Wistron AI 網站

人工智慧問題集

（觀念篇）

版次：第 1 版

編輯群：

王光宇 KY Wang
鄭志偉 CW Cheng
李旭明 Jeff Lee
申再生 Hunter Shen
盧廷傑 Jack TC Lu
張聰耀 Benjamin Chang
梁維國 David Liang
陳慶明 Casper Chen

修訂紀錄

版本	修訂日期	修訂原因	修訂內容
第 1 版	2018/10/31	第 1 版定稿	增訂全文

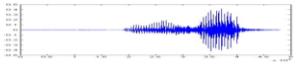
目錄

觀念篇（一）機器學習基礎觀念

1. 機器學習的機器是怎麼從資料中「學」到東西的？

人學習是爲了習得一種技能，比如學習辨認男生或女生，而我們可以從觀察中累積經驗而學會辨認男生或女生，這就是人學習的過程，觀察 -> 累積經驗、學習 -> 習得技能；而機器怎麼學習呢？其實有點相似，機器爲了學習一種技能，比如一樣是學習辨認男生或女生，電腦可以從觀察資料及計算累積模型而學會辨認男生或女生，這就是機器學習的過程，資料 -> 計算、學習出模型 -> 習得技能。

「機器學習技術，就是讓機器可以自我學習的技術。」機器學習就是讓機器根據一些訓練資料，自動找出有用的函數（function）。例如，如果將機器學習技術運用在語音辨識系統，就是要機器根據一堆聲音訊號和其對應的文字，找出如下的「語音辨識函數」：輸入一段聲音訊號，輸出就是該聲音訊號所對應的文字。

$f($  $)=$ “你好”

如果機器學習技術應用在影像辨識系統，那就是要機器根據一堆圖片和圖片中物件名稱的標注，找出「影像辨識函數」：輸入一張圖片，輸出是圖片中的物件名稱。

$f($  $)=$ “貓”

如果要機器下圍棋，就是讓機器根據一堆棋譜找出「下圍棋的函數」：輸入是棋盤上所有黑子和白子的位置，輸出是下一步應該落子的位置。

$f($  $)=$ “5之5”

以上要找的函數，共通點是它們都複雜到人類絕對沒有能力寫出它們的數學式，只有靠機器才有辦法找出來。那麼，機器要如何根據訓練資料找到函數呢？

一般機器學習方法要經過三個步驟：一、人類提供給機器一個由函數構成的集合（簡稱函數集）；二、人類根據訓練資料定義函數的優劣；三、機器自動從函數集內找出最佳的函數。

在機器學習上，技能就是透過計算所搜集到的資料來提升一些可量測的性能，比如預測得更準確，實例上像是我們可以搜集股票的交易資料，然後透過機器學習的計算及預測後，是否可以得到更多的投資報酬。如果可以增加預測的準確度，那麼我們就可以說電腦透過機器學

習得到了預測股票買賣的技能了。

假設老闆要你寫一個可以辨識樹的圖片的程式，你會怎麼寫呢？你可能寫一個程式檢查圖片中有沒有綠綠的或是有沒有像葉子的形狀的部份，然後寫了幾百條規則來完成辨識樹的圖片的功能，現在功能上線了，好死不死來了一張樹的圖片上面剛好都沒有葉子，你寫的幾百條規則都沒用了，辨識樹的圖片的功能只能以失敗收場。機器學習可以解決這樣的問題，透過觀察資料的方式來讓電腦自己辨識樹的圖片，可能會比寫幾百條判斷規則更有效。這有點像是教電腦學會釣魚（透過觀察資料學習），而不是直接給電腦魚吃（直接寫規則給電腦）。

我們在用銀行核發信用卡的例子來描述機器學習，我們可以把信用卡申請者的資料想成是 x，而 y 是銀行是否核發信用卡。所以這就是一個函數，它有一個潛在規則，可以讓 x 對應到 y，機器學習就是要算出這個 f 函式是什麼。現在我們有大量的信用卡對申請者核發信用卡的資料，就是 D，我們可以從資料觀察中得到一些假設，然後讓電腦去學習這些假設是對的還是錯的，慢慢習得技能，最後電腦可能會算出一個 g 函數，雖然不是完全跟 f 一樣，但跟 f 很像，所以能夠做出還蠻精確的預測。

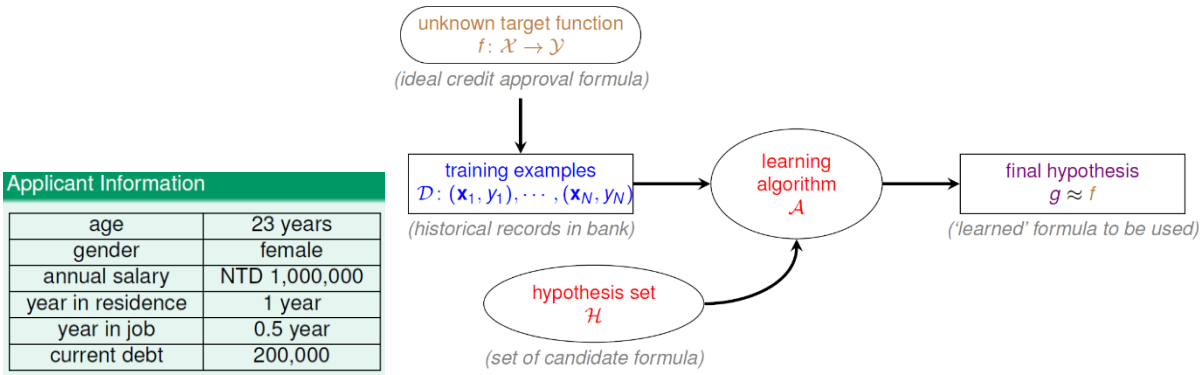


圖 1-1 以過往申請者核發信用卡資料學習是否核發信用卡的函數

所以機器學習銀行是否核發信用卡的流程就像這樣，我們想要找到目標函數（target function），可以完整預測銀行對申請者是否要核發信用卡才會賺錢，這時我們會餵給電腦大量的資料，然後透過學習演算法找出重要的特徵值，這些重要的特徵值就可以組成函數 g，雖然跟 f 不一樣，但很接近。

從上面的學習流程，我們可以知道最後電腦會學習出 g 可以辨認資料中較重要的特徵值，這些特徵值可能是我們一開始觀察資料所整理出來的假設，所以我們餵資料給電腦做學習演算法做計算時，也會餵一些假設進去，以銀行核發信用卡的例子就是這個申請者年薪是否有 80 萬、負債是否有 10 萬、工作是否小於兩年等等假設，這些假設就是 H，學習演算法再去計算實際資料與假設是否吻合，這個演算法就是 A，最後演算法會挑出最好的假設集合是哪些。 H 與 A 我們就稱為是機器學習 Model

編者：鄭志偉，AI Office，20181015

i. **資料來源：網路課程_台灣大學林軒田教授機器學習基石課程第一講**

ii. **資料來源：數理人文雜誌第 10 期 什麼是深度學習？**

2. 機器學習可以處理的工作可以分成哪些類型?

下面是我們在機器學習的機器是怎麼從資料中「學」到東西的？小節所講述的機器學習流程。機器學習根據所學習情境與狀態的不同，可以分為有監督學習，無監督學習，半監督學習和增強式學習（強化學習）等幾種類型。而依照輸出的狀態可以分爲二元分類、多元分類、回歸（數值預測）、分群、結構化學習等。

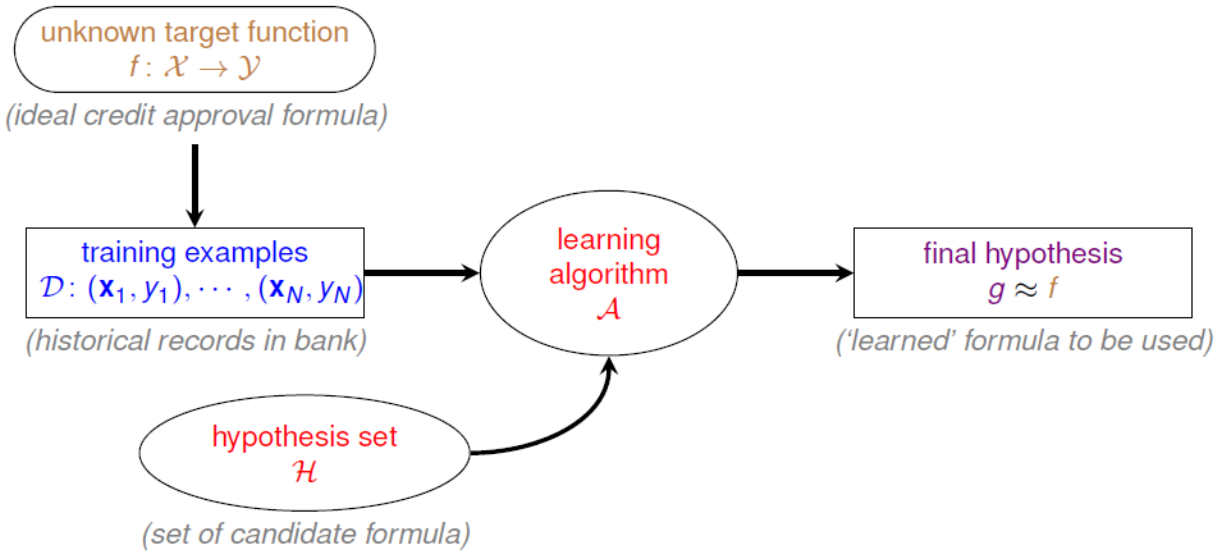


圖 2-1 機器學習的過程

- **二元分類**：從輸出 y 的角度看機器學習，y 只有兩個答案選一個，就叫二元分類（Binary Classification）。例如：
  - 銀行從過往顧客的資料與有沒有核發信用卡來讓機器學習一個一個技能，當我們有新的顧客時『要』、『不要』發信用卡給他的新申請信用卡的顧客。
  - 自動判定電子郵件『是』垃圾信件、『不是』垃圾郵件。
  - 判定病人『有』生病、『沒有』生病，有個廣告『會』賺錢、『不會』賺錢...等。
- **多元分類**：從輸出 y 的角度看機器學習，y 有多個答案選一個，就叫多元分類（Multiclass Classification），例如：
  - 自動販賣機辨識投入的多種錢幣的問題就是一個多元分類（1 元、5 元、10 元、50 元）的問題，所以我們可以將分類問題推廣到分成 K 類，這樣二元分類就是一個 K=2 的分類問題。
  - 手寫郵遞區號讓電腦辨識 0~9 中的哪一個數字。

- 有水果的圖片請電腦辨識哪一種水果。或視覺辨識物體屬於哪一類如圖 2-2。
- 電子郵件區分為垃圾郵件、重要郵件、社交郵件、促銷郵件等。
- 輸入病人的狀況，電腦自動判定病人得了哪一類疾病或哪一類的癌症或沒有癌症。

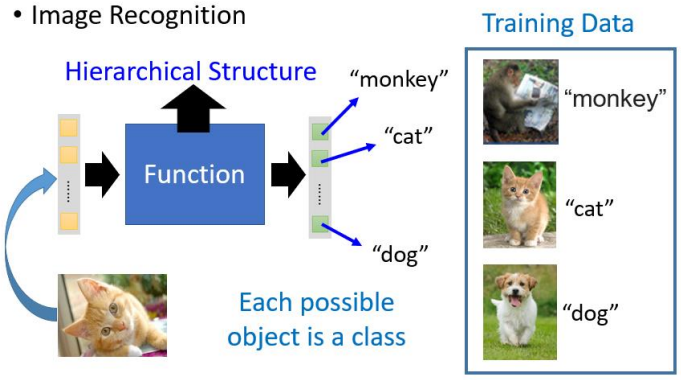


圖 2-2 單一目標之圖像識別

- **回歸：**從輸出 y 的角度看機器學習，y 為一個實數，就叫回歸（Regression），例如要預估病人再過幾天病會好，或是輸入輸出是一個實數的範圍[lower, upper]，這就需要用到 Bounded Regression，這也會用到許多統計學相關的工具。
- **結構化學習：**從輸出 y 的角度看機器學習，y 為一個結構序列，就叫 Structured Learning，比如一個句子的詞性分析，會需要考慮到句子中的前後文，而句子的組合可能有無限多種，因此不能單純用 Multiclass Classification 來做到，這就需要用到 Structured Learning 相關的機器學習方法。圖 2-3、圖 2-4 是結構化學習的一些範例。

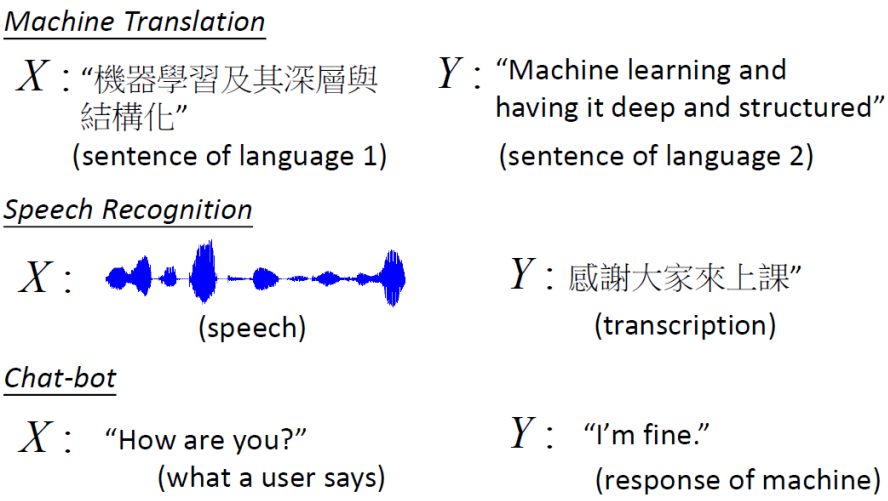


圖 2-3 結構化學習（語言翻譯、語音辨識、聊天軟體機器人）

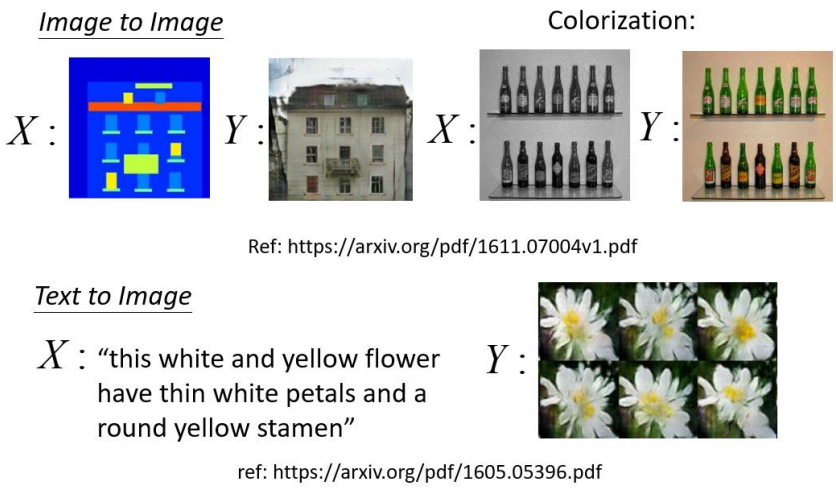


圖 2-4 結構化學習（自動圖片轉換、自動圖片上色、輸入文字產生圖片）

- **監督式學習：**從輸入的資料 Y_n 的角度看機器學習，如果每個 X_n 都有明確對應的 Y_n，這就叫監督式學習（Supervised Learning），比如在訓練投飲機辨識錢幣的時候，我們很完整個告訴他什麼大小、什麼重量就是什麼幣值的錢幣，這樣就是一種監督式學習方法。

- **非監督式學習：**從輸入的資料 Y_n 的角度看機器學習，如果每個 X_n 都沒有標示 Y_n，這就叫非監督式學習（Unsupervised Learning）。比如在訓練自動販賣機辨識錢幣的時候，我們只告訴自動販賣錢幣的大小及重量，但不告訴他什麼大小及重量個錢幣是哪個幣值的錢幣，讓機器自己去觀察特徵將這些錢幣分成一群一群，這又叫做分群（Clustering），這就是一種非監督式學習方法。其他的應用包含網路一堆文章不告訴電腦怎麼分類，讓電腦自動分成幾類再做不同的處理、一堆顧客自動分成幾群做出不同的促銷活動、分群後產生密度函數偵測危險事故發生的可能性、分群後偵測異常的訊號或入侵者偵測問題。

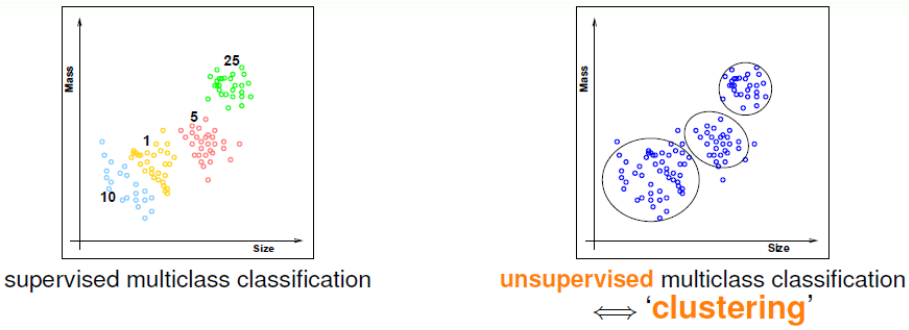


圖 2-5 監督式學習的分類與非監督式學習的分群

- **半監督式學習：**從輸入的資料 Y_n 的角度看機器學習，如果 X_n 只有部分有標示 Y_n，這就叫半監督式學習（Semi-supervised Learning），有些資料較難取得的狀況下，或是取得資料成本很高，我們會使用到半監督式學習，來增強學習的效果，比如在預測藥物是否對病人有效時，由於做人體實驗成本高且可能要等一段時間來看藥效，這樣的情況

下標示藥物有效或沒效的成本很高，所以就可能需要用到半監督式學習。

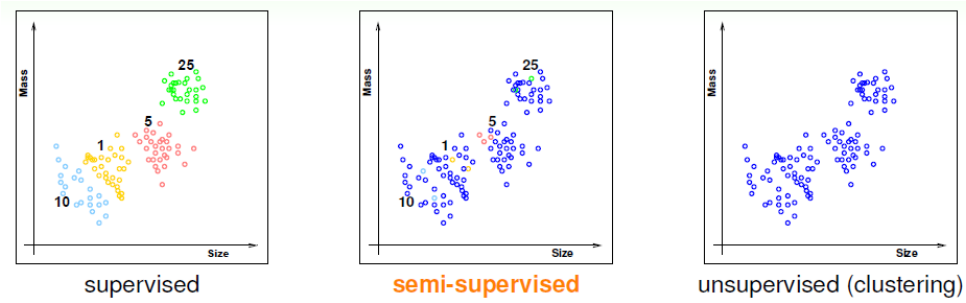


圖 2-6 監督學習、半監督學習、非監督學習在學習的資料差別

- **增強式學習（強化學習）**：從輸入的資料 Y_n 的角度看機器學習，如果 Y_n 是很難確知描述的，只能在機器作出反應時使用處罰及獎勵的方式讓機器知道對或錯，這就叫增強式學習（Reinforcement Learning）。這樣的機器學習方式，比較像自然界生物的學習方式，就像你要教一隻狗坐下，你很難直接告訴他怎麼做，而是用獎勵或處罰的方式讓狗狗漸漸知道坐下是什麼。增強式學習也就是這樣的機器學習方法，透過一次一次經驗的累積讓機器能夠學習到一個技能。比如像是教機器學習下棋，我們也可以透過勝負讓機器漸漸學習到如何下棋會下得更好。

編者：鄭志偉，AI Office，20181015

i. 資料來源：網路課程_台灣大學林軒田教授機器學習基石課程第三講

ii. 資料來源：網路課程_台灣大學李宏毅教授機器學習課程

3. 基於機器學習開發和部署模型的工作流程是什麼?

