深度學習 - 第六章

Status	Book
Created time	@2024年12月30日 下午9:28

第六章

6-0 開頭

該注意的道德倫理問題：科技並不會自動中立

科技並不會自動中立，而是會隨人們的影響，展現出不同的面貌。在應用科技時，我們一定要時刻考慮，我們希望做出的成果能體現什麼樣的價值？訓練的模型是否會對真實世界造成負面影響？

舉例參考：「建構可以根據面孔照片，判斷當事人是否可信的AI辨識工具」

針對舉例的說明：首先，這個專案本身就很可疑，因為沒有和證據可以表明外表和一個人的可信度有關。再來，收集資料的時候會把標註者的主觀成見滲透進來。重點來了，假如這個模型真的上線了，對AI領域不了解的人們，會更相信AI所說的，因為他們會覺得AI是中立且可信的。這也是開頭為何說：「科技並不會自動中立」的原因。

另外供參考文章：
https://rd.coach/ai%e5%80%ab%e7%90%86%e4%b9%8b%e9%81%93%ef%bc%9a%e4%ba%ba%e5%b7%a5%e6

文章內第一大段中的第三段提到：首先，AI在決策過程中可能會受到偏見的影響。例如，在訓練AI時，我們可能會使用與現實狀況不符的數據，這可能會導致AI在某些情況下做出歧視性的決定。因此，我們需要確保訓練AI的數據是公正和多樣化的，以防止偏見的產生。

6-0 前導

機器學習的通用工作流程，可以歸納為以下3個部分

1. 定義任務（Define the task）：了解客戶需求背後的問題領域、業務邏輯、收集資料、了解資料內容，且選擇衡量任務成功與否的標準。
2. 開發模型（Develop a model）：準備好適合的資料，並選擇評估模型的機制以及要打敗的基準線（baseline）。接著訓練出具備一定普適化能力以及能過度配適的第一個模型，接著再進行常規化（regularize）和模型調整，重複步驟到可以展現最佳普適化能力。
3. 部屬模型 （Deploy a model）：向客戶展示開發成果，並將模型部屬到網路伺服器、行動app、網頁、嵌入式裝置。同時監控模型的真正表現，並建構下一代模型所需的資料。

6-1 定義任務

6-1-1 定義問題範圍

通常需要和相關人員進行多次詳細的討論。在討論時，以下問題應該擺第一順位思考。

（三角形點開可以查看更多）

▼ 輸入資料是甚麼？我們想預測甚麼？

必須要先有訓練資料，才能有下一步。
舉例：我們要分類電影評論的情緒，必須要先有電影評論以及情緒標註。

▼ 我們面對的是甚麼類型的機器學習任務？

是二元分類任務？多類別分類任務？純量回歸任務？向量回歸任務？多類別、多標籤的分類任務？影像分割任務？排序任務？資料分群任務？生成任務？強化式學習任務？...等等等。
對於某些任務來說，傳統的統計分析方法會比機器學習來的更用效率。

