x 是 shape 為 (樣本數, 特徵) 的 2D 資料矩陣  
x /= x.std(axis=0)
```

# 處理缺失值

可選擇忽略或以下選項：

- 1.如果這個特徵是分類，可為該特徵新增一個分類值
- 2.如果這個特徵是數值，避免隨意替代缺失值

## 6-2-2 選擇驗證機制

模型最終目標是要實現普適化，每個決定都是由驗證評量指標所引導，可用來衡量普適化表現。

1. 拆分驗證集
2. K折交叉驗證
3. 多次迭代的K折驗證

# 6-2-3 超越基準線

第一個目標是取得統計能力。

- 1.特徵工程
- 2.選擇適合既有架構
- 3.選擇足夠好的訓練配置

## 選擇合適的損失函數

一般來說，不太可能直接對評量指標（例如：準確度）進行優化。有時，也很難將評量指標轉化為損失函數；畢竟損失函數要能在小批次（mini-batch）上計算（理想中，即使只給定單一的資料點，也應該要能算出損失函數值），而且還必須可微分（否則就無法使用反向傳播來訓練模型）。舉例來說，廣泛使用的分類任務評量指標 ROC/AUC 就沒辦法直接用來優化模型。因此在分類任務中，時常會用其它指標來代替 ROC/AUC（例如：交叉熵 crossentropy）以進行優化。一般來說，如果交叉熵越低，ROC/AUC 就會越高。

下表可協助我們針對一些常見的問題類型，決定輸出層的激活函數與損失函數：

為模型輸出層選擇激活函數與損失函數

問題類型	輸出層激活函數	損失函數
二元分類	sigmoid	binary_crossentropy
多類別、單標籤分類	softmax	categorical_crossentropy
多類別、多標籤分類	sigmoid	binary_crossentropy

# 6-2-4 擴大規模：開發一個會 過度配適的模型

需先開發出會過度適配的模型：

1. 添加更多神經層
2. 讓每一神經層更寬
3. 訓練更多週期

# 6-2-5 將模型常規化 並調整超參數

最大化其普適化能力。

1. 嘗試不同架構
2. 使用丟棄法
3. 如果模型不大，嘗試用L1L2常規化
4. 嘗試不同超參數
5. 嘗試使用資料篩選

丟棄法:指在訓練神經網路過程中隨機丟掉一部分神經元來減少神經網路複雜度，從而防止過度。

L1常規化:把模型裏頭所有的參數都取絕對值。

$$L1: \|\boldsymbol{w}\|_1 = \sum_{j=1}^m |w_j|$$

L2常規化:把模型裏頭所有的參數都取平方求和。

$$L2: \|\boldsymbol{w}\|_2^2 = \sum_{j=1}^m w_j^2$$

# 6-3 部屬模型

@reallygreatsite

# 6-3-1 向客戶說明成 果，並建立合理期待

所謂開發成功與客戶信任，都來自於滿足甚至超越期待。

## 6-3-2 交付推論模型

首先，需要將模型匯出至Python以外環境：

1. 生產環境未必支援Python

2. 應用程式的其餘部分未必是以Python運行

# 以REST API部署模型

將模型轉為產品常見方式，是在伺服器或雲端虛擬機上安裝Tensorflow，然後透過網路以REST API來呼叫模型並取得結果。

1. 需取得模型預測結果的應用程式可以穩定連上網路
2. 應用程式延遲時間要求不高
3. 用以推論的輸入資料不敏感(不具隱私性)

# 以裝置上部屬模型

有時，我們需要在運行應用程式的同一裝置上運作模型，可能是手機、ARM CPU的機器人或是小型裝置上的控制器。

1. 模型需有嚴格的延遲時間限制
2. 可以設計非常小，才能夠在記憶體和功率限制下工作
3. 首要任務並非取得盡可能高的準確度
4. 輸入資料敏感(隱私要求很高)

# 以瀏覽器上部屬模型

深度學習常用在瀏覽器或是桌面應用的JavaScript程式中。



1. 想把運算成本給使用者，降低伺服器成本
2. 輸入資料必須留在使用者手機或電腦。
3. 模型需有嚴格的延遲時間限制
4. 下載模型緩存後，在不連線下仍可運作

# 優化推論模型

當部屬在記憶體和功率有限制的環境下，或是有低延遲需求時，優化模型對推論就顯得重要。

1. 權重剪枝:並非每個參數都對預測結果有貢獻。
2. 權重量化:在推論時，可將權重化成8為元整數，可縮小模型 $1/4$ ，也可接近相同水準。

## 6-3-3 監控模式運作狀況

部屬完模型，也須持續監控模型行為、掌握表現、觀察模型與應用程式互動、以及最終如何影響商業評量指標。

1. 在部屬音樂推薦系統後，參與度變高還是變低？  
可導入A/B測試，獨立觀察模型效果
2. 對模型的結果進行人工審查
3. 如果無法審查，可採用使用者調查替代

# 6-3-4 維護模型

模型不會永遠都表現良好。

一旦正式啟用，就該準備下一代模型：

1. 時刻關注實際運作資料的變動
2. 持續收集與標註資料，並且不斷改進標註過程

*Thank  
You*

@reallygreatsite