構建一個典型的機器學習專案，一般分成以下步驟：收集原始數據、合併數據來源、清洗數據、特徵工程、構建模型、超參數調優、驗證模型和設備端部署模型。

整個過程中，模型構建最能體現創造力，而最耗時的，要數特徵工程和超參數調優。於是，有時候會因爲趕時間，過早將模型從實驗階段轉移到生產階段，導致它們發揮不出最佳效果；也有時候，會因爲花了太多時間調優導致部署延遲。

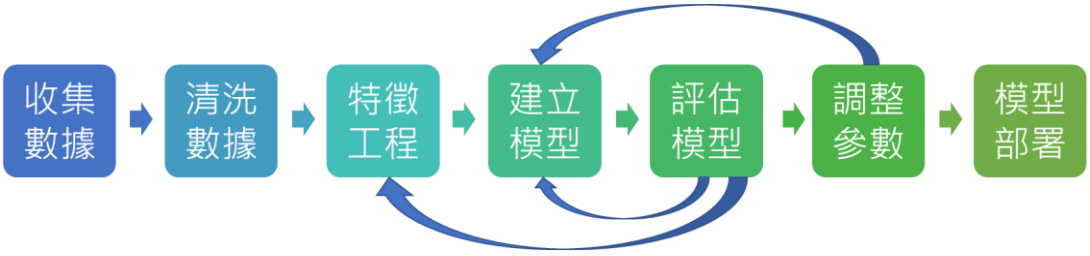


圖 3-1 常見之機器學習開發與部署流程

編者：鄭志偉，AI Office，20181015

i. 資料來源：自動機器學習工具全景圖：精選 22 種框架，解放煉丹師

4. 訓練 AI 模型所需的資料（數據）可能的問題為何?

要讓機器學習（machine learning）廣泛應用且可創造獲利，最大的障礙就是資料品質太差。「垃圾進，垃圾出」（garbage-in，garbage-out）這個犀利的觀察，多年來一直困擾著分析和決策領域，但這對機器學習來說，更具有特殊的警惕意義。機器學習對品質的要求特別高，而品質差的資料有兩次出現機會，第一次是用來訓練預測模型的歷史資料，第二次則是那個模型在未來要做決策時所使用的新資料。

爲了用恰當的方式來訓練預測模型，歷史資料必須符合特別廣泛且高品質的標準。首先，資料必須是正確的：它必須正確、恰當地標示、已去除重複內容等。但你也必須有對的資料，也就是要有許多無偏誤的資料，而且你打算開發的預測模型所需要輸入的所有範圍的資料，都必須包括在內。大多數的資料品質管控工作，只著重在上述兩個標準的其中之一，但對機器學習來說，你必須同時採用這兩個標準。

只是在今日，大部分資料都不符合基本的「資料是正確的」標準。原因有很多，像是產生資料的人不瞭解該怎麼做、沒有好好校準的衡量工具、過度複雜的流程、人爲過失等。爲彌補這些缺失，資料科學家在訓練預測模型之前，會先清理這些資料。清理工作很耗時間（約佔資料科學家 80%的時間），而且單調乏味，這是他們最常抱怨的問題。即使做了這些努力，資料清理工作仍無法偵測並修正所有的錯誤，而且目前還沒有辦法瞭解這對預測模型有什麼影響。此外，資料不見得都能符合「要有對的資料」的標準，例如有報告指出人臉辨識與刑事司法體系裏會出現資料偏誤。

日益困難的問題，不只需要更多資料，還需要更多元、更完整的資料，因而產生更多資料品質的問題。舉例來說，IBM 致力把機器學習應用到癌症治療上，例如運用華生（Watson）系統，而手寫的筆記和小範圍內通行的縮寫用法，讓 IBM 這項努力變得更複雜。

在執行方面，資料品質問題也一樣造成麻煩。假設有一家公司打算透過機器學習計畫來提高生產力。開發出預測模型的資料科學團隊，雖然之前可能已經很認真地清理過訓練用的資料，但之後實際運作時，預測模型仍可能因爲品質差的資料而影響到成效。同樣還是需要人力（而且是很多人力），來找出並修改錯誤。於是破壞了原先想提高生產力的期望。另外，隨著機器學習技術普及到全公司，一個預測模型產出的結果，會輸入另一個預測模型，一輪一輪持續下去，甚至傳送到公司以外的地方。這情況的風險在於，在某一個步驟產生的微小錯誤，會一步步傳遞下去，在整個過程中造成更多錯誤，演變成更嚴重的錯誤。

再者，很多演算法都有一個基本假設，那就是數據分佈是均勻的。當我們把這些演算法直接應用於實際數據時，大多數情況下都無法取得理想的結果。因爲實際數據往往分佈得很不均勻，都會存在“長尾現象”，也就是所謂的“二八原理”。在監督機器學習任務中。當遇到

不平衡數據時，以總體分類準確率爲學習目標的傳統分類演算法會過多地關注多數類，從而使得少數類樣本的分類性能下降。

編者：鄭志偉，AI Office，20181015

i. **資料來源：哈佛商業評論 20180515，資料差，機器學習工具就無效**

ii. **資料來源：如何解決機器學習中數據不平衡問題**

5. 什麼是特徵工程 Feature Engineering?

特徵工程是將原始數據轉化爲特徵，更好表示預測模型處理的實際問題，提升對於未知數據的準確性。它是用目標問題所在的特定領域知識或者自動化的方法來生成、提取、刪減或者組合變化得到特徵。

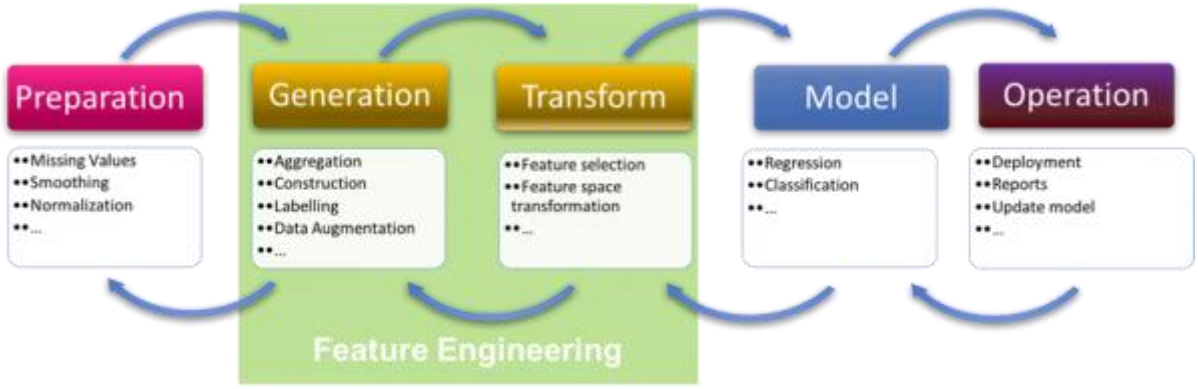


圖 5-1 預測模型建立過程在很大程度上依賴於特徵工程

機器學習中的特徵（Feature）

在機器學習和模式識別中，特徵是在觀測現象中的一種獨立、可測量的屬性。選擇信息量大的、有差別性的、獨立的特徵是模式識別、分類和回歸問題的關鍵一步。

最初的原始特徵數據集可能太大，或資訊雜亂，因此在機器學習的應用中，一個初始步驟就是選擇特徵的子集，或構建一套新的特徵集，來促進演算法的學習，提高泛化能力和可解釋性。

在表格數據中，觀測數據或實例（對應表格的一行）由不同的變量或者屬性（表格的一列）構成，這裏屬性其實就是特徵。但是與屬性一詞不同的是，特徵是對於分析和解決問題有用、有意義的屬性。

在機器視覺中，一幅圖像是一個觀測，但是特徵可能是圖中的一條線；在自然語言處理中，一個文本是一個觀測，但是其中的段落或者詞頻可能才是一種特徵；在語音識別中，一段語音是一個觀測，但是一個詞或者音素才是一種特徵。

特徵的重要性（Feature Importance）

你可以客觀的評價特徵的實用性。判別特徵的重要性是對特徵進行選擇的預先指標，特徵根據重要性被分配分數，然後根據分數不同進行排序，其中高分的特徵被選擇出來放入訓練數據集。如果與預測的事物高度相關，則這個特徵可能很重要，其中相關係數和獨立變量方法是常用的方法。

在構建模型的過程中，一些複雜的預測模型會在演算法內部進行特徵重要性的評價和選擇，如多元自適應回歸樣條法（Multivariate Adaptive Regression Splines，MARS）、隨機森林（Random Forest）、梯度提升機（Gradient Boosting Machines）。這些模型在模型準備階段會進行變量重要性的確定。

特徵提取（Feature Extraction）

一些觀測數據如果直接建模，其原始狀態的數據太多。像圖像、音頻和文本數據，如果將其看做是表格數據，那麼其中包含了數以千計的屬性。

特徵提取是自動地對原始觀測降維，使其特徵集合小到可以進行建模的過程。

對於表格式數據，可以使用主元素分析（Principal Component Analysis）、聚類等映射方法；對於圖像數據，可以進行線（line）或邊緣（edge）的提取；根據相應的領域，圖像、視頻和音頻數據可以有很多數字信號處理的方法對其進行處理。

特徵選擇（Feature Selection）

不同的特徵對模型的準確度的影響不同，有些特徵與要解決的問題不相關，有些特徵是冗餘資訊，這些特徵都應該被移除掉。

特徵選擇是自動地選擇出對於問題最重要的那些特徵子集的過程。

特徵選擇演算法可以使用評分的方法來進行排序；還有些方法通過反覆試驗來搜索出特徵子集，自動地創建並評估模型以得到客觀的、預測效果最好的特徵子集；還有一些方法，將特徵選擇作爲模型的附加功能，像逐步回歸法（Stepwise regression）就是一個在模型構建過程中自動進行特徵選擇的演算法。

特徵構建（Feature Construction）

特徵重要性和選擇是告訴使用者特徵的客觀特性，但這些工作之後，需要你人工進行特徵的構建。

特徵構建需要花費大量的時間對實際樣本數據進行處理，思考數據的結構，和如何將特徵數據輸入給預測演算法。

對於表格數據，特徵構建意味著將特徵進行混合或組合以得到新的特徵，或通過對特徵進行分解或切分來構造新的特徵；對于文本數據，特徵夠自己按意味著設計出針對特定問題的文

本指標；對於圖像數據，這意味著自動過濾，得到相關的結構。

特徵學習（Feature Learning）

特徵學習是在原始數據中自動識別和使用特徵。

現代深度學習方法在特徵學習領域有很多成功案例，比如自編碼器和受限玻爾茲曼機。它們以無監督或半監督的方式實現自動的學習抽象的特徵表示（壓縮形式），其結果用於支撐像語音識別、圖像分類、物體識別和其他領域的先進成果。抽象的特徵表達可以自動得到，但是你無法理解和利用這些學習得到的結果，只有黑盒的方式才可以使用這些特徵。你不可能輕易懂得如何創造和那些效果很好的特徵相似或相異的特徵。這個技能是很難的，但同時它也是很有魅力的，很重要的。

編者：鄭志偉，AI Office，20181015

i. **資料來源：想搞機器學習，不會特徵工程？你 TM 逗我那！**

ii. **資料來源：Removing the hunch in data science with AI-based automated feature engineering**

6. 機器學習有哪些常見的演算法?

機器學習的演算法很多。很多時候困惑人們都是，很多演算法是一類演算法，而有些演算法又是從其他演算法中延伸出來的。

根據資料型別的不同，對一個問題的建模有不同的方式。在**錯誤! 找不到參照來源。錯誤! 找不到參照來源。** 一節提到依演算法的學習方式可以分為監督式學習、非監督式學習、半監督式學習、強化學習幾種主要的學習方式。將演算法按照學習方式分類是一個不錯的想法，這樣可以讓人們在建模和演算法選擇的時候考慮能根據輸入資料來選擇最合適的演算法來獲得最好的結果。

而根據演算法的功能和形式的類似性，我們也可以把演算法分類，比如說基於樹的演算法，基於神經網路的演算法等等。當然，機器學習的範圍非常龐大，有些演算法很難明確歸類到某一類。而對於有些分類來說，同一分類的演算法可以針對不同型別的問題。這裏，我們儘量把常用的演算法按照最容易理解的方式進行分類。

迴歸演算法：迴歸演算法是試圖採用對誤差的衡量來探索變數之間的關係的一類演算法。迴歸演算法是統計機器學習的利器。在機器學習領域，人們說起迴歸，有時候是指一類問題，有時候是指一類演算法，這一點常常會使初學者有所困惑。常見的迴歸演算法包括：普通最小平方法（Ordinary Least Square）、邏輯迴歸（Logistic Regression）、逐步式迴歸（Stepwise Regression）、多元自適應迴歸樣條（Multivariate Adaptive Regression Splines, MARS）以及局部散點平滑估計（Locally Estimated Scatterplot Smoothing,

LOESS）

基於例項的演算法（instance-based learning）：基於例項的演算法常常用來對決策問題建立模型，這樣的模型常常先選取一批樣本資料，然後根據某些近似性把新資料與樣本資料進行比較。通過這種方式來尋找最佳的匹配。因此，基於例項的演算法常常也被稱爲「贏家通吃」學習或者「基於記憶的學習」。常見的演算法包括最近鄰居法（k-Nearest Neighbor，KNN）、學習式向量量化（Learning Vector Quantization，LVQ），以及自組織對映演算法（Self-Organizing Map，SOM）

正則化方法：正則化方法是其他演算法（通常是迴歸演算法）的延伸，根據演算法的複雜度對演算法進行調整。正則化方法通常對簡單模型予以獎勵而對複雜演算法予以懲罰。常見的演算法包括：脊迴歸（Ridge Regression）、最小絕對值收斂和選擇運算元、套索演算法（Least Absolute Shrinkage and Selection Operator，LASSO），以及彈性網路（Elastic Net）。

決策樹學習：決策樹演算法根據資料的屬性採用樹狀結構建立決策模型，決策樹模型常常用來解決分類和迴歸問題。常見的演算法包括：分類及迴歸樹（Classification And Regression Tree， CART）、ID3 演算法（疊代二元樹 3 代，Iterative Dichotomiser 3）、C4.5、卡方自動交互檢視法（Chi-squared Automatic Interaction Detection，CHAID）、決策樹樁（Decision Stump）、隨機森林（Random Forest）、多元自適應迴歸樣條（MARS）以及梯度提升機（Gradient Boosting Machine，GBM）

貝葉斯方法（Bayesian Methods）：貝葉斯方法演算法是基於貝葉斯定理的一類演算法，主要用來解決分類和迴歸問題。常見演算法包括：樸素貝葉斯演算法，平均單依賴估計（Averaged One-Dependence Estimators，AODE），以及 Bayesian Belief Network（BBN）。

基於核的演算法：基於核的演算法中最著名的莫過於支援向量機（SVM）了。基於核的演算法把輸入資料對映到一個高階的向量空間， 在這些高階向量空間裏，有些分類或者回歸問題能夠更容易的解決。常見的基於核的演算法包括：支援向量機（Support Vector Machine，SVM）， 徑向基函式（Radial Basis Function，RBF）， 以及線性判別分析（Linear Discriminate Analysis，LDA）等

聚類演算法：聚類，就像迴歸一樣，有時候人們描述的是一類問題，有時候描述的是一類演算法。聚類演算法通常按照中心點或者分層的方式對輸入資料進行歸並。所以的聚類演算法都試圖找到資料的內在結構，以便按照最大的共同點將資料進行歸類。常見的聚類演算法包括 k-Means 演算法以及期望最大化演算法（Expectation Maximization， EM）。

關聯規則學習：關聯規則學習通過尋找最能夠解釋資料變數之間關係的規則，來找出大量多後設資料集中有用的關聯規則。常見演算法包括 Apriori 演算法和 Eclat 演算法等。

人工神經網路：人工神經網路演算法模擬生物神經網路，是一類模式匹配演算法。通常用於解決分類和迴歸問題。人工神經網路是機器學習的一個龐大的分支，有幾百種不同的演算法。其中深度學習就是其中的一類演算法，我們會單獨討論，重要的人工神經網路演算法包括：感知器神經網路（Perceptron Neural Network）、反向傳遞（Back Propagation）、Hopfield 網路，自組織對映（Self-Organizing Map，SOM）、學習向量量化（Learning Vector Quantization，LVQ）

深度學習：深度學習演算法是對人工神經網路的發展。在計算能力變得日益廉價的今天，深度學習試圖建立大得多也複雜得多的神經網路。很多深度學習的演算法是半監督式學習演算法，用來處理存在少量未標識資料的大資料集。常見的深度學習演算法包括：受限波爾茲曼機（Restricted Boltzmann Machine，RBN）、Deep Belief Networks（DBN）、卷積網路（Convolutional Network）、堆疊式自動編碼器（Stacked Auto-encoders）。

降低維度演算法（Dimension Reduction）：像聚類演算法一樣，降低維度演算法試圖分析資料的內在結構，不過降低維度演算法是以非監督學習的方式試圖利用較少的資訊來歸納或者解釋資料。這類演算法可以用於高維資料的視覺化或者用來簡化資料以便監督式學習使用。常見的演算法包括：主成份分析（Principle Component Analysis, PCA）、偏最小二乘迴歸（Partial Least Square Regression, PLS）、Sammon 對映、多維尺度（Multi-Dimensional Scaling, MDS）、投影追蹤（Projection Pursuit）等。

整合演算法（Ensemble Learning）：整合演算法用一些相對較弱的學習模型獨立地就同樣的樣本進行訓練，然後把結果整合起來進行整體預測。整合演算法的主要難點在於究竟整合哪些獨立的較弱的學習模型以及如何把學習結果整合起來。這是一類非常強大的演算法，同時也非常流行。常見的演算法包括：Boosting、Bootstrapped Aggregation（Bagging）、AdaBoost、堆疊泛化（Stacked Generalization, Blending）、梯度提升機（Gradient Boosting Machine, GBM）、隨機森林（Random Forest）。

編者：鄭志偉，AI Office，20181015

i. **資料來源：機器學習常見演算法分類彙總**

7. 這麼多 AI 演算法，目前檯面上的主角是誰?