▼ 現有的解決方案長甚麼樣子？

客戶手上應該會有一些人工打造的演算法或流程來解決當前的問題。所以我們應該先明白既存的系統為何，以及如何運作。

▼ 是否需要考量特定的限制條件？

是否有些使用情況是需要低延遲的環境？是的話就應該在嵌入式裝置上運行，而非在遠端的伺服器上。

完成這些研究之後，應該能掌握輸入（inputs）和目標值（targets），以及當前問題對應到的機器學習種類為何。接下來會有以下兩個假設，來進行後面的步驟。

- 可以透過輸入來預測出目標值
- 現有的資料足夠用來學習輸入和目標值之間的關係

6-1-2 建立資料集

當我們明白任務的本質，也搞清楚輸入與目標值後，就可以來收集資料了。在大部分的機器學習中，這是最費力、耗時、成本最高的部分。以下舉幾個例子。

▼ 照片搜尋引擎專案

先挑選用來分類的一組標籤，然後手工標註這些照片

▼ 偵測垃圾內容專案

取得未經過濾的公開貼文，將內容標註為「垃圾內容」、「惡意內容」、「正常內容」...等等。

▼ 音樂推薦引擎專案

把使用者給的「讚」當作資料，以及歷年的點擊率。

▼ 工廠異常餅乾偵測專案

在工廠生產線安裝相機並收集影像，接著對這些影像進行手工標註是否為異常的餅乾。

！！投資在建立良好的資料集是值得的。如果有多出時間預算來處理專案的話，把這些時間花在收集更多資料，一定會比不斷修正、優化模型來的更有效率。

▼ 資料標註工具的投資

資料標註流程決定了目標值的品質，也會影響到模型的品質。標註資料前，要先思考以下選項。

- 我們需要自己標註資料嗎？
- 我們需要使用外包平台來收集標籤嗎（EX：亞馬遜的Mechanical Turk）？
- 我們需要使用專業資料標註公司的服務嗎？

將標註任務外包或許可以節省成本與時間，但同時也很難控制標註品質。使用外包平台或許可以處理大量資料、減少花費，但得到的標註結果可能會是很雜的。

所以我們要找出最佳選項，可以考慮以下面向：

- 資料標註者要是特定領域的專家？還是誰都可以做？
- 如果需要專家，那有辦法訓練人來做嗎？如果沒辦法，要怎麼找到相關領域的專家？
- 我們本身了解專家怎麼進行標註嗎？如果不了解就會有黑箱的狀況產生，而且沒辦法手動進行特徵工程。雖然這個問題不是非常嚴重，但一定會造成某些限制。

如果我們決定要自己標註，那需要用甚麼軟體來記錄標註結果？我們有可能需要自己開發相關工具。具生產力的資料標註軟體可以幫我們節省大量時間，所以在專案前對相關工具的投資是非常值得的。

▼ 留意不具代表性的資料

機器學習模型只能處理與 '' 曾經見過 '' 的資料相似的輸入資料。
因此
訓練資料一定要足以代表實際運作的資料（production data），這是所有資料收集工作的基礎

另外需要留意的現象是概念飄移（concept drift）。
概念漂移的根源來自實際資料的特性不斷變動，導致模型準確度逐漸下降。
舉個例子：

- 2013年的音樂推薦清單，放到今天就會不太具參考性了。
- 詐騙手法每年甚至每天都在更新。

想要減緩概念漂移的問題，就必須持續收集資料、進行標註，並重新訓練模型。

要注意！！機器學習只能用來記憶訓練資料中的態樣（pattern），因此只能辨識曾經看過的東西。「利用過去的資料來訓練模型，然後用來預測未來」→ 其實只是假設未來的運作模式會跟過去一樣，並非真實的預測未來。

▼ 抽樣偏差（sampling bias）

抽樣偏差的根源 ：資料收集的方式與某些要預測的事物產生關聯，進而導致資料內容有所偏差。

最著名的例子是1948年美國總統選舉，因為某論壇只相信電話民調的結果，所以預測錯誤。因為在1948年，不是所有人都擁有電話，以至於無法代表整個投票群體，導致了預測有抽樣偏差。

6-1-3 理解資料

把資料集當成黑箱處理，是不好的做法。在開始訓練模型前，我們應該先探索及視覺化資料，對資料有整體的概念後再思考他們如何協助預測，這樣有助於進行特徵工程並找出潛在問題。

▼ 如果資料中包含影像或自然語言

直接抽樣本（以及對應的標籤）出來查看。

▼ 如果資料中包含數值特徵

將這些特徵值繪製出直方圖，大概看一下圖片分布，有概括的了解。

▼ 如果資料中包含位置訊息

直接畫在地圖上，也許會出現較為清晰的態樣。

▼ 某些樣本是否缺少某些特徵值（即存在缺失值）？

6-2 開發模型會講到如何處理缺失值。

▼ 如果要處理分類任務

計算每個類別的樣本數，盡量使每個類別的樣本數一致，否則會有樣本不平衡的問題。

▼ 檢查是否存在目標值洩漏（target leaking）

目標值洩漏：訓練資料中的特徵提供了目標值的資訊，但這些資訊在實際應用場景中無法取得。這樣會讓模型「提前知道」測試集或未來資訊。

是否得肺炎
False
True
True

年齡	體重	性別	是否服用抗生素
65	50	男	False
72	65	女	True
58	50	男	True

從以上這個資料來舉例，就是肺炎和抗生素成一個正相關，就會有目標值洩漏的問題。

詳細可以參考這篇：https://blog.csdn.net/l695290718/article/details/100117712

6-1-4 選擇測量成效的方法

要在專案上取得成功，就必須先定義何謂「成功」？是準確度嗎？還是精準度？或是故障召回率？客戶回流率？成功的評量指標會引導專案中所有的技術選擇。

舉個例子：

- 平衡的分類問題
  準確度和ROC（曲線下面積）是常用的評量指標
- 類別不平衡問題
  精準度和召回率是常用的評量指標
- 排名問題、多標籤分類問題
  加權形式的準確度是常用的評量指標