深度學習是機器學習的一種方式，先前擊敗棋王的 Google AlphaGo、以及奪得益智問答比賽大獎的 IBM Watson 都是深度學習的代表案例。在人工智慧領域提及深度學習與認知運算，就不能不知道「深度神經網路」（Deep Neural Network, DNN）演算法的機器學

習。DNN 是模仿生物神經系統的數學模型，能夠讓程式具自我學習功能，設計欲解決問題的數學模型。透過大量數據搜集與標記，反覆訓練數據資料調整數學模型，完成具有可信準確度的數學模型來解決辨識、預測、分類等問題。目前有很多開源深度學習框架如 TensorFlow、PyTorch、MXNet、Caffee 與相關工具，提供深度學習模型開發者一個方便且有效率的深度學習訓練環境，大幅提升 AI 應用開發。

編者：鄭志偉，AI Office，20181015

i. 資料來源：AI 人工智慧透過深度學習帶來落地應用

8. 什麼是混淆矩陣 Confusion Matrix?

混淆矩陣是用來總結一個分類器結果的矩陣。對於 k 元分類，其實它就是一個 k x k 的表格，用來記錄分類器的預測結果。對於最常見的二元分類來說，它的混淆矩陣是 2 乘 2 的，這四種結局可以畫成 2 × 2 的混淆矩陣，如 圖 8-1：

		真實值		總數
		p	n	
預測輸出	*p'*	真陽性（TP）	偽陽性（FP）	P'
	n'	偽陰性（FN）	真陰性（TN）	N'
總數		P	N	

圖 8-1 混淆矩陣

樣本總數=P + N = TP + FN + FP + TN = P'+ N'

TP = True Postive = 真陽性； FP = False Positive = 偽陽性（假陽性）

FN = False Negative = 偽陰性（假陰性）； TN = True Negative = 真陰性

舉例來說，用血壓值來檢測一個人是否有高血壓，測出的血壓值是連續的實數（從 0~200 都有可能），以收縮壓 140 / 舒張壓 90 為閾值，閾值以上便診斷為有高血壓，閾值未滿者診斷為無高血壓。二元分類模型的個案預測有四種結局：

真陽性（TP）：診斷為有，實際上也有高血壓。

偽陽性（FP）：診斷為有，實際卻沒有高血壓。

真陰性（TN）：診斷為沒有，實際上也沒有高血壓。

偽陰性（FN）：診斷為沒有，實際卻有高血壓。

混淆矩陣中的這四個數值，經常被用來定義其他一些度量。

- **準確率（Accuracy）**

  =（TP+TN）/（TP+TN+FN+TN）（預測正確的結果佔樣本總數的百分比）

  雖然準確率可以判斷總的正確率，但是在樣本不平衡的情況下，並不能作爲很好的指標來衡量結果。舉個簡單的例子，比如在一個總樣本中，正樣本佔 90%，負樣本佔 10%，樣本是嚴重不平衡的。對於這種情況，我們只需要將全部樣本預測爲正樣本即可得到 90%的高準確率，但實際上我們並沒有很用心的分類，只是隨便無腦一分而已。這就說明瞭：由於樣本不平衡的問題，導致了得到的高準確率結果含有很大的水分。即如果樣本不平衡，準確率就會失效。

- **精度率（Precision）、查準率（Precision Rate）、陽性預測值（PPV、Positive Predictive Value）**

  = TP / (TP + FP)

  針對預測結果而言的，它的含義是在所有被預測爲正的樣本中實際爲正的樣本的概率，意思就是在預測爲正樣本的結果中，我們有多少把握可以預測正確

- **真陽性率（TPR、True Positive Rate）、召回率（Recall rate）、查全率、敏感度（Sensitivity）**

  = TP / P = TP / (TP + FN)（在所有實際為陽性的樣本中，被正確地判斷為陽性之比率）

  比如拿網貸違約率爲例，相對好用戶，我們更關心壞用戶，不能錯放過任何一個壞用戶。因爲如果我們過多的將壞用戶當成好用戶，這樣後續可能發生的違約金額會遠超過好用戶償還的借貸利息金額，造成嚴重償失。召回率越高，代表實際壞用戶被預測出來的概率越高，它的含義類似：寧可錯殺一千，絕不放過一個。

- **偽陽性率（FPR、False Positive Rate）、假警報率（False Alarm Rate）、錯誤命中率**

  = FP/N = FP/ (TN + FP)（所有實際為陰性的樣本中，被錯誤地判斷為陽性之比率）

- **特異度（specificity）、真陰性率（TNR、True Negative Rate）**

  = TN / N = TN / (TN + FP) = 1-FPR

- **陰性預測值（NPV、Negative Predictive Value））= TN / (TN + FN)**

- **假發現率（FDR）= FP / (FP + TP)**

- **F1-值（F1-score）= 2*TP / (2*TP+FP+FN)**

編者：鄭志偉，AI Office，20181015

i. 資料來源：維基百科 ROC 曲線

9. 什麼是 ROC 曲線（Receiver Operating Characteristic Curve）?

ROC 空間將偽陽性率（FPR）定義為 *X* 軸，真陽性率（TPR）定義為 *Y* 軸。

給定一個二元分類模型和它的閾值，就能從所有樣本的（陽性 / 陰性）真實值和預測值計算出一個（X=FPR, Y=TPR）座標點。

從（0, 0）到（1, 1）的對角線將 ROC 空間劃分爲左上 / 右下兩個區域，在這條線的以上的點代表了一個好的分類結果（勝過隨機分類），而在這條線以下的點代表了差的分類結果（劣於隨機分類）。

完美的預測是一個在左上角的點，在 ROC 空間座標（0, 1）點，X=0 代表著沒有僞陽性，Y=1 代表著沒有僞陰性（所有的陽性都是真陽性）；也就是說，不管分類器輸出結果是陽性或陰性，都是 100%正確。一個隨機的預測會得到位於從（0, 0）到（1, 1）對角線（也叫無識別率線）上的一個點；最直觀的隨機預測的例子就是拋硬幣。

讓我們來看在實際有 100 個陽性和 100 個陰性的案例時，四種預測方法（可能是四種分類器，或是同一分類器的四種閾值設定）的結果差異，並將這 4 種結果畫在 ROC 空間裡：

A			B			C			C'		
TP=63	FP=28	91	TP=77	FP=77	154	TP=24	FP=88	112	TP=76	FP=12	88
FN=37	TN=72	109	FN=23	TN=23	46	FN=76	TN=12	88	FN=24	TN=88	112
100	100	200	100	100	200	100	100	200	100	100	200
TPR = 0.63			TPR = 0.77			TPR = 0.24			TPR = 0.76		
FPR = 0.28			FPR = 0.77			FPR = 0.88			FPR = 0.12		
ACC = 0.68			ACC = 0.50			ACC = 0.18			ACC = 0.82		

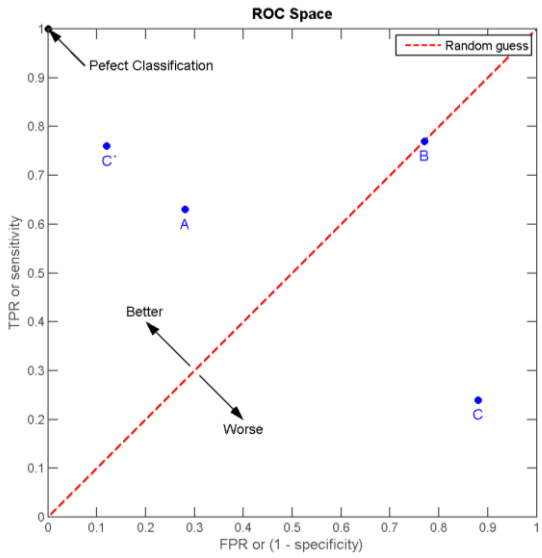


圖 9-1 不同分類器在 ROC 空間的結果

點與隨機猜測線的距離，是預測力的指標：離左上角越近的點預測（診斷）準確率越高。離右下角越近的點，預測越不準。

在 A、B、C 三者當中，最好的結果是 A 方法。B 方法的結果位於隨機猜測線（對角線）上，在例子中我們可以看到 B 的準確度（ACC，定義見前面表格）是 50%。

C 雖然預測準確度最差，甚至劣於隨機分類，也就是低於 0.5（低於對角線）。然而，當將 C 以（0.5, 0.5）為中點作一個鏡像後，C'的結果甚至要比 A 還要好。這個作鏡像的方法，簡單說，不管 C（或任何 ROC 點低於對角線的情況）預測了什麼，就做相反的結論

ROC 曲線：上述 ROC 空間裏的單點，是給定分類模型且給定閾值後得出的。但同一個二元分類模型的閾值可能設定爲高或低，每種閾值的設定會得出不同的 FPR 和 TPR。將同一模型每個閾值 的（FPR, TPR）座標都畫在 ROC 空間裏，就成爲特定模型的 ROC 曲線。

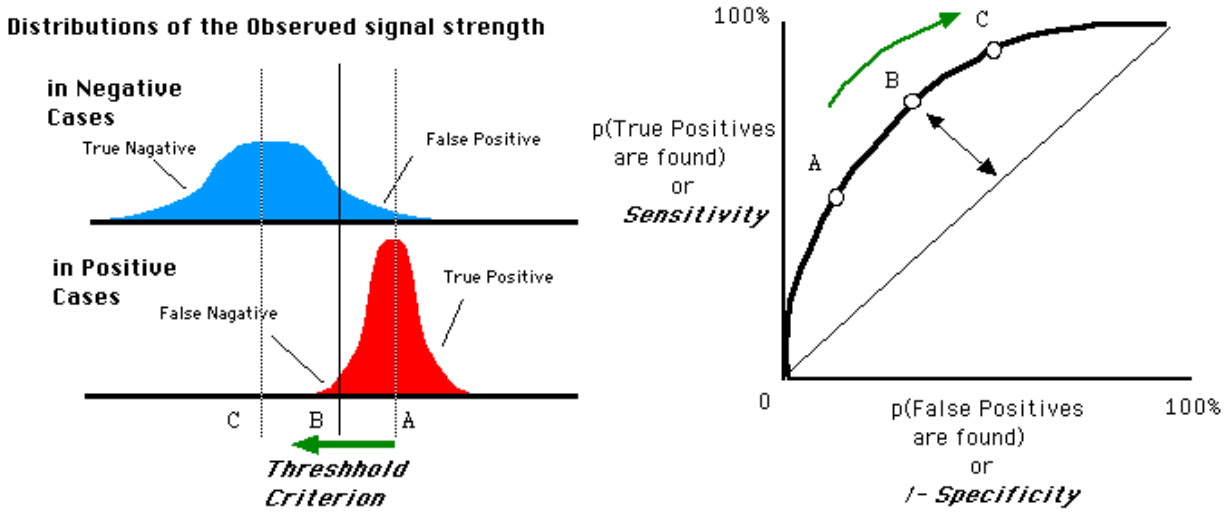


圖 9-2 隨著閾值調整，ROC 座標系裏的點如何移動

例如圖 9-2，人體的血液蛋白濃度是呈常態分佈的連續變數，病人的分佈是紅色，平均值爲 A g/dL，健康人的分佈是藍色，平均值是 C g/dL。健康檢查會測量血液樣本中的某種蛋白質濃度，達到某個值（閾值，threshold）以上診斷為有疾病徵兆。研究者可以調整閾值的高低（將左上圖的 B 垂直線往左或右移動），便會得出不同的僞陽性率與真陽性率，總之即得出不同的預測準確率。

1. 由於每個不同的分類器（診斷工具、偵測工具）有各自的測量標準和測量值的單位（標示為：「健康人 - 病人分佈圖」的橫軸），所以不同分類器的「健康人 - 病人分佈圖」都長得不一樣。

2. 比較不同分類器時，ROC 曲線的實際形狀，便視兩個實際分佈的重疊範圍而定，沒有規律可循。

3. 但在同一個分類器之內，閾值的不同設定對 ROC 曲線的影響，仍有一些規律可循：

- 當閾值設定為最高時，亦即所有樣本都被預測為陰性，沒有樣本被預測為陽性，此時在偽陽性率 FPR = FP / (FP + TN) 算式中的 FP = 0，所以 FPR = 0%。同時在真陽性率（TPR）算式中， TPR = TP / (TP + FN) 算式中的 TP = 0，所以 TPR = 0% → 當閾值設定爲最高時，必得出 ROC 座標系左下角的點（0, 0）。
- 當閾值設定為最低時，亦即所有樣本都被預測為陽性，沒有樣本被預測為陰性，此時在偽陽性率 FPR = FP / (FP + TN) 算式中的 TN = 0，所以 FPR = 100%。同時在真陽性率 TPR = TP / (TP + FN) 算式中的 FN = 0，所以 TPR=100%→當閾值設定爲最低時，必得出 ROC 座標系右上角的點（1, 1）。
- 因為 TP、FP、TN、FN 都是累積次數，TN 和 FN 隨著閾值調低而減少（或持平），TP 和 FP 隨著閾值調低而增加（或持平），所以 FPR 和 TPR 皆必隨著閾值調低而增加（或持平）。→ 隨著閾值調低，ROC 點 往右上（或右 / 或上）移動，或不動；但絕不會往左下（或左 / 或下）移動。

編者：鄭志偉，AI Office，20181015

i. **資料來源：維基百科 ROC 曲線**

10. ROC 曲線下的面積 AUC（Aear Under the Curve）所代表的意義爲何?

在比較不同的分類模型時，可以將每個模型的 ROC 曲線都畫出來，比較曲線下面積做爲模型優劣的指標。

ROC 曲線下方的面積（英語：Area under the Curve of ROC），其意義是：

因為是在 1x1 的方格裏求面積，AUC 必在 0~1 之間。假設閾值以上是陽性，以下是陰性；

若隨機抽取一個陽性樣本和一個陰性樣本，分類器正確判斷陽性樣本的值高於陰性樣本之機率=AUC。簡單說：AUC 值越大的分類器，正確率越高。

改變閾值只是不斷地改變預測的正負樣本數，即 TPR 和 FPR，但是曲線本身是不會變的。那麼如何判斷一個模型的 ROC 曲線是好的呢？這個還是要回歸到我們的目的：FPR 表示模型虛報的響應程度，而 TPR 表示模型預測響應的覆蓋程度。我們所希望的當然是：虛報的越少越好，覆蓋的越多越好。所以就是 TPR 越高，同時 FPR 越低（即 ROC 曲線越陡），那麼模型的性能就越好。

從 AUC 判斷分類器（預測模型）優劣的標準：

- AUC = 1，是完美分類器，採用這個預測模型時，存在至少一個閾值能得出完美預測。絕大多數預測的場合，不存在完美分類器。

- 0.5 < AUC < 1，優於隨機猜測。這個分類器（模型）妥善設定閾值的話，能有預測價值。
- AUC = 0.5，跟隨機猜測一樣（例：丟銅板），模型沒有預測價值。
- AUC < 0.5，比隨機猜測還差；但只要總是反預測而行，就優於隨機猜測。

編者：鄭志偉，AI Office，20181015

i. 資料來源：維基百科 ROC 曲線

11. 訓練出來的機器學習模型怎麼評估好壞?

在機器學習領域，對模型的評估非常的重要，只有選擇與問題相匹配的評估方法，才能更好的模型訓練和模型選擇的時候出現的問題，才能更好的對模型進行疊代優化

模型評估主要分爲離線評估和在線評估。針對分類、排序、回歸、序列預測等不同類型的機器學習問題，模型評估指標的選擇也有所不同。知道每種評估指標的精確定義、有針對性的選擇合適的評估指標、根據評估指標的反饋進行模型的調整，這些都是機器學習在模型評估階段的關鍵問題，也是一名合格的演算法工程師應該具備的基本功，爲了瞭解模型的泛化能力，我們需要用某個指標來衡量，這就是性能度量的意義。有了一個指標，我們就可以對比不同模型了，從而知道哪個模型相對好，那個模型相對差，並通過這個指標來進一步調參逐步優化我們的模型。

模型評估指標反映模型效果。在預測問題中，要評估模型的效果，就需要將模型預測結果 f（X）和真實標註的 Y 進行比較，評估指標定義為 f（X）和 Y 的函數：

$$Score = metric(f(X), Y)$$

模型的好壞是相對的，對比不同的模型效果時，使用不同評估指標往往會導致不同的結論。

1.常用分類問題模型評估指標：準確率（Accuracy）、精確率（Precision）、召回率（Recall）、F1 值（F1 score）、ROC 曲線與 AUC、對數損失（Log-Loss）

2.常用回歸問題模型評估指標：平均絕對偏差 MAE（Mean Absolute Error）、均方誤差 RMSE（Root Mean Squared Error）、R Squared（R^2）、Adjusted R Squared

編者：鄭志偉，AI Office，20181015

i. 資料來源：演算法工程師，都要瞭解這幾個模型評估指標

ii. 資料來源：一文讓你徹底理解準確率，精准率，召回率，真正率，假正率，ROC/AUC

iii. 資料來源：如何爲模型選擇合適的度量標準？

12. 如何選擇機器學習模型?

挑選機器學習的演算法（machine learning algorithm）就和挑鞋一樣，我們不會只考慮性能，否則我們都應該穿著要價上千元的輕量慢跑鞋；我們會根據使用情況挑選適合的款式：有些鞋子適合用來站一整天，有些鞋子則適合用來爬山。鞋子的價格通常很重要；當然，鞋子的外型可能比什麼都還重要。

這些考量套用在演算法上也是一樣的。雖然模型的預測能力很重要，但這並不是全部。有些演算法很容易解釋，有些演算法很能處理雜訊和資料缺失等問題。演算法所需的執行時間很重要，而且演算法的名稱有時就跟鞋子外型一樣重要有的人堅持，不管做什麼，分析名稱都得包含「神經網路」（neural network）。

初學者經常面臨著如何從各種各樣的機器學習演算法中選擇解決自己感興趣問題的方法，要解決該問題可以從以下幾個因素來考慮：

- 數據的大小、質量和性質
- 擁有的計算資源和可以接收的計算時間
- 任務的緊迫程度
- 期望從數據中挖掘的內容

即使是經驗豐富的數據科學家也無法在嘗試不同的演算法之前，告訴你哪個演算法性能最好。當然，這裏也不是要提出一種能夠一次性解決演算法選擇的問題，這裏只是希望能夠提供一些指導性的建議，讓你可以根據一些因素先嘗試一些可能性較大的演算法。

機器學習演算法速查表（Which machine learning algorithm should I use?）可以幫助你從各種各樣的機器學習演算法中查找適合於特定問題的演算法。

速查表是針對數據挖掘或分析的初學者，因此在討論這些機器學習演算法時做一些簡單的假設。速查表中推薦的演算法根據一些數據科學家、機器學習專家和開發者的反饋和提示編制的。速查表中的路徑（path）和演算法（algorithm）標籤按照“如果（if）<路徑標籤>，那麼（then）<演算法標籤>”的方式進行查閱。例如：如果你想執行降維（dimension reduction）操作，那麼可以採用主成分分析（principle component analysis）的方法。如果你需要快速進行數值預測（numberic prediction），可以採用決策樹（decision trees）或者邏輯回歸（logistic regression）。如果你需要分層的輸出結果（hieriarchical results），可以選用層次聚類（hieriarchical clustering）

有些時候可能會採用多個分支，而有時也可能沒有非常匹配的演算法選擇。注意：這些演算法路徑是爲了提供一種經驗上的建議，所以可能有些建議不一定準確。我和很多數據科學家

討論關於最優演算法的選擇，得出的結論是只能將所有的演算法全部測試一遍。

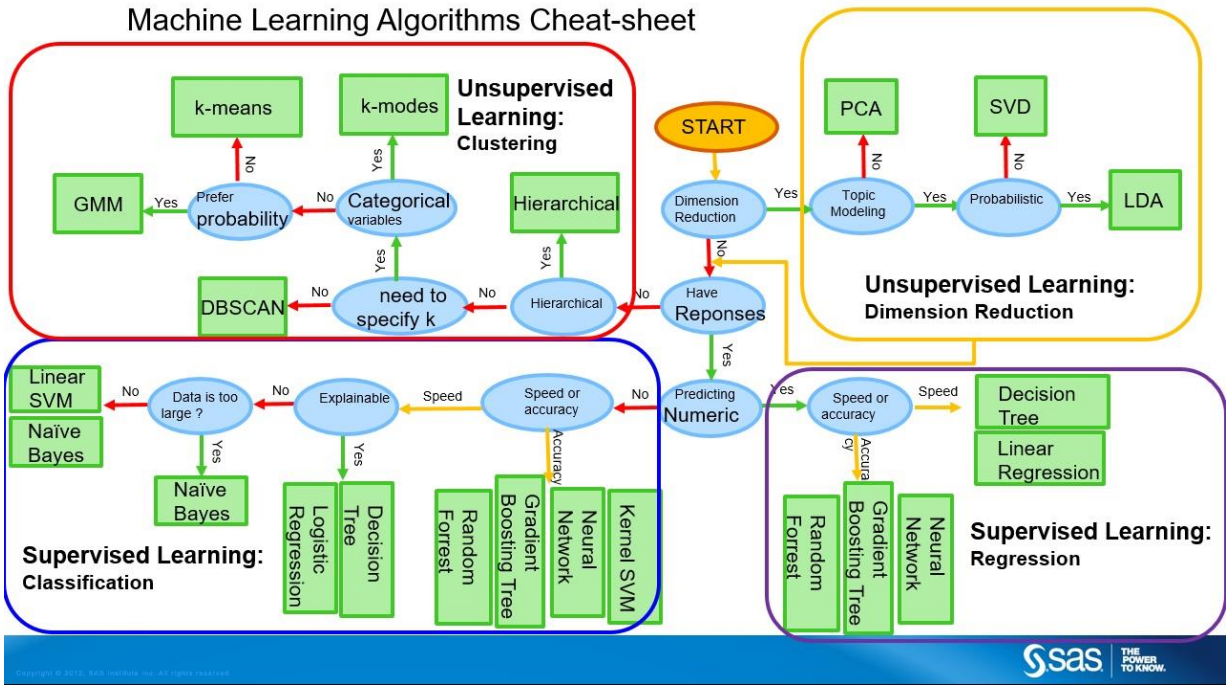


圖 12-1 機器學習演算法速查表

編者：鄭志偉，AI Office，20181015

i. 資料來源：如何選擇機器學習演算法

ii. 資料來源：Which machine learning algorithm should I use?

13. 為什麼要做資料樣本選擇?

當我們決定使用機器學習模型解決業務場景時，在確定好模型的目標和評價指標之後，就可以擼起袖子開始幹了，第一步就是樣本問題，怎麼選擇樣本？

樣本選擇主要是從海量數據中識別和選擇相關性高的數據作爲機器學習模型的輸入，最理想的情況就是選擇了最少量的樣本，模型的效果依然不會變差。

樣本選擇有以下三點好處：

當數據量過大，程式會耗費大量的計算資源，減少數據量能夠縮減模型的運算時間，使得某些因爲數據量過大而無法應用的機器學習模型的問題變得可能。

- 全部數據包含太多冗餘資訊，相關性太低的數據對模型解決業務問題是沒有任何幫助的，徒增浪費資源煩惱。
- 只要是數據，都會有噪聲存在，不管是錯誤的還是重複的噪聲。樣本選擇過程中去除了噪聲，改善了數據質量。
- 樣本選擇常用數據去噪、採樣這些簡單有效的方法，也可以使用複雜的方法，通過搜索

整個數據集或利用演算法來實現樣本選擇，這類方法就是原型選擇和訓練集選擇。

在數據去噪的過程中，最重要的就是怎麼識別噪聲，識別出噪聲之後才可以做直接過濾或者修改數據等操作。噪聲數據可能是重複值、缺失值、超出範圍的異常值等，也可能是標註錯誤，對標註錯誤的處理方法常見的是集成過濾、交叉驗證過濾和反覆運算分割過濾三種方法，都是基於融合或者投票思想進行數據過濾的。

採樣是一種統計技術，從整體選擇一部分進行推論。一個好的樣本子集應該具有無偏性和很小的樣本方差，其中無偏性指的是對樣本的期望等於全體樣本的期望。樣本方差是衡量樣本估計值和真實值的偏差，小方差能保證估計值不會產生太大的偏差。

樣本採樣的方法主要有，無放回簡單隨機抽樣，有放回簡單抽樣，平衡採樣，整群採樣，分層採樣。無放回簡單隨機採樣就是隨機抽取固定數量的樣本。有放回簡單抽樣是指每次抽取一條樣本之後不將該樣本從原始數據中剔除，繼續抽，可能這條樣本會被抽中多次。平衡採樣是指根據目標進行採樣，十分適合在不平衡分類中使用。假如正負樣本的比例在 1：100，我們想要得到正負樣本比在 1：10 的數據樣本，那麼在正樣本數據中進行上採樣，把正樣本複製 10 遍，在負樣本數據中進行下採樣，隨機刪除部分樣本保留原來的十分之一。整群採樣是指先將數據集分成互斥的幾個群，然後在這幾個群中分別進行簡單隨機抽樣作爲樣本集。分層抽樣是指將數據集劃分成不同的層，在層內部進行數據採樣，最後匯合成總樣本集。

編者：鄭志偉，AI Office，20181015

i. **資料來源：開始構建機器學習模型之前，我們該怎麼選擇樣本？**

14. 什麼是交叉驗證（Cross-validation）？

交叉驗證在機器學習上通常是用來驗證「你設計出來模型」的好壞。交叉驗證主要使用在下面兩個場合：

- 數據庫沒有先切割好「訓練資料（Training data）」和「測試資料（Testing data）」
- 你要從「訓練資料（Training data）」找到一組最合適參數出來，比如 SVM 的懲罰參數（Penalty parameter），就可以從訓練資料（Training data）做交叉驗證找出來，而不是從「測試資料（Testing data）」得到參數。

機器學習最忌諱把「測試資料（Testing data）」偷偷拿進到模型內訓練或是找參數。在做模型 performance 評估時，要記住一件事情「測試資料（Testing data）」絕對不能進到模型內訓練或是找參數。一般在說交叉驗證通常會分成 Resubstitution、Holdout CV、Leave-one-out CV、K-fold CV，以下用圖例說明這些方法：

假設有一組數據，共 **20 筆資料（兩個類別）**

12 筆綠色的資料{G1, G2,..., G12}與 8 筆淡紅色的資料{R1, R2,..., R8}

G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	R1	R2	R3	R4	R5	R6	R7	R8

1. Resubstitution：可以稱爲自我一致性評估法，是用相同的資料進行訓練和測試。這種方法會觸碰到我剛提到的機器學習最忌諱的事情，所以這個方法通常只用在從訓練資料集找參數用，執行起來會比較快，但不太會用在驗證模型的優劣。

2.Holdout CV：Holdout 是指從資料集中隨機取得 p%資料當作「訓練資料（Training data）」和剩下的（1-p）%當做「測試資料（Testing data）」。最後的結果（Predication results）在和測試資料的真實答案（ground truth）進行成效比對（Performance Comparison）。這個隨機不是不考慮資料的類別，也就是說 Holdout 是從每一類都取 p%資料當作「訓練資料（Training data）」，從每一類剩下的（1-p）%當做「測試資料（Testing data）」，如下圖所示。此例 Holdout 假設 p=75%當做訓練資料，剩下 25%當做測試資料

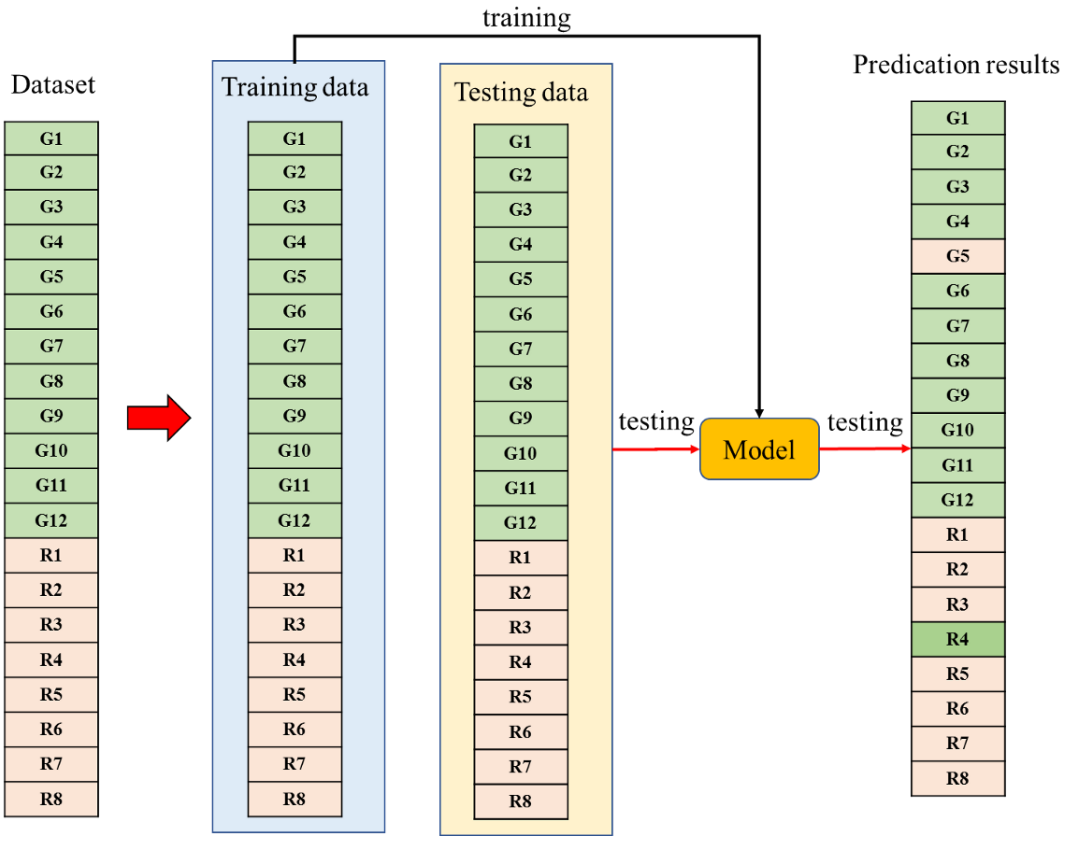


圖 14-1 Resubstitution

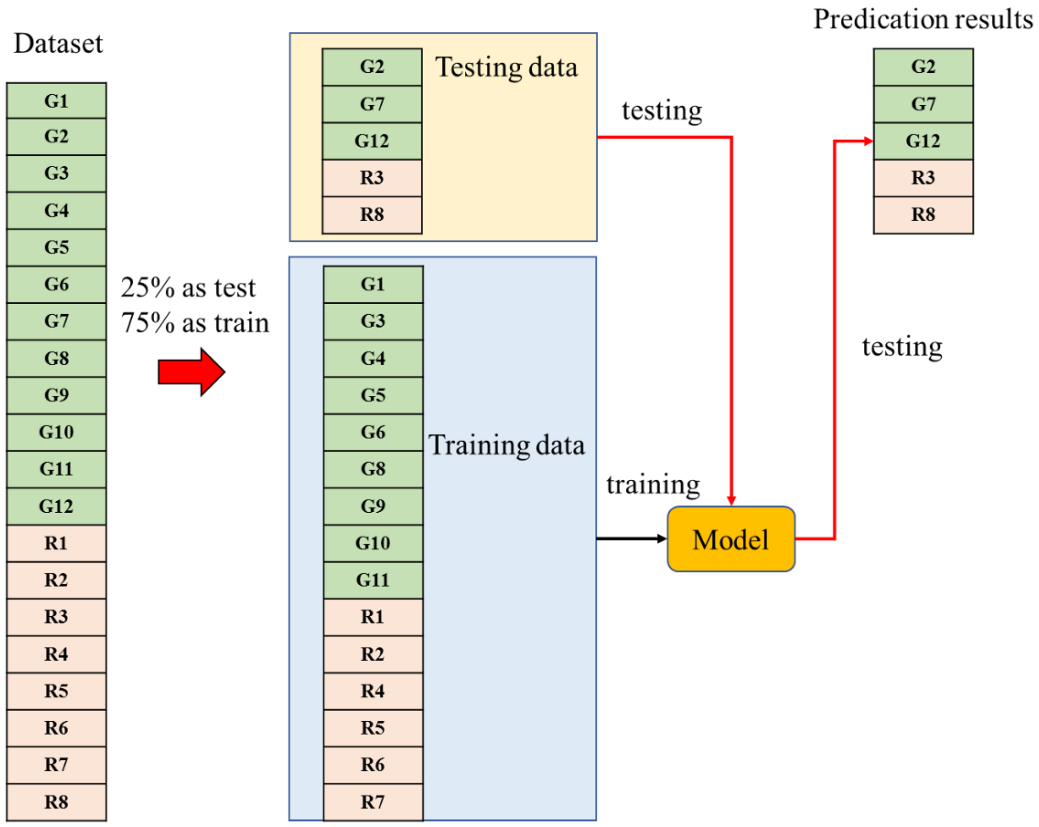


圖 14-2 Holdout CV

3. Leave-one-out CV：每次都將每一筆資料都視為「測試資料（Testing data）」，剩下的做爲「訓練資料（Training data）」，如此重複進行直到每一筆都被當做「測試資料（Testing data）」為止，如圖所示。最後的結果（Predication results）在和真實答案（ground truth）進行成效比對（Performance Comparison）。

圖 14-3 Leave-one-out CV

4. K-fold CV：K-fold 是比較常用的交叉驗證方法。做法是將資料隨機平均分成 k 個集合，然後將某一個集合當做「測試資料（Testing data）」，剩下的 k-1 個集合做爲「訓練資料（Training data）」，如此重複進行直到每一個集合都被當做「測試資料（Testing data）」為止。最後的結果（Predication results）在和真實答案（ground truth）進行成效比對（Performance Comparison）。這個隨機分 k 個集合也是要考慮資料的類別，也就是說 K-fold 是從每一類都隨機分割成 k 個集合，如下圖所示，在此舉 3-fold CV 的例子