很多時候，我們也必須自行定義指標來評量專案的成功與否。

Kaggle上有資料科學競賽，可以在裡面的案例查看各種領域的問題和評量指標做為參考。

https://www.kaggle.com/

6-2 開發模型

在機器學習的步驟內，開發模型只是流程中的一步，而且還不是最難的部分。最難的部分是 6-1 定義問題以及收集、處理資料。

6-2-1 準備資料

在前面有提到，深度學習通常無法接受原始資料，資料必須先經過「預處理」才行。預處理的目的就是讓原始資料更適合模型處理。其中會用到的技巧包括：向量化（vectorization）、正規化（normalization）、處理缺失值。大部分預處理的技巧都只適合用在特定的領域。

▼ 向量化

神經網路的所有輸入和目標值必須是浮點數張量（特定狀況下可以是整數或字串張量）。無論處理什麼樣的資料（聲音、影像、文字...等等），都必須先將這些資料轉換成張量，這個步驟稱為**資料向量化**（data vectoriztion）。

例如：第四章 4-1-2 中有將電影評論、新聞主題兩個文字分類範例中，一開始以整數串列（list）來代表單字序列，並使用 one-hot 編碼將其轉換成 float32 的張量。 深度學習 - 第四章

▼ 數值正規化

在第二章 " MNIST 分類數字 " 的範例中，影像資料原本被編碼成 0-255的整數，用來顯示其灰階值（garyscale）。要把這些資料輸入神經網路前，必須先將其型別轉換成 float32 並除以 255，這樣才能得到介於 0~1 之間的浮點數。 深度學習 - 第二章

```
# 資料預先處理
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255
```

在第四章 " 預測房價 " 的範例中，某些特徵是很小的浮點數（如犯罪率），而某些特徵是相當大的整數值（如年齡）。把這些資料輸入神經網路之前，必須將每個特徵正規化，使標準差為 1（用標準差為量測單位），平均值為 0（分布曲線平移到以0為中心點）。 深度學習 - 第四章

```
#沿著第0軸(樣本軸)計算平均值
mean = train_data.mean(axis=0)
train_data -= mean

#沿著第0軸(樣本軸)計算標準差
std=train_data.std(axis=0)
train_data /=std

#正規化處理
test_data -=mean
test_data /=std
```

一般來說，將差距過大的數值或異質資料（例：某個特徵值於 0~1，另一個特徵值於 100~200）輸入神經網路並不好。兩個特徵的範圍差距太大，不利神經網路收斂（會觸發很大的梯度更新）。為了保持神經網路的學習品質，我們的資料應該要有以下特性：

- 數值較小：大部分數值應介於 0~1 的範圍內
- 具備同質性（homogenous）：所有特徵應採用大致相同的數值範圍

除此之外，以下更嚴謹的正規化規則，常見的同時也很有用，但並非絕對必要。

- 單獨正規化每個特徵，使平均值為 0
- 單獨正規化每個特徵，使標準差為 1

```
#正規化處理（其實就是上面第四章的預測房價正規化範例）
x -= x.mean(axis = 0)
x /= x.std(axis = 0)
```

▼ 處理缺失值

有時候，資料中會有缺失值（missimg values）。例如在第四章的房價預測範例中，首個特徵（資料中索引為 0 的直行）是人均犯罪率。不是所有樣本都有這個特徵怎麼辦？這樣就會導致缺失值的產生。

其實我們可以直接忽略這個特徵，但實際上有更好的選項！！

▼ 如果這個特徵是分類（categorical）特徵

可以為該特徵新增一個分類值，用來代表缺失值。模型會自動學習如何將這個分類值對應到目標值。

▼ 如果這個特徵是數值（numerical）特徵

要避免隨意以一個數字（EX：0）來代表缺失值，因為這會使特徵形成潛在空間（latent space）中出現不連續性（discontinuity），導致模型會很難具有好的普適化能力。

正確的做法應該是：使用該特徵的平均值（average）或中位數（median）當作缺失值的值。又或是訓練另一個可以根據其他特徵值來預測缺失值的模型。

注意！！如果目前訓練集中沒有缺失值的狀況，但知道測試集（或未來會遇到的資料）中會有缺失值的產生，我們應該人工增加一些有缺失值的訓練樣本，才能確保神經網路能夠學會正確處理缺失值。

做法就是，複製幾個訓練樣本，並刪除未來也許會有缺失值的特徵值。

6-2-2 選擇驗證機制

第五章提過，模型的最終目標就是要實現普適化，在開發模型的過程中，每個決定都是由驗證評量指標來衡量普適化的表現。而選擇驗證評量指標的目的就是：希望未來的實際運作環境中也能有很好的表現。