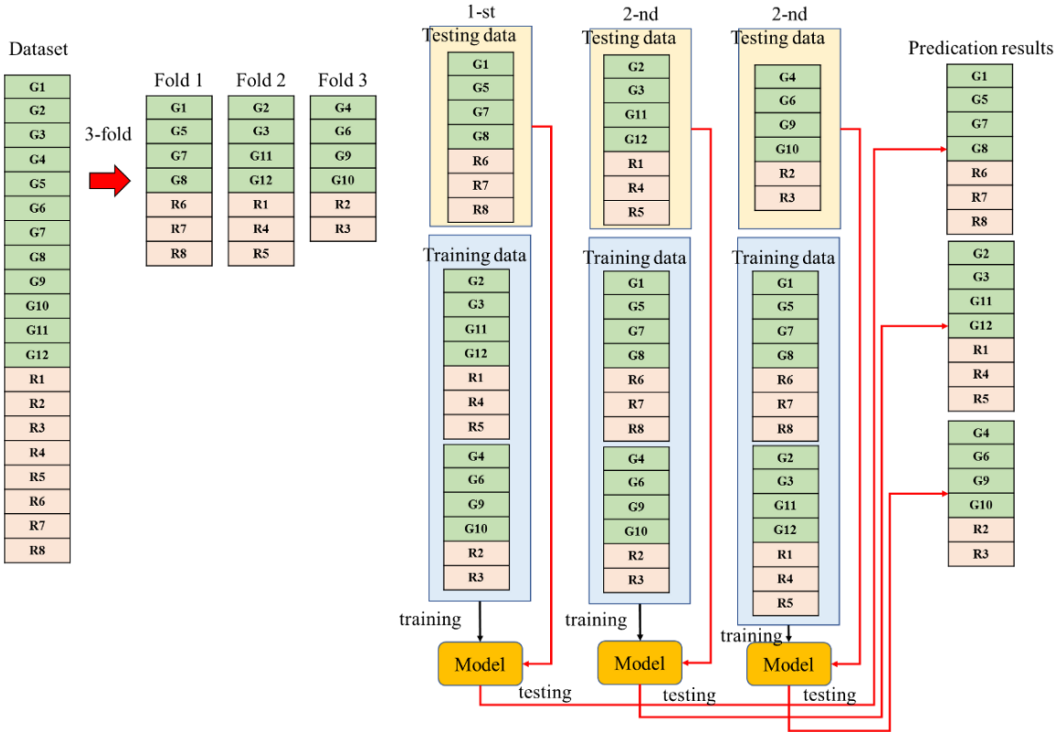


圖 14-4 K-fold CV

編者：鄭志偉，AI Office，20181015

i. 資料來源：交叉驗證（Cross-validation, CV）

15. 什麼是損失函數 Loss Function?

通過前面的介紹，大家對資料整理，特徵選擇，演算法選擇都有了一個整體的認識。但是選擇演算法後，演算法是如何工作的，如何確定演算法模型對真實資料的擬合效果怎麼樣，這就是損失函數的作用。在深度學習、機器學習中常會設計一個損失函數（loss function），用來評估機器有多接近我們的理想目標。越接近理想的行爲，比方說文字與圖片辨識得越準確、下棋下得越好、翻譯得越正確，那損失函數的值就會越小，如此一來只要找到適當的參數，讓損失函數盡量小，那機器就能執行我們希望的行爲。

這個損失函數一定都是數學上能計算的函數，而這就是數學上標準的極值問題，有很多方法來處理。其實不只是機器學習與人工智慧，許多的科學、工程技術、甚至日常生活問題，都能以極值問題的形式出現。在數學上，我們知道梯度是函數值上升最快的方向，而反梯度方

向則是下降最快的方向，所以我們只要把「損失函數」對我們想調整的「參數」去微分得到梯度，然後就知道要往哪個方向去調整參數、進而讓損失函數有效地降低。

演算法模型的訓練就是通過利用損失函數還衡量模型是否已經收斂，如果未收斂，通過計算損失函數的梯度，沿著梯度下降的方法不斷不斷調整參數的值，重新計算演算法的輸出。周而復始，不斷反覆運算直到模型收斂，損失函數達到一個極小值爲止。

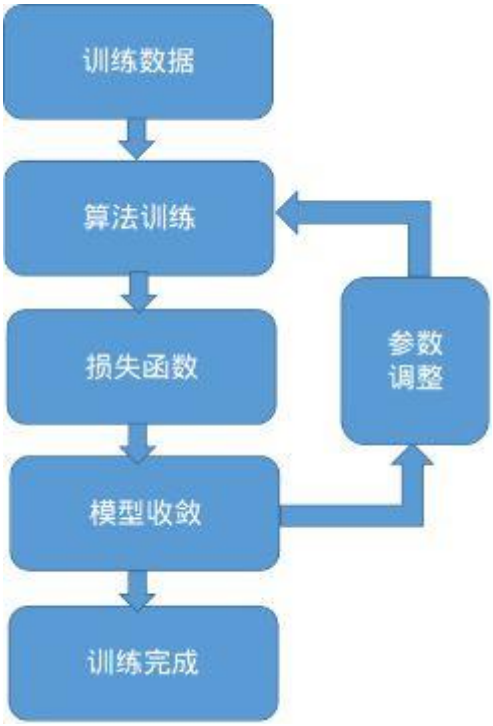


圖 15-1 機器學習模型的訓練過程

編者：鄭志偉，AI Office，20181015

i. **資料來源：人工智慧浪潮下的數學教育**

ii. **資料來源：機器學習演算法應用 - 損失函數**

16. 機器學習中的偏差（Bias）和方差（Variance）？

首先，假設你知道訓練集和測試集的關係。簡單來講是我們要在訓練集上學習一個模型，然後拿到測試集去用，效果好不好要根據測試集的錯誤率來衡量。但很多時候，我們只能假設測試集和訓練集的是符合同一個數據分布的，但卻拿不到真正的測試數據。這時候怎麼在只看到訓練錯誤率的情況下，去衡量測試錯誤率呢？

由於訓練樣本很少（至少不足夠多），所以通過訓練集得到的模型，總不是真正正確的。（就算在訓練集上正確率 100%，也不能說明它刻畫了真實的數據分布，要知道刻畫真實的數據分布才是我們的目的，而不是只刻畫訓練集的有限的數據點）。而且，實際中，訓練樣本往往還有一定的噪音誤差，所以如果太追求在訓練集上的完美而採用一個很複雜的模型，

會使得模型把訓練集裡面的誤差都當成了真實的數據分布特徵，從而得到錯誤的數據分布估計。這樣的話，到了真正的測試集上就錯的一塌糊塗了（這種現象叫過擬合）。但是也不能用太簡單的模型，否則在數據分布比較複雜的時候，模型就不足以刻畫數據分布了（體現為連在訓練集上的錯誤率都很高，這種現象叫欠擬合）。過擬合表明採用的模型比真實的數據分布更複雜，而欠擬合表示採用的模型比真實的數據分布要簡單。

偏差：描述的是預測值（估計值）的期望與真實值之間的差距。偏差越大，越偏離真實數據，如圖 16-1 第二行所示。

方差：描述的是預測值的變化範圍，離散程度，也就是離其期望值的距離。方差越大，數據的分布越分散，如圖 16-1 右列所示。

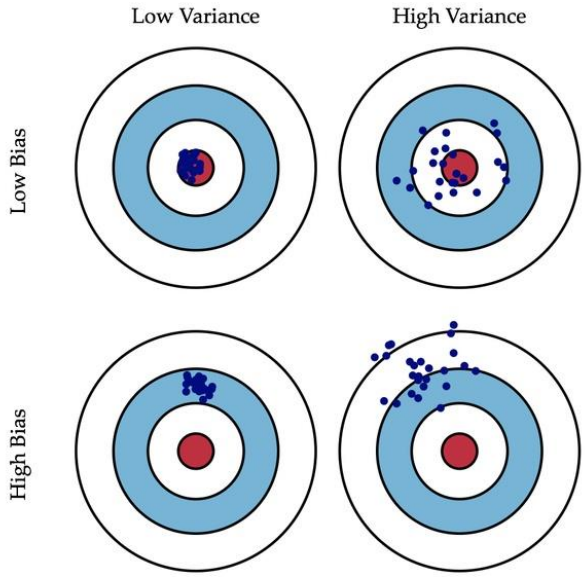


圖 16-1 方差與偏差

在統計學習框架下，大家刻畫模型複雜度的時候，有這麼個觀點，認為誤差=方差+偏差+噪聲（Error = Bias +Variance+Noise）。這裡的誤差大概可以理解為模型的預測錯誤率，是有兩部分組成的，一部分是偏差（Bias），是由於模型太簡單而帶來的估計不準確的部分。另一部分是方差（Variance），是由於模型太複雜而帶來的更大的變化空間和不確定性。

為了讓誤差儘量小，我們在選擇模型的時候需要平衡偏差和方差所占的比例，也就是平衡過擬合和欠擬合。

方差和偏差一般來說，是從同一個數據集中，用科學的採樣方法得到幾個不同的子數據集，用這些子數據集得到的模型，就可以談他們的方差和偏差的情況了。方差和偏差的變化一般是和模型的複雜程度成正比的，就像圖 16-1 那四張小圖片一樣，當我們一味的追求模型精確匹配，則可能會導致同一組數據訓練出不同的模型，它們之間的差異非常大。這就叫做方差，不過他們的偏差就很小了，如下圖所示：

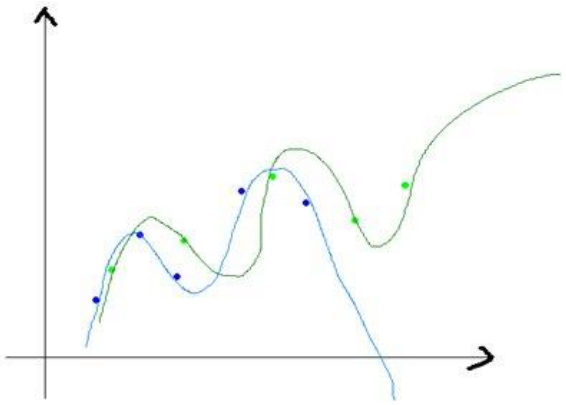


圖 16-2 以 N 次曲線擬合數據

上圖的藍色和綠色的點是表示一個數據集中採樣得到的不同的子數據集，我們有兩個 N 次的曲線去擬合這些點集，則可以得到兩條曲線（藍色和深綠色），它們的差異就很大，但是他們本是由同一個數據集生成的，這個就是模型複雜造成的方差大。模型越複雜，偏差就越小，而模型越簡單，偏差就越大，方差和偏差是按下面的方式進行變化的：

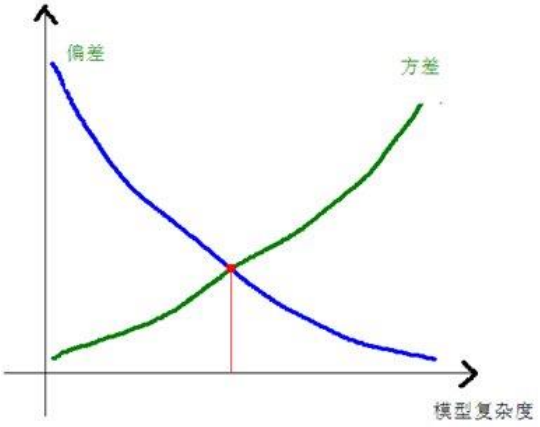


圖 16-3 最佳模型複雜度

當方差和偏差加起來最優的點，就是我們最佳的模型複雜度。

編者：鄭志偉，AI Office，20181015

i. 資料來源：機器學習中的偏差和方差

17. 什麼是過擬合（Over Fitting）與欠擬合（Under Fitting）?

在模型評估與優化過程中，我們經常會遇到過擬合（over fitting）和欠擬合（under fitting）的情況，那麼到底什麼是過擬合和欠擬合。過擬合是指模型對於訓練集數據呈現過當的情況，從評估指標上看就是模型在訓練集上表現的很好，但是在測試集合新的數據上就表現得很差。欠擬合是指在訓練集上和測試集上表現得都不好的情況。

圖 17-1 左圖就是欠擬合的情況，虛線沒有很好的區分數據類別，不能夠很好的擬合數據，圖 17-1 右圖的模型過於複雜，把噪聲數據的特徵也學習到模型中，導致模型泛化能力下降，在後期應用過程中很容易輸出錯誤的預測結果。

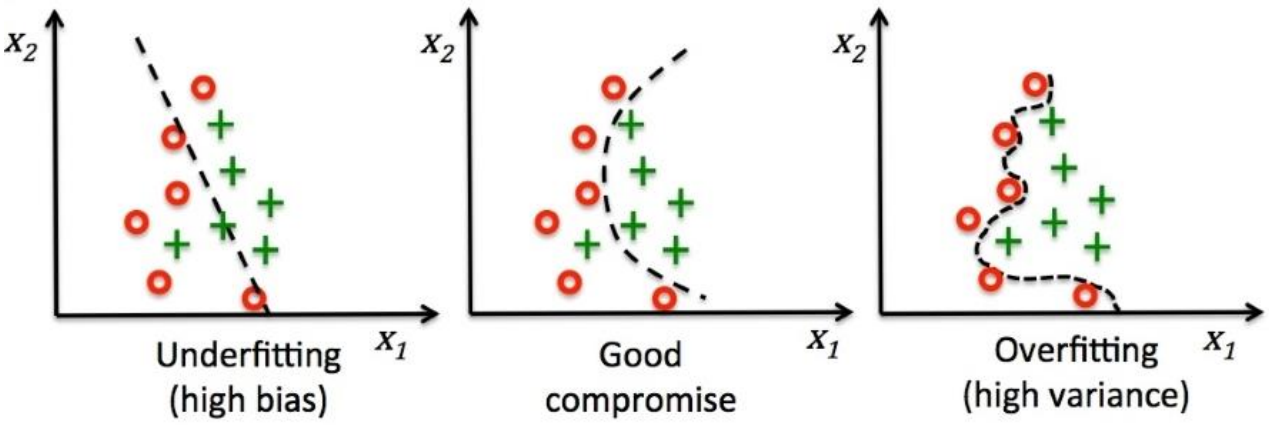


圖 17-1 欠擬合與過擬合

編者：鄭志偉，AI Office，20181015

i. 資料來源：怎麼解決過擬合和欠擬合情況

18. 什麼是機器學習模型的泛化能力（generalization ability）?

所謂泛化能力（generalization ability）是指機器學習演算法對新鮮樣本的適應能力。學習的目的是學到隱含在資料對背後的規律，對具有同一規律的學習集以外的資料，經過訓練的網路也能給出合適的輸出，該能力稱爲泛化能力。

機器學習的主要目標是對訓練集之外的樣本進行泛化。因為無論有多少資料，都不太可能在測試中再次看到完全相同的例子。在訓練集上具有良好表現很容易。機器學習初學者最常犯的錯誤是把模型放在訓練資料中進行測試，從而產生成功的錯覺。如果被選擇的分類器在新的資料上進行測試，一般情況，結果往往和隨機猜測相差無幾。所以，如果你雇傭他人建立分類器，一定要留一些資料給你自己，以便在他們給你的分類器中進行測試。相反，如果有人雇傭你建立一個分類器，請保留一部分資料對你的分類器進行最終測試。

編者：鄭志偉，AI Office，20181015

i. 資料來源：A Few Useful Things to Know about Machine Learning

ii. 資料來源：關於機器學習必須要瞭解的幾個要點

19. 什麼是維度災難（curse of dimensionality）?

維度增多主要會帶來高維空間資料稀疏化問題，也就是說，資料會更加的分散，因而就需要更大的資料量才能獲得較好的偏差（bias）和方差（variance），達到較好的預測效果。

另一方面看，當維度增加時，也可能導致過擬合現象，訓練集上表現好，但是對新資料缺乏泛化能力。高維空間訓練形成的分類器，相當於在低維空間的一個複雜的非線性分類器，這種分類器過多的強調了訓練集的準確率甚至於對一些錯誤/異常的資料也進行了學習，而正

確的資料卻無法覆蓋整個特徵空間。因此，這樣得到的分類器在對新資料進行預測時將會出現錯誤。

更詳細說明可以參考以下三篇文章：

i. 怎樣理解"curse of dimensionality"，

ii. 談談機器學習中的維度災難

iii. 理解機器學習中的維度詛咒

圖 19-1 是隨著維度（特徵數量）的增加，分類器性能的描述：

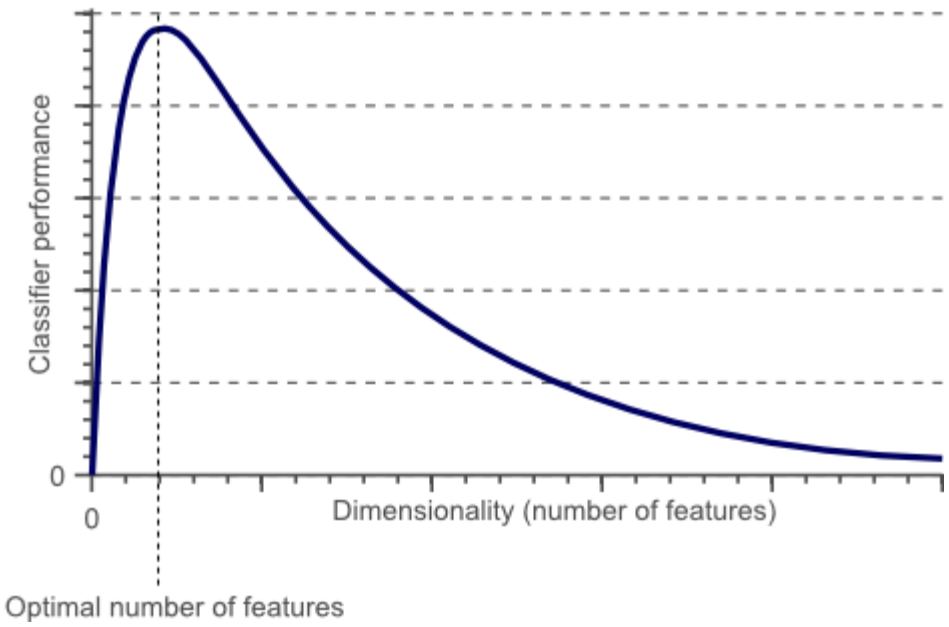


圖 19-1 隨著維度增加，分類器性能提升；維度增加到某值後，分類器性能下降

但是如何避免維度災難呢？並沒有固定的規則規定在分類問題中應該使用多少個特徵，因為維度災難問題和訓練樣本的資料有關。事實上，避免維度災難主要有幾種方法：降維、從原特徵的基礎上構造新特徵、交叉驗證

編者：鄭志偉，AI Office，20181015

i. **資料來源：維度災難**

ii. **資料來源：怎樣理解"curse of dimensionality"**

iii. **資料來源：談談機器學習中的維度災難**

iv. **資料來源：理解機器學習中的維度詛咒**

20. 想要降低過擬合（over fitting）的風險，該怎麼辦？

- 從數據入手，獲得更多的訓練數據。使用更多的訓練數據是解決過擬合問題最有效的方

法，更多的數據能讓模型學習到更多有效的特徵，減少噪聲的影響。在實際的工作中，直接增加訓練數據有一定困難，但是可以通過一定的規則來擴充。比如，在圖像分類問題上，通過圖像的平移、旋轉、縮放等方式擴充數據，更複雜的方法，還可以使用生成對抗網絡來合成大量的新的訓練數據。

- 降低模型的複雜度。在數據較少的時候，模型過於複雜是產生過擬合的主要原因，適當的降低模型複雜度可以避免模型擬合過多的噪聲數據。例如，在決策樹模型中，適當的剪枝或者降低樹的深度；在神經網絡模型中減少網絡層數、神經元個數等。
- 正則化方法。給模型的參數加上一定的正則約束，比如將權值的大小加入到損失函數中。
- 集成學習方法。集成學習是把多個模型集成在一起，來降低單一模型的過擬合風險，比如 Bagging 方法。

編者：鄭志偉，AI Office，20181015

資料來源：怎麼解決過擬合和欠擬合情況

21. 想要改善欠擬合（under fitting）該怎麼辦？

- 擴展新的特徵。當特徵不足或者現有的特徵數據跟樣本標籤的相關性不強時，很容易處罰欠擬合現象。通過挖掘上下文特徵，組合特徵等行的特徵，一般都會得到不錯的效果。在深度學習潮流中，有很多模型可以幫助完成特徵工程，比如因數分解機、梯度提升決策樹、Deep-crossing 等都可以成為擴展特徵的方法。
- 增加模型的複雜度。簡單模型的學習能力較差，通過增加模型的複雜度可以使模型擁有更強的擬合能力。例如，在線性模型中添加高次項，在神經網絡模型中增加網絡層數或者神經元個數等。
- 減小正則化係數。正則化是用來防止過擬合的，當模型出現欠擬合的時候，需要根據實際情況減小正則化係數。

編者：鄭志偉，AI Office，20181015

i. **資料來源：怎麼解決過擬合和欠擬合情況**

觀念篇（二）深度學習基礎觀念

22. 什麼是深度學習 Deep Learning?

自從 2016 年 3 月 AlphaGo 以四比一擊敗李世乭以後，AlphaGo 所使用的深度學習技術引起了各界的關注。事實上，早在 AlphaGo 問世之前，深度學習技術即已廣泛應用在各個領域。當我們對 iPhone 的語音助理軟體 Siri 說一句話，Siri 可以將聲音訊號辨識成文字，用的就是深度學習的技術；當我們上傳一張相片到 Facebook，Facebook 可以自動找出相片中的人臉，用的也是深度學習的技術。其實人們早已享受深度學習所帶來的便利很長一段時間了。

深度學習是機器學習的一種方法，「深度學習是讓機器模擬人腦的運作方式，進而和人類一樣具備學習的能力。」這個科普的說法，相信大家已耳熟能詳。會有這樣的說法，是因為深度學習中，人類提供的函數集是由類神經網絡（artificial neural network）的結構所定義。

類神經網絡和人腦確實有幾分相似之處，我們都知道人腦是由神經元（neuron）所構成，類神經網絡也是由「神經元」連接而成。類神經網絡中的神經元構造及其運作方式如圖 22-1 所示。每個神經元都是一個簡單的函數，這些函數的輸入是一組數值（也就是一個向量），輸出是一個數值。以圖 22-1 的神經元為例，該神經元的輸入為左側橘色框內的 2、-1、1 三個數值，輸出爲右側藍色框內的數值 4。

那麼，神經元是如何運作的呢？ 每個輸入都有一個對應的權重（weight），圖 22-1 中每個輸入對應的權重，分別為灰色框內的 1、-2、-1 三個數值。先將每個輸入數值和其對應權重相乘後加總，再加上綠色框內的閾值（bias）後，其總和便成為神經元中啟動函數（activation function）的輸入。圖 22-1 中啟動函數的輸入是 4，也就是 2×1+（−1）×（−2）+1×（−1）+1=4

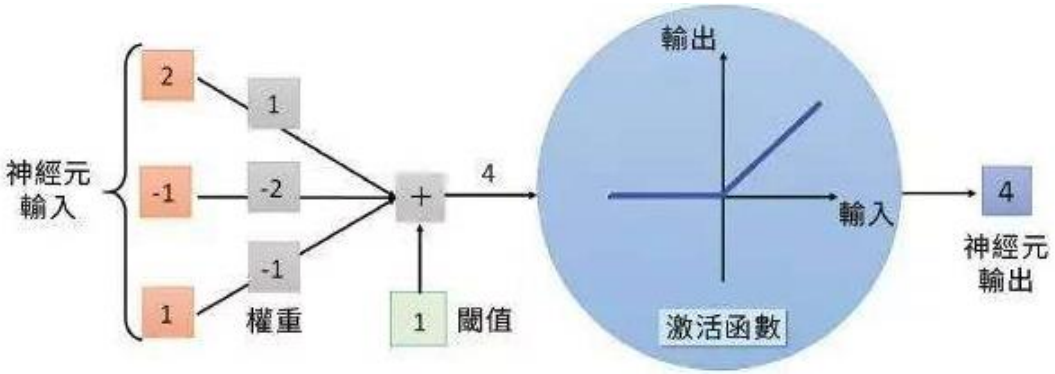


圖 22-1 類神經網絡中單一神經元及其運作方式

啟動函數是由人類事先定義好的非線性函數，其輸入和輸出都是一個數值，而其輸出就是神經元的輸出。圖 22-1 中的啟動函數， 其輸入和輸出的關係如藍色圓域所示（橫軸代表輸入、縱軸代表輸出），其中當輸入小於 0 時，輸出爲 0；當輸入大於 0 時，輸出等於輸入。這種啟動函數稱為整流線性單元（rectified linear unit，ReLU），是目前常用的一種啟動函數。圖 22-1 中的啟動函數輸入為 4，因為大於 0，故神經元的輸出就是 4。神經元中的權重和閾值都稱為參數（parameter），它們決定了神經元的運作方式。

步驟一：類神經網絡就是函數集

瞭解神經元後，接著來看類神經網絡。類神經網絡由很多神經元連接而成，人類只需要決定類神經網絡的連結方式，機器可以自己根據訓練資料找出每個神經元的參數。圖 22-2 是一個類神經網絡的例子，上方的類神經網絡共有六個神經元，分別排成三排，橘色方塊代表外界的輸入，外界的輸入可以是圖片、聲音訊號或棋盤上棋子的位置等等，只要能以向量（一組數字）表示即可。以圖片為例，一張 28×28 大小的黑白圖片，可以視為一個 784（=28×28）維的向量，每一分量對應到圖片中的一個圖元，該分量的值表示圖元顏色的深淺，接近黑色其值就接近 1，反之就接近 0。來自外界的資訊被輸入給第一排的藍色神經元，藍色神經元的輸出是第二排黃色神經元的輸入，黃色神經元的輸出則是下一排綠色神經元的輸入，因爲綠色神經元的輸出沒有再轉給其他神經元，故其輸出就是整個類神經網絡的輸出。

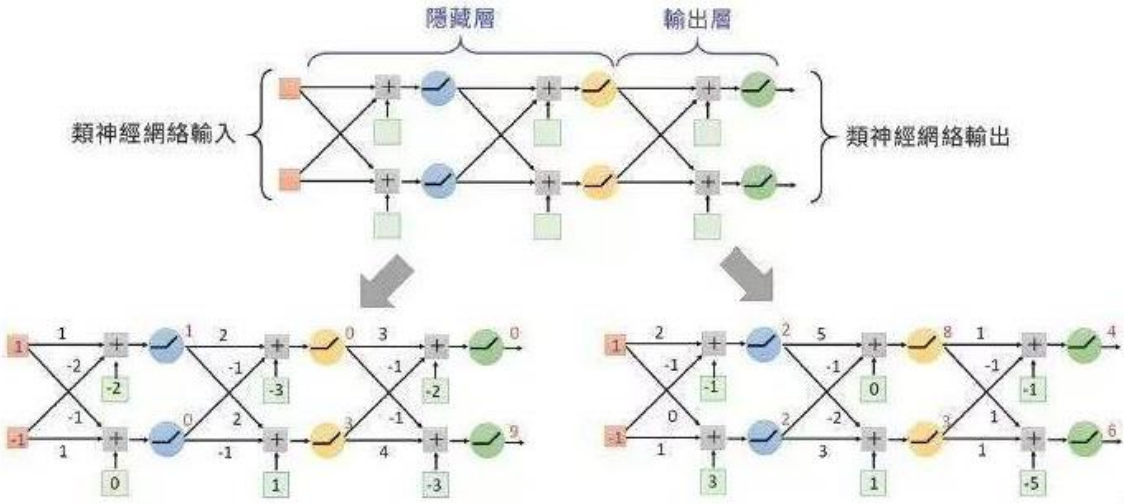


圖 22-2 圖上方為一完全連接前饋式網絡結構，下方爲兩組不同的參數示例，分別代表兩個不同的函數。輸入同樣的數值，左下和右下的神經網絡會有不同的輸出。

圖 22-2 這種類神經網絡結構，稱為完全連接前饋式網絡（fully connected feed forward network）。在這種架構中，神經元排成一排一排，每排稱為一「層」（layer），每層神經元的輸出爲下一層各神經元的輸入，最後一層稱爲「輸出層」（output layer），其他層則稱之為「隱藏層」（hidden layer）。所謂深度學習的「深」，意味著有很多的隱藏層（至

於要多少隱藏層才足以稱之爲「深」就見仁見智了）。

當類神經網絡中每個神經元參數都確定時，該類神經網絡就是一個非常複雜的函數。同樣的網絡結構，若參數不同將形成不同的函數。例如：圖 22-2 左下圖和右下圖分別具備兩組不同的參數，當參數如左下圖，若輸入 1 和-1，這個類神經網絡所定義的函數輸出爲 0 和 9；同理，以右下圖的參數可得到另一個函數，如果輸入一樣的 1 和-1，這個函數的輸出則爲 4 和 6。因為一組參數等於一個函數，如果先把神經元連接確定，再讓機器根據訓練資料自己找出參數時，就相當於先提供一個函數集，再讓機器從函數集自行選出有用的函數。

類神經網絡的結構，目前仍需要人類在機器學習前事先決定。如果結構設定不當，由此結構所定義的函數集根本找不到好的函數，接下來再怎麼努力都是徒勞。這就好比想要在大海撈針，結果針根本不在海裡。

不同任務適用的類神經網絡結構並不相同，例如卷積式類神經網絡（convolutional neural network）這種特殊結構，特別適合做影像處理。卷積式類神經網絡也是 AlphaGo 的架構，但 AlphaGo 中的卷積式類神經網絡和前者略有不同，沒有最大池化（max pooling）這個影像處理的常用結構，這應該是考慮圍棋特性而刻意設計的。

目前，雖然有些技術已能讓類神經網絡自動決定結構，但這些技術尚未有太多成功的應用。想要決定網絡結構，通常還是必須仰賴深度學習技術使用者的經驗、直覺，以及嘗試錯誤，其中的「運用之妙，存乎一心」，這裡無法詳談。

步驟二：定義函數的優劣

有了類神經網絡結構作爲函數集後，接下來就要定義函數的優劣，以便在下一步驟讓機器挑選出最佳函數，但什麼叫作好的函數呢？讓我們舉個具體的例子來說明。假設任務是手寫數字辨識，也就是讓機器檢視圖片，每張圖中有手寫數字，機器必須辨認出這個數字。執行手寫數字辨識的類神經網絡，其輸出層有 10 個神經元分別對應到從 0 到 9 的數碼，而每個神經元的輸出值則分別表示對應數碼的信心分數，機器會把信心分數最高的數碼當作最終的輸出。

在手寫數字辨識中，訓練資料是一堆圖片和每張圖片中的數碼（數碼需由人工標註）。這些訓練資料告訴機器，當訓練資料中某張圖片輸入時，要辨識出哪個數碼才正確。如果將訓練資料中的圖片一張一張輸入到某些函數來辨識，越能夠正確辨識結果（亦即在輸出層中圖片數碼的信心分數最高）的函數，可能就越優秀。當然也可能發生某函數能準確辨識訓練資料，可是輸入新圖片時卻發生辨識錯誤的情況。因此函數優劣的定義本身是很大的學問，只考慮訓練資料的正確率是不夠的，此處不深入細談。

步驟三：找出最佳函數

根據訓練資料可以決定一個函數的優劣，接下來就要從類神經網絡所定義的函數集中，找到最佳的函數，也就是最佳的參數組合。我們可以把找出最佳函數的過程，想像成是機器在「學習」。

如何找出最佳的類神經網絡參數呢？最無腦的方法就是暴力窮舉類神經網絡中的所有可能參數值。然而，爲了讓機器完成複雜的任務，現在的類神經網絡往往非常龐大，其中可能包括數十個隱藏層，每個隱藏層有上千個神經元，因此參數數目可能動輒千萬以上。要窮舉這上千萬的參數組合，再根據訓練資料計算每個參數組合（函數）的優劣來找出最佳函數，就算是電腦也辦不到，因此暴力窮舉法是不切實際的。

目前最常採用的「學習」方法稱為「梯度下降法」（gradient descent）。在此法中，機器先隨機指定第一組參數， 再稍微調整第一組參數，找出比第一組參數略佳的第二組參數，接下來再稍微調整第二組參數，找出比第二組參數略佳的第三組參數，以此類推，讓這個步驟反覆進行，直到找不出更佳的參數時就停止。參數的調整次數可能多達上萬次，這就是經常聽說「深度學習需要耗用大量運算資源」的原因。

因爲類神經網絡中有大量參數，所以還會使用名爲「反向傳遞法」（backpropagation）的演算法，來提高參數調整的效率。梯度下降法和反向傳遞法沒有什麼高深的數學，理論上只要高三以上的理組學生就有能力理解這個方法，現在的深度學習軟體都具備這些方法。

雖然用梯度下降法可以訓練出厲害的 AlphaGo，但這個方法本身其實不怎麼厲害。它無法保證一定能從函數集中挑出最佳函數，而僅能從類神經網絡定義的函數集中，找出局部最好的函數。更糟的是，這個演算法有隨機性，每次找出來的函數可能都不一樣，因此能找出多好的函數要靠點運氣。也因此，在深度學習的領域裡，充斥各種可以幫助梯度下降法找到較佳函數的「撇步」，但至今尚未有任何撇步可以保證能找到最佳函數。

編者：鄭志偉，AI Office，20181015

i. **資料來源：什麼是深度學習——機器學習的捲土重來**

23. 深度學習是新的突破嗎？

其實，上述的類神經網絡技術在 1980 年代就已經發展成熟，只是當時並不用「深度學習」這個詞彙， 而稱為「多層次感知器」（multi-layer perceptron, MLP）。許多人把深度學習近年的走紅，歸功於辛騰（Geoffrey Hinton）在 2006 年提出， 以限制波茲曼機（Restricted Boltzmann Machine, RBM）初始化參數的方法。因此曾有一度，深度學習和多層次感知器的差異在於，隨機初始化參數的叫作多層次感知器；用限制波茲曼機初始化的稱爲深度學習。不過實際上，只要給予適當的啟動函數、足夠的訓練資料，限制波茲曼機法所帶來的幫助並不顯著，故此方法已不能算是深度學習的同義詞。

今日深度學習所應用的類神經網絡，和 1980 年代的多層次感知器雖然本質上非常相似，但還是有些不同。首先，80 年代的網絡通常不超過三層，但現在的網絡往往比三層深得多，例如語音辨識需要七、八層，影像辨識需要 20 餘層，微軟用來辨識影像的「深度殘留網絡」（deep residual network），甚至深達 152 層。另一個不同是，過去比較流行用 S 型函數（sigmoid function）作為啟動函數，但 S 型函數在網絡很深時，難以訓練出好的結果，故近年較流行用前述整流線性單元，在訓練很深的網絡時，整流線性單元的效果遠優於 S 型函數。此外，現在還有技術可以讓機器自己決定啟動函數。

另外，雖然訓練還是以梯度下降法為主，卻有一些新的訊練技巧，例如：Adam 演算法可以減少訓練時參數更新的次數，加速網絡提早完成訓練 Dropout 演算法在訓練時隨機丟掉一些神經元，可讓網絡在遇到沒看過的資料時表現得更好。硬體的發展同樣也不容忽視，例如以圖形處理器（graphics processing unit, GPU）來加速矩陣運算，使得訓練時間大幅縮短，可以多次嘗試錯誤，縮短開發週期，也是加速深度學習這個領域發展的原因之一。

編者：鄭志偉，AI Office，20181015

i. **資料來源：數理人文雜誌第 10 期 什麼是深度學習？**

24. 深度學習為什麼需要「深」？

為什麼類神經網絡需要很多層神經元？簡單的答案是，深層類神經網絡比淺層厲害。例如在實務上，較深的類神經網絡可得到較低的語音辨識錯誤率。但是這個論述的說服力夠嗎？

如果深層類神經網絡較淺層效能好的原因，來自神經元數量的增加，而跟多層次的結構沒有關係，那麼，把所有神經元放在同一層，會不會效果跟深層網絡一樣好？甚至已有理論保證，單一隱藏層的淺層類神經網絡，只要神經元夠多，就可以描述任何函數。於是把神經元排成多層的理由似乎更削弱了，甚至有人懷疑把神經元排成多層的深度學習只是噱頭。

有趣的是，雖然淺層類神經網絡可以表示和深層類神經網絡一樣的函數，但淺層網絡卻需要更多神經元才能描述一樣的函數。為什麼會這樣呢？打個比方，如果把網絡的輸入比擬爲原料、輸出比擬爲產品，那麼類神經網絡就是一條生產線，每個神經元是一位工作人員，把來自前一站的半成品加工再送往下一站。假設現在輸入的是圖片，輸出是圖片中的物件，第一層神經元的工作可能是偵測圖片中直線、橫線的位置 4 ，將結果交給第二層，第二層神經元根據第一層的輸出，判斷圖片中有沒有出現圓形、正方形等幾何圖案，再將結果交給第三層……直到最後一層的神經元輸出結果。衆所周知，生產線可以提高生產效率，因此很深的類神經網絡好比一條分工很多的生產線，可以比淺層網絡更有效率。

同樣的函數，淺層網絡需要較多神經元，也意味著需要較多的訓練資料。這表示淺層網絡想達成深層網絡的效能，需要更多的訓練資料。也就是說，如果固定訓練資料量，深層網絡將

有淺層網絡難以企及的效能。

是的，你沒看錯，深層網絡可以用比淺層網絡少的訓練資料，就達成同樣的任務。一般人聽到「深」這個字眼，往往聯想到需要很「多」資料，但事實剛好相反。我們常聽到「人工智慧」=「大數據」+「深度學習」的說法，但這並不表示「因爲大數據，所以深度學習才比其他機器學習法成功」。事實是大數據可以讓所有機器學習方法都獲得提升，不獨厚深度學習。相反的，正是因為數據永遠都嫌少，所以才需要深度學習。

編者：鄭志偉，AI Office，20181015

i. **資料來源：數理人文雜誌第 10 期 什麼是深度學習？**

25. 深度學習有哪些主要的模型?