在第五章中，有介紹了3種常用的驗證機制：深度學習 - 第五章

- 拆分驗證集：擁有大量資料時，這個方法最簡單
- K 折交叉驗證：樣本數不多時，是合適的方法
- 多次迭代的 K 折驗證：資料很少時，可以使用的方法

大部分的狀況下，第一種機制就夠用了。除此之外還要時刻注意驗證集的代表性，而且要注意排除訓練集與驗證集中重複出現的樣本。

6-2-3 超越基準線

當我們開始處理模型時，第一個目標就是要取得統計能力（statistical power），就如第五章所說：開發一個能夠打敗基準線的小模型。

在這個階段，我們要專注於 3 件事情：

- 特徵工程：過濾不含有用資訊的特徵（挑選特徵）。對於要處理的問題，找出可能有用的特徵。
- 選擇合適的既有架構：密集連接網路？卷積網路？循環神經網路？還是 Transformer 模型？
- 選擇足夠好的訓練配置：選擇甚麼損失函數？批次量和學習率要多大？

下表是常見的問題類型與對應的激活函數、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

對於多數問題，都有現成的範本可以供參。我們都必須花點時間研究他人既有的成果，了解有哪些特徵工程的技巧及模型架構，再嘗試應用到自己的專案上。

接著我們再帶著先前提到的兩個假設，繼續往下一個步驟。

- 可以透過輸入來預測出目標值
- 現有的資料足夠用來學習輸入和目標值之間的關係

6-2-4 擴大規模：開發一個會過度配適的模型

當我們獲得了具有統計能力（表現超過基準線）的模型，接下來的問題就是：我們的模型是否足夠強大？他是否有足夠的神經層和參數來正確擬合手上的問題？

另外要注意！！機器學習是在優化和普適化之間做取捨。理想的模型是位於低度適配與過度配適的交界處、模型太大與太小之間。為了找到這個交界點，我們必須先超越它，這也是本段的命名緣由。

為了弄清楚我們需要多大的模型，必須先開發一個會過度配適的模型。
依照第五章的步驟：

1. 添加更多的神經層
2. 讓每一神經層更寬
3. 訓練更多週期（epoch）

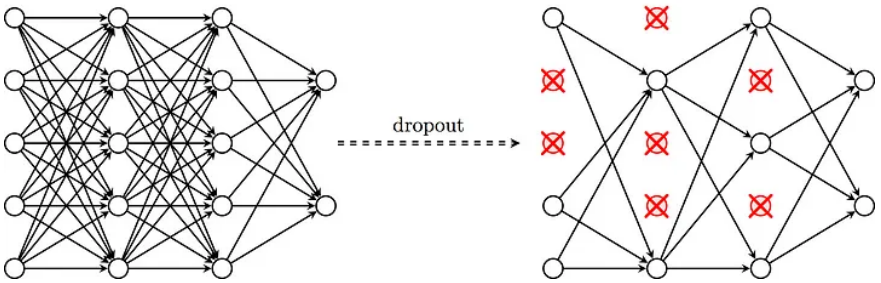
透過以上步驟與持續監控訓練損失和驗證損失與監看評量指標。如果發現驗證資料的表現開始下降的時候，就是發生過度配適了。

6-2-5 將模型常規化並調整超參數

當模型的表現超越基準線，且有過度配適的能力後，下一個目標就是最大化普適化能力。

這一步會佔很多時間，我們會反覆的：修改 → 訓練 → 使用驗證資料來評估 → 再次修改。值到模型表現不再進步為止。
以下幾點是我們該嘗試的做法：

- 嘗試不同的架構（添加或刪除神經層）
- 使用丟棄法（dropout）（一種過度配適的解決方法）
    - 在訓練時每一次的迭代 （epoch）皆以一定的機率丟棄隱藏層神經元，而被丟棄的神經元不會傳遞訊息。可參考連結：https://medium.com/%E6%89%8B%E5%AF%AB%E7%AD%86%E8%A8%98/%E4%BD%BF%E7%94%A8-tensorflow-%E4%BA%86%E8%A7%A3-dropout-bf64a6785431



- 如果模型不大，可以試試 L1、L2 常規化（兩者也可以同時使用）
    - L1正規化：把模型內所有參數都取絕對值
    - L2正規化：把模型內所有參數都取平方求和
    - 可參考連結：https://dysonma.github.io/2021/01/27/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-%E6%AD%A3%E8%A6%8F%E5%8C%96-Regularization/
- 重複資料篩選（data curation）、特徵工程
    - 收集並標註更多資料
    - 找出更好的新特徵
    - 刪除無關緊要的特徵