深度學習大熱以後各種模型層出不窮，很多人都在問到底什麼是 DNN、CNN 和 RNN，這麼多個網絡到底有什麼不同，作用各是什麼？

大部分神經網絡都可以用深度（depth）和連接結構（connection）來定義

1.神經網絡（Artificial Neural Networks）和深度神經網絡（Deep Neural Networks）

追根溯源的話，神經網絡的基礎模型是感知機（Perceptron），因此神經網絡 ANN 也可以叫做多層感知機（Multi-layer Perceptron），簡稱 MLP。

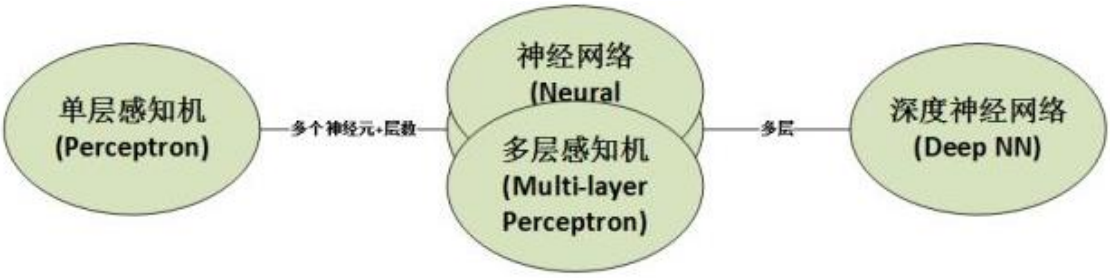


圖 25-1 深度神經網路與感知機關係

那麼多層到底是幾層？一般來說有 1-2 個隱藏層的神經網絡就可以叫做多層，準確的說是（淺層）神經網絡（Shallow Neural Networks）。隨著隱藏層的增多，更深的神經網絡（一般來說超過 5 層）就都叫做深度學習（DNN）。然而深度只是一個商業概念，很多時候工業界把 3 層隱藏層也叫做“深度學習”，所以不要在層數上太較真。在機器學習領域的約定俗成是，名字中有深度（Deep）的網絡僅代表其有超過 5-7 層的隱藏層。

神經網絡的結構指的是“神經元”之間如何連接，它可以是任意深度。以下圖的 3 種不同結構爲例，我們可以看到連接結構是非常靈活多樣的。

需要特別指出的是，卷積網絡（CNN）和循環網絡（RNN）一般不加 Deep 在名字中的原因是：它們的結構一般都較深，因此不需要特別指明深度。想對比的，自編碼器（Auto

Encoder）可以是很淺的網絡，也可以很深。

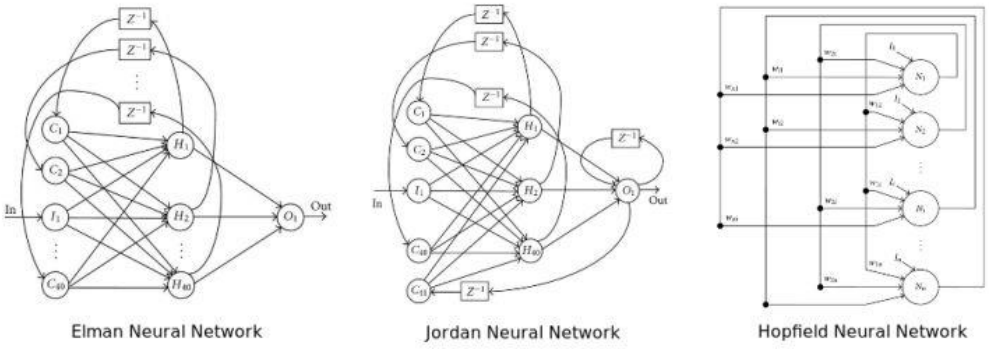


圖 25-2 神經網路的結構靈活多樣

所以你會看到人們用 Deep Auto Encoder 來特別指明其深度。全連接的前饋深度神經網絡（Fully Connected Feed Forward Neural Networks），也就是 DNN 適用於大部分分類（Classification）任務，比如數字識別等。但一般的現實場景中我們很少有那麼大的數據量來支持 DNN，所以純粹的全連接網絡應用性並不是很強。

2.循環神經網絡（Recurrent Neural Networks）和遞歸神經網絡（Recursive Neural Networks）

雖然很多時候我們把這兩種網絡都叫做 RNN，但事實上這兩種網路的結構事實上是不同的。而我們常常把兩個網絡放在一起的原因是：它們都可以處理有序列的問題，比如時間序列等。舉個最簡單的例子，我們預測股票走勢用 RNN 就比普通的 DNN 效果要好，原因是股票走勢和時間相關，今天的價格和昨天、上周、上個月都有關係。而 RNN 有“記憶”能力，可以“模擬”數據間的依賴關係（Dependency）。爲了加強這種“記憶能力”，人們開發各種各樣的變形體，如非常著名的 Long Short-term Memory（LSTM），用於解決“長期及遠距離的依賴關係”。

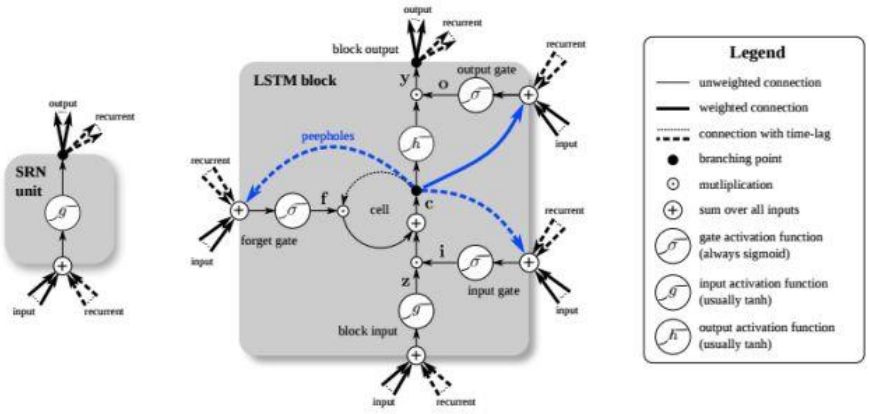


Figure 1. Detailed schematic of the Simple Recurrent Network (SRN) unit (left) and a Long Short-Term Memory block (right) as used in the hidden layers of a recurrent neural network.

圖 25-3 SRN 與 LSTM 循環神經網絡

圖 25-3 左邊的小圖是最簡單版本的循環網絡，而右邊是人們爲了增強記憶能力而開發的

LSTM。

同理，另一個循環網絡的變種 - 雙向循環網絡（Bi-directional RNN）也是現階段自然語言處理和語音分析中的重要模型。開發雙向循環網絡的原因是語言/語音的構成取決於上下文，即“現在”依託於“過去”和“未來”。單向的循環網絡僅著重於從“過去”推出“現在”，而無法對“未來”的依賴性有效的建模。

遞歸神經網絡和循環神經網絡不同，它的計算圖結構是樹狀結構而不是網狀結構。遞歸循環網絡的目標和循環網絡相似，也是希望解決數據之間的長期依賴問題。而且其比較好的特點是用樹狀可以降低序列的長度，熟悉數據結構的朋友都不陌生。但和其他樹狀數據結構一樣，如何構造最佳的樹狀結構如平衡樹/平衡二叉樹並不容易。

循環神經網路與遞歸神經網主要在語音分析，文字分析，時間序列分析。主要的重點就是數據之間存在前後依賴關係，有序列關係。一般首選 LSTM，如果預測對象同時取決於過去和未來，可以選擇雙向結構，如雙向 LSTM。

3. 卷積神經網絡（Convolutional Neural Networks）

卷積網絡早已大名鼎鼎，從某種意義上也是爲深度學習打下良好口碑的功臣。不僅如此，卷積網絡也是一個很好的計算機科學借鑒神經科學的例子。卷積網絡的精髓其實就是在多個空間位置上共用參數，據說我們的視覺系統也有相類似的模式。

首先簡單說什麼是卷積。卷積運算是一種數學計算，和矩陣相乘不同，卷積運算可以實現稀疏相乘和參數共用，可以壓縮輸入端的維度。和普通 DNN 不同，CNN 並不需要爲每一個神經元所對應的每一個輸入數據提供單獨的權重。與池化（pooling）相結合，CNN 可以被理解爲一種公共特徵的提取過程，不僅是 CNN 大部分神經網絡都可以近似的認爲大部分神經元都被用於特徵提取。

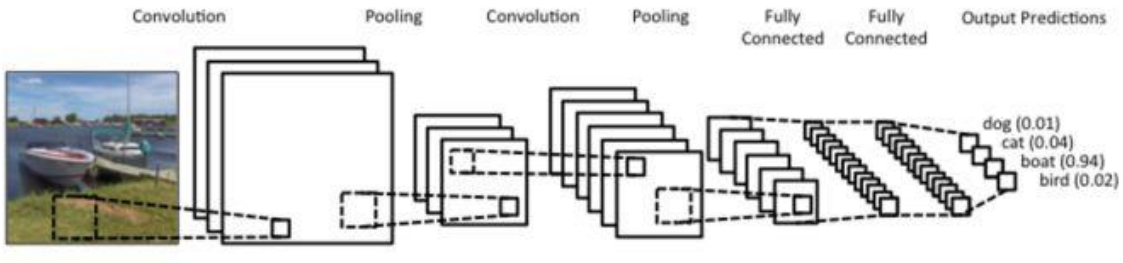


圖 25-4 常見卷積神經網路架構

以圖 25-4 爲例，卷積、池化的過程將一張圖片的維度進行了壓縮。從圖示上我們不難看出卷積網絡的精髓就是適合處理結構化數據，而該數據在跨區域上依然有關聯。

雖然我們一般都把 CNN 和圖片聯繫在一起，但事實上 CNN 可以處理大部分格狀結構化數據（Grid-like Data）。舉個例子，圖片的圖元是二維的格狀數據，時間序列在等時間上抽

取相當於一維的的格狀數據，而視頻數據可以理解爲對應視頻幀寬度、高度、時間的三維數據。

4.生成式對抗網絡（Generative Adversarial Networks）

生成式對抗網絡同時訓練兩個模型，內核哲學取自於博弈論.

簡單的說，GAN 訓練兩個網絡：1. 生成網絡用於生成圖片使其與訓練數據相似 2. 判別式網絡用於判斷生成網絡中得到的圖片是否是真的是訓練數據還是僞裝的數據。生成網絡一般有逆卷積層（deconvolutional layer）而判別網絡一般就是上文介紹的 CNN。下圖左邊是生成網絡，右邊是判別網絡，相愛相殺。

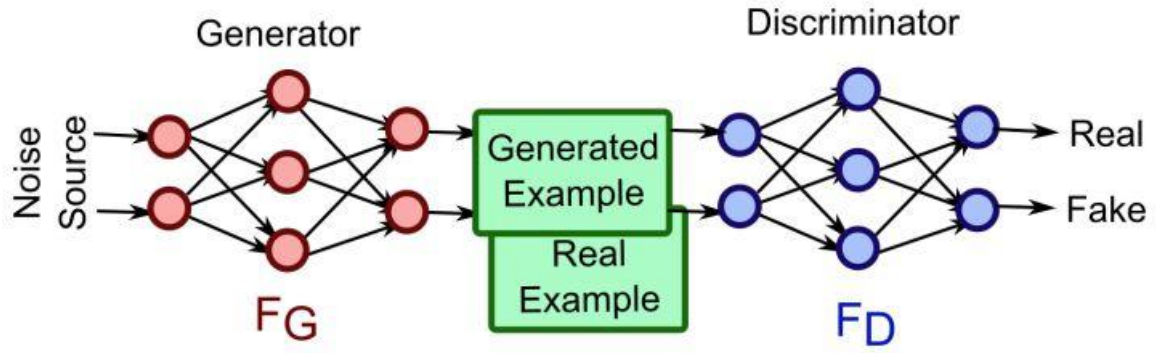


圖 25-5 生成對抗式神經網路

熟悉博弈論的人都知道零和遊戲（zero-sum game）會很難得到優化方程，或很難優化，GAN 也不可避免這個問題。但有趣的是，GAN 的實際表現比我們預期的要好，而且所需的參數也遠比按照正常方法訓練神經網絡，可以更加有效率的學到數據的分佈。

另一個常常被放在 GAN 一起討論的模型叫做變分自編碼器（Variational Auto-encoder），有興趣的讀者可以自己搜索。現階段 GAN 主要是在圖像領域比較流行，它有很大的潛力大規模推廣到聲音、視頻領域。

編者：鄭志偉，AI Office，20181015

i. 資料來源：主流的深度學習模型有哪些？

26. 有了深度學習是否還需要做特徵工程?

“深度能自動獲取特徵”只是對某些領域而言的。實際上深度學習只是能自動對輸入的低階特徵進行組合、變換，得到高階特徵。對於圖像處理之類的領域來說，圖元點就可以作爲低階特徵輸入，組合、變換得到的高階特徵也有比較好的效果，所以看似可以自動獲取特徵。

在其他領域的情況就不是這樣了。例如自然語言處理中，輸入的字或詞都是離散、稀疏的值，不像圖片一樣是連續、稠密的。輸入原始數據進行組合、變換得到的高階特徵並不是那麼有效。而且有的語義並不來自數據，而來自人們的先驗知識，所以利用先驗知識構造的特

徵是很有幫助的。

所以在深度學習中，原來的特徵選擇方法仍然適用。不過方便的一點是神經網絡能對特徵自動進行排列組合，所以只要輸入一階特徵就行，省去了手動構造高階特徵的工作量。

編者：鄭志偉，AI Office，20181015

i. 資料來源：深度學習如何做特徵工程？

27. 有哪些深度學習常見的啟動函數 Activation Function?