以上這些都可以透過「自動化的超參數調整軟體（EX：KerasTuner）」來進行，會在 13 章介紹。

注意！！如果經過系統化的 " 多次 " 迭代使用驗證集的回饋資訊來調整模型時，會造成過度配適於驗證資料，進而導致驗證的結果不可信。

經過以上步驟，我們得到令人滿意的模型配置後，就可以重新用訓練集和驗證集來訓練最終的成品模型，並用測試集做最後一次評估。

- 如果測試集的表現差於驗證資料的表現 → 代表驗證過程有問題或過度配適驗證資料

那該如何解決呢？可以嘗試切換不同的驗證機制。

6-3 部署模型

6-3-1 向客戶說明成果，並建立合理的期待

外行人對於 AI 系統的期待通常過於理想化，因為機器只能 " 逼近 " 人類的表現，所以我們需要向客戶展示甚麼狀況下會 " 失效 " 。

另外，要量化 AI 系統的成效，不要只是說「這個模型準確度為98%」，這樣過於抽象，而且大部分會直接進位到100%。在說明上應該要更明確：「每天平均會發現 300 起疑似詐騙的案件，每天平均漏掉 14 起詐騙案件。平均來說，每天會正確捕捉到 266 起詐騙案件」。還可以搭配偽陰性率和偽陽性做說明。

6-3-2 交付推論的模型

首先要先將模型會匯出至 python 以外的環境，因為運用模型的環境未必支援 python，或是有些應用程式未必以 python 運行，使用 python 交付模型會增加運營時的負擔。

▼ 雲端遠端部署

延遲時間的要求不高／運用模型的環境可以穩定的連上網路／用於進行推論的輸入資料不敏感（不具隱私性） → **以 REST API 部屬模型**

將模型轉換為產品的常見方式，是在伺服器或雲端虛擬機上安裝Tensorflow，然後將應用程式透過網路以 REST API 來呼叫模型並取得預測結果。

我們可以利用 Flask（或其他 python 網頁開發函式庫）來自己打造伺服應用網站，或是利用 Tensorflow 自身的函式庫（Tensorflow Serving）將模型輸出成可以直些運行的 API 網站程式。

▼ 裝置上部署

對延遲時間的敏感較高／網路連接不穩定／模型設計的大小相當小／輸入資料十分敏感（隱私要求高） → **使用 Tensorflow Lite**

Tensorflow Lite 框架可以在 Android、iOS、ARM-64 電腦、Raspberry Pi（樹梅派）、為控制器上運行。它包含了一個轉換器，可以將 Keras 模型轉換成 Tensorflow Lite 的格式。

▼ 瀏覽器上部屬

想把運算成本轉移給使用者（利用電腦GPU進行運算）／輸入資料必須留在使用者的裝置／延遲時間的敏感較高／下載模型後可以離線操作模型 → **使用 Tensorflow.js**

Tensorflow.js 是 JavaScript 的函式庫（以前叫WebKKeras），Tensorflow.js 提供了幾乎所有的 Keras API 的功能，也包含許多底層的 Tensorflow API。可以輕易將 Keras 匯入 Tensorflow.js，將其作為瀏覽器或桌面程式中的 JavaScript API 來進行呼叫。

優化推論模型：當部屬在記憶體與功率有限的環境（例：手機、嵌入式裝置），或是有低延遲的需求時，優化模型就顯得非常重要。在將模型匯入到 Tensorflow.js、Tensorflow Lite 之前，我們可以採用以下兩種熱門的優化技巧：

- 權重剪枝（Weigth pruning）：並非每個模型參數都對預測結果有相同的貢獻，所以可以先減少模型中的參數數量，只留取影響最多的一部分。這樣可以犧牲一點點的準確度，降低模型的運算成本。可以自行決定剪枝的比例，在模型大小與準確度中找到平衡。
- 權重量化（Weigth quantization）：深度學習模型的權重值式單精度浮點數（float32）。但可以將模型權重量化成 8 位元整數（int8），這樣可以將模型規模縮小至原本的四分之一，準確度也維持在原先的水準。

6-3-3 監控模型的運作狀況

部署好模型後，還是要持續監控模型的行為、掌握模型在新資料上的表現。

建議人工審查，如果不能人工審查，可以採取使用者調查等等替代方式，評估模型的運作狀況。

6-3-4 維護模型

最後要注意的是，先前提過的「概念飄移」，隨著時間的推進，一定會發生，導致模型的表現越來越差。所以！！一旦正式啟用模型，就應該準備訓練下一代模型！

- 關注新資料的變動，是否出現新特徵？
- 持續收集與標註資料，不斷改進標註過程