在類神經網路中如果不使用啟動函數，那麼在類神經網路中皆是以上層輸入的線性組合作爲這一層的輸出（也就是矩陣相乘），輸出和輸入依然脫離不了線性關係，做深度類神經網路便失去意義。啟動函數實質是非線性方程。常見的啟動函數選擇有 sigmoid、tanh、ReLU、Leaky ReLU、Maxout，實用上最常使用 ReLU 。在卷積神經網路 CNN 的卷積層中，常用的啟動函數是 ReLU，在迴圈神經網路 RNN 中，常用的是 tanh 或者是 ReLU。

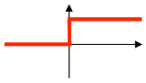
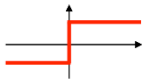
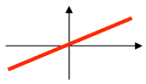
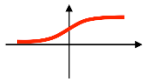
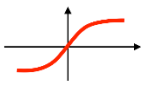
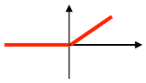
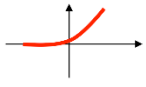
Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

圖 27-1 深度學習常見的啟動函數

編者：鄭志偉，AI Office，20181015

i. 資料來源：深度學習：使用激勵函數的目的、如何選擇激勵函數

ii. 資料來源：Neural Network Activation Function

28. 深度學習中啟動函數的選擇為何 ReLU 常勝出？

深度學習領域 Relu 激勵函數蔚爲主流，主要考量的因素有以下幾點：

梯度消失問題（vanishing gradient problem）：ReLU 的分段線性性質能有效的克服梯度消失的問題。對使用反向傳播訓練的類神經網絡來說，梯度的問題是最重要的，使用 sigmoid 和 tanh 函數容易發生梯度消失問題，是類神經網絡加深時主要的訓練障礙。具體的原因是這兩者函數在接近飽和區（如 sigmoid 函數在 [-4, +4] 之外），求導後趨近於 0，也就是所謂梯度消失，造成更新的訊息無法藉由反向傳播傳遞。

類神經網路的稀疏性：Relu 會使部分神經元的輸出爲 0，可以讓神經網路變得稀疏，緩解過度擬合的問題。但衍生出另一個問題是，如果把一個神經元停止後，就難以再次開啓（Dead ReLU Problem），因此又有 Leaky ReLU 類（x<0 時取一個微小值而非 0），maxout（增加激勵函數專用隱藏層，有點暴力）等方法，或使用 adagrad 等可以調節學習率的演算法。

生物事實-全有全無律：在神經生理方面，當刺激未達一定的強度時，神經元不會興奮，因此不會產生神經衝動。如果超過某個強度，才會引起神經衝動。Relu 比較好的捕捉了這個生物神經元的特徵。

計算量節省：ReLU 計算量小，只需要判斷輸入是否大於 0，不用指數運算。

編者：鄭志偉，AI Office，20181015

i. **資料來源：**深度學習：使用激勵函數的目的、如何選擇激勵函數

29. 什麼是梯度消失（Gradient Vanishing）？

梯度消失問題是一種在使用梯度下降法（Gradient descent ）和反向傳播演算法（Back propagation）訓練人工神經網絡時發現的難題。在這類訓練方法的每個疊代中，神經網絡權重的更新值與誤差函數梯度成比例，然而在某些情況下，梯度值會幾乎消失，使得權重無法得到有效更新，甚至神經網絡可能完全無法繼續訓練。

神經網路更多的層數讓網絡更能夠刻畫現實世界中的複雜情形。理論上而言，參數越多的模型複雜度越高，“容量”也就越大，也就意味著它能完成更複雜的學習任務。多層感知機給我們帶來的啓示是，神經網絡的層數直接決定了它對現實的刻畫能力——利用每層更少的神經元擬合更加複雜的函數。但是隨著神經網絡層數的加深，優化函數越來越容易陷入局部最優解（即過擬合，在訓練樣本上有很好的擬合效果，但是在測試集上效果很差），並且這個“陷阱”越來越偏離真正的全域最優。利用有限數據訓練的深層網絡，性能還不如較淺層網絡。同時，另一個不可忽略的問題是隨著網絡層數增加，“梯度消失”（或者說是梯度發散 diverge）現象更加嚴重。具體來說，我們常常使用 sigmoid 作爲神經元的輸入輸出函

數。對於幅度爲 1 的信號，在 BP 反向傳播梯度時，每傳遞一層，梯度衰減爲原來的 0.25。層數一多，梯度指數衰減後低層基本上接受不到有效的訓練信號。梯度發散會不利於學習，梯度一旦發散就等價於淺層網絡的學習，那麼使用深度學習就沒有任何優勢。

編者：鄭志偉，AI Office，20181015

i. 資料來源：詳解梯度爆炸和梯度消失

ii. 資料來源：梯度消失，梯度爆炸

iii. 資料來源：梯度爆炸和梯度消失的本質原因

30. 什麼是深度學習的超參數?

所謂超參數，就是機器學習模型裏面的框架參數，比如聚類（分群）方法裏面類的個數，或者話題模型裏面話題的個數等等，都稱爲超參數。它們跟訓練過程中學習的參數（權重）是不一樣的，通常是手工設定，不斷試錯調整，或者對一系列窮舉出來的參數組合一通枚舉（叫做網格搜索）。深度學習和神經網路模型，有很多這樣的參數需要學習，這就是為什麼過去這麼多年從業者棄之不顧的原因。以前給人的印象，深度學習就是“黑魔法”。時至今日，非參數學習研究正在幫助深度學習更加自動的優化模型參數選擇，當然有經驗的專家仍然是必須的。深度學習的參數就是模型可以根據資料可以自動學習出的變數。比如權重，偏差等。超參數就是用來確定模型的一些參數，超參數不同，模型就是有微小的區別，比如假設都是 CNN 模型，如果層數不同，模型不一樣。超參數一般就是根據經驗確定的變數。在深度學習中，超參數有：學習速率，反覆運算次數，層數，每層神經元的個數等等。

編者：鄭志偉，AI Office，20181015

i. 資料來源：什麼是超參數

ii. 資料來源：參數（parameters）和超參數（hyperparameters）

31. 目前深度學習的發展趨勢有哪些?

神經網絡的基本思想是模擬計算機「大腦」中多個相互連接的細胞，這樣它就能從環境中學習，識別不同的模式，進而做出與人類相似的決定。

下面將就神經網絡與深度學習發展的幾大重要趨勢進行討論：

1.膠囊網絡（Capsule Networks）：Geoffery Hinton 等人最近關於膠囊網絡（Capsule networks）的論文在機器學習領域造成相當震撼的影響。它提出了理論上能更好地替代卷積神經網絡的方案，是當前計算機視覺領域的最新技術。膠囊網絡是一種新興的深層神經網絡，其處理資訊的方式類似於人腦。

膠囊網絡與卷積神經網絡相反，雖然卷積神經網絡是迄今爲止應用最廣泛的神經網絡之一，

但其未能考慮簡單對象和複雜對象之間存在的關鍵空間層次結構。這導致了誤分類並帶來了更高的錯誤率。在處理簡單的識別任務時，膠囊網絡擁有更高的精度，更少的錯誤數量，並且不需要大量的訓練模型數據。

2.深度強化學習（Deep Reinforcement Learning）：深度強化學習是神經網絡的一種形式，它的學習方式是通過觀察、行動和獎勵，與周圍環境進行交互。深度強化學習已經被成功地用於遊戲策略的制定，如 Atari 和 Go。AlphaGo 擊敗了人類冠軍棋手，是深度強化學習最為著名的應用。

3.數據增強（Lean and augmented data learning）：到目前為止，機器學習與深度學習遇到的最大挑戰是：需要大量使用帶標籤的數據來訓練系統。目前有兩種應用廣泛的技巧可以幫助解決這個問題：合成新的數據、遷移學習

「遷移學習」，即把從一個任務或領域學到的經驗遷移到另一個任務或領域，「一次學習」指遷移學習應用到極端情況下，在只有一個相關例子，甚至沒有例子的情況下學習。由此它們成爲了「精簡數據」的學習技巧。與之相仿，當使用模擬或內插合成新的數據時，它有助於獲取更多的訓練數據，因而能夠增強現有數據以改進學習。

通過運用上述技巧，我們能夠解決更多的問題，尤其是在歷史數據較少的情況下。

4.監督模型（Supervised Model）：監督模型時一種學習形式，它根據預先標記的訓練數據學到或建立一個模式，並依此模式推斷新的實例。監督模型使用一種監督學習的演算法，該演算法包括一組輸入和標記正確的輸出。

將標記的輸入與標記的輸出進行比較。給定兩者之間的變化，計算一個誤差值，然後使用一個演算法來學習輸入和輸出之間的映射關係。

5. 網絡記憶模型（Networks With Memory Model）：人類和機器的一個典型區別在於工作和嚴謹思考的能力。我們可以對計算機進行編程，使其以極高的準確率完成特定的任務。但是如果我們想要它在不同的環境中工作，還有需要解決很多問題。要想使機器適應現實世界的環境，神經網絡必須能夠學習連續的任務且不產生「災難性忘卻（catastrophic forgetting）」，這便需要許多方法的幫助，如：

長期記憶網絡（Long-Term Memory Networks）：它能夠處理和預測時間序列

彈性權重鞏固演算法（Elastic Weight Consolidation）：該方法能夠選擇性地減慢對這些任務而言比較重要的權重的學習速率

漸進式神經網絡（Progressive Neural Networks）：不會產生「災難性忘卻」，它能夠從已經學會的網絡中提取有用的特徵，用於新的任務

6.混合學習模式（Hybrid Learning Models）：不同類型的深度神經網絡，例如生成式對抗網絡（GANs）以及深度強化學習（DRL），在性能提升和廣泛應用方面展現了了巨大的潛力。不過，深度學習模型不能像貝葉斯機率那樣爲不確定性的數據場景建模。

混合學習模式結合了這兩種方法的優勢，典型的混合學習模式包括貝葉斯生成對抗網絡（Bayesian GANs）以及貝葉斯條件生成對抗網絡（Bayesian Conditional GANs）。

混合學習模式將商業問題的範圍擴大，使其能夠解決具有不確定性的深度學習問題，從而提高模型的性能，增強模型的可解釋性，實現更加廣泛的運用

編者：鄭志偉，AI Office，20181015

i. **資料來源：深度學習與神經網絡：最值得關注的 6 大趨勢**

32. AI 深度學習開發工具有哪些?

許多人準備開始進入機器學習尤其是深度學習世界時，最常問的問題就是到底要挑選哪一種框架來入門？有如圖 32-1、表 32-1 所示，從最大的開源社群 Github 上，就可找到二十種星星數超過一千的深度學習框架，包括 TensorFlow、Keras、Caffe、PyTorch、CNTK、MXNet、DL4J、Theano、Torch7、Caffe2、Paddle、DSSTNE、tiny-dnn、Chainer、neon、ONNX、BigDL、DyNet、brainstorm、CoreML 等，而排名第一名的 TensorFlow 更有近十萬個星星。由此可知，深度學習非常受到大家重視，當選擇深度框架時可以從好幾個面向來考慮。

圖 32-1 常見深度學習框架

框架名稱	組織	API支援語言	Star	Fork	參考網址
Tensorflow	Google	C++, Python, GO, Java	98,120	62,206	https://github.com/tensorflow/tensorflow
Keras	François Chollet	Python	28,880	10,732	https://github.com/keras-team/keras
Caffe	BVLC	C++, Python, Matlab	23,939	14,649	https://github.com/BVLC/caffe
PyTorch	Adam Paszke	Python	14,692	3,290	https://github.com/pytorch/pytorch
CNTK	Microsoft	C++, C#, Python, Java	14,345	3,819	https://github.com/Microsoft/CNTK
MXNet	DMLC	C++, Scala, R, JS, Python, Julia, Matlab, Go	13,816	5,120	https://github.com/apache/incubator-mxnet
DL4J	DeepLearning4Java	Java, Scala	8,807	4,216	https://github.com/deeplearning4j/deeplearning4j
Theano	University of Montreal	Python	8,175	2,454	https://github.com/Theano/Theano
Torch7	Facebook	Lua	7,866	2,276	https://github.com/torch/torch7
Caffe2	Facebook	C++, Python	7,849	1,914	https://github.com/caffe2/caffe2
Paddle	Baidu(百度)	C++, Python	6,818	1,853	https://github.com/PaddlePaddle/Paddle
DSSTNE	Amazon	C++	4,098	676	https://github.com/amzn/amazon-dsstne
tiny-dnn	tiny-dnn	C++	4,044	1,075	https://github.com/tiny-dnn/tiny-dnn
Chainer	Chainer	Python	3,727	982	https://github.com/chainer/chainer
neon	Nervana Systems	Python	3,476	793	https://github.com/NervanaSystems/neon
ONNX	Microsoft	Python	3,251	392	https://github.com/onnx/onnx
BigDL	Intel	Scala	2,431	548	https://github.com/intel-analytics/BigDL
DyNet	Carnegie Mellon University	C++, Python	2,248	540	https://github.com/clab/dynet
brainstorm	IDSIA	Python	1,275	154	https://github.com/IDSIA/brainstorm
CoreML	Apple	Python	1,032	97	https://github.com/apple/coremltools

表 32-1 深度學習框架比較表

編者：鄭志偉，AI Office，20181015

i. **資料來源：深度學習，從「框架」開始學起**

ii. **資料來源：開發者如何挑選最合適的機器學習框架？**

33. 開發者如何挑選最合適的深度學習框架？

深度學習非常受到大家重視，選擇深度框架時可以從以下面向來考慮：

1.程式語言：首先是開發時所使用的程式語言，如果要執行「訓練」及「推論」效率好些，則可能要用 C++。若要上手容易、支援性強，則要考慮 Python，從（表格一）中可看出有 3 / 4 的框架都支援 Python。

若習慣使用 Java 開發程式，那大概就只有 TensorFlow、DL4J 和 MXNet 可選了。目前有一些框架（如 TensorFlow、Caffe 等）底層是 C++，應用層 API 是用 Python，這類框架能取得不錯的開發及執行效率。

若想改善底層效率時，還要考慮依不同硬體，學會 OpenCL 或 Nvidia 的 CUDA / cuDNN 等平行加速程式寫法。

2.執行平臺：再來考慮的是可執行的作業系統及硬體（CPU、GPU）平臺，目前大多數的框架都是在 Linux CPU+GPU 的環境下執行，部份有支援單機多 CPU（多執行緒）或多 GPU 協同計算以加速執行時間，甚至像 TensorFlow、CNTK、DL4J、MXNet 等框架還有支援叢集（Cluster）運算。

許多雲端服務商（Google、Amazon、Microsoft 等）也是採取這類組合，方便佈署開發好的應用程式。另外也有些非主流的框架（如：tiny-dnn）只支援 CPU 而不支援 GPU，這類框架的好處就是移植性較強，但工作效率很差，較適合小型系統。

如果是用 Windows 的人，英特爾（Intel）及微軟（Microsoft）也分別有提供 BigDL 及 CNTK，若使用 Mac 系統則要考慮使用 CoreML。不過最近 Google 的 TensorFlow 爲了吃下所有的市場，已經可以支援 Linux、Windows、Mac 甚至是 Android，因此成為開源排行榜第一名。

3.模型支援：目前常見的深度學習模型包含監督型（如 CNN）、時序型（如 RNN / LSTM）、增強學習（如 Q-Learning）、轉移學習、對抗生成（GAN）等，但不是每個框架都能全部支援。

舉例來說：老牌的 Caffe 適合做監督型學習，尤其是圖像辨識等應用，但對於時序型學習就不太合適，遇到對抗生成模型時就更使不上力。若不清楚那些框架可支援的模型類型，可參考表 32-1 的對應網址，前往各官網瞭解使用上的限制。

4.框架轉換：如果遇到需要串接不同平臺或框架時，則要考慮選用具有提供跨框架功能。目前有幾大陣營，像 Keras 可支援 TensorFlow、Theano、MXNet、DL4J、CNTK，而微軟和臉書聯盟推的 ONNX 可支援 Caffe2、CNTK、PyTorch 等，但 Google 的 TensorFlow 卻不願加入該聯盟。另外，雖然框架之間可以轉換，但不代表轉換後的執行效率會和直接使用某個框架一樣好，此時只能依實際需求來取捨是否拿彈性換取效能。

5.社群支援：最後要考慮選用的框架社群是否活躍，是否很久沒有維護（升級），甚至被預告即將淘汰，像 Theano 雖然功能強大也有很多人在用，但目前確定已不再更新版本，因此建議不要再跳坑了。另外對于英文不好的朋友，選用框架的社群討論區、文字教程、操作視頻等是否有中文支援也是很重要的，以免遇到問題不知向誰求救。

編者：鄭志偉，AI Office，20181015

i. **資料來源：深度學習，從「框架」開始學起**

ii. **資料來源：開發者如何挑選最合適的機器學習框架？**

34. 什麼是機器學習自動建模 Auto ML?

Google 首席執行官 Sundar Pichai 和 Google AI 負責人 Jeff Dean 宣稱，神經網絡結構搜索及其所需的大量計算能力對於機器學習的大眾化至關重要。科技媒體爭相報道了 Google 在神經網絡結構搜索方面的工作。

在 2018 年 3 月舉辦的 TensorFlow DevSummit 大會上，Jeff Dean 在主題演講中宣稱，未

來穀歌可能會用 100 倍的計算能力取代機器學習專家。 他給出了需要龐大計算力的神經網絡結構搜索作爲例子（他給出的唯一例子），來説明為什麼我們需要 100 倍的計算能力才能使更多人能夠使用 ML。

傳統上，術語 AutoML 用於描述模型選擇和/或超參數優化的自動化方法。這些方法適用於許多類型的演算法，例如隨機森林，梯度提升機器（gradient boosting machines），神經網絡等。 AutoML 領域包括開源 AutoML 庫，研討會，研究和比賽。 初學者常常覺得他們在爲模型測試不同的超參數時通常僅憑猜測，而將這部分過程的自動化可以使機器學習變得更加容易。即使是對經驗豐富的機器學習從業者而言，這一自動化過程也可以加快他們的速度。

業內現存有許多 AutoML 庫，其中最早出現的是 AutoWEKA，它於 2013 年首次發布，可以自動選擇模型和超參數。其他值得注意的 AutoML 庫包括 auto-sklearn（將 AutoWEKA 拓展到了 python 環境），H2O AutoML 和 TPOT。 AutoML.org（以前被稱為 ML4AAD，Machine Learning for Automated Algorithm Design）小組，自 2014 年以來一直在 ICML 機器學習學術會議上組織 AutoML 研討會。

編者：鄭志偉，AI Office，20181015

i. **資料來源：AutoML 和神經架構搜索初探**

35. AutoML 有用嗎？

AutoML 提供了一種選擇模型和優化超參數的方法。它還可以用於獲取對於一個問題可能性能的基準結果。這是否意味著數據科學家將被取代？ 並非如此，因為我們知道，機器學習從業者還有許多其他事情要做。

對於許多機器學習項目，選擇模型不過是構建機器學習產品複雜過程中的一部分，如果參與者不瞭解項目各個部分是如何相互關聯的，那麼項目必然會失敗。過程中可能會涉及的 30 多個不同步驟，機器學習（特別是深度學習）中最耗時的兩個方面是清理數據（這是機器學習中不可或缺的一部分）和訓練模型。 雖然 AutoML 可以幫助選擇模型並選擇超參數，但重要的是，我們仍然要理清有哪些數據科學的技能是需要的以及那些仍未解決的難題。

編者：鄭志偉，AI Office，20181015

i. **資料來源：AutoML 和神經架構搜索初探**

36. 什麼是神經網路結構搜索 Neural Architecture Search?

神經網路結構搜索。Google 首席執行 Sundar Pichai 在博客中寫道： “設計神經網路是非常耗時的，其對專業知識的極高要求使得只有小部分科研人員和工程師才能參與設計。這就

是我們創建 AutoML 方法的原因，有了它，神經網路也可以設計神經網路。”

Pichai 所說的使用 “神經網路也可以設計神經網路” 就是神經網路結構搜索; 通常使用強化學習或進化演算法來設計新的神經網路網路結構。這很有用，因為它使得我們能夠發現比人們想像的要複雜得多的網路結構，並且這些網路結構可以針對特定目標進行優化。神經網路結構搜索通常需要大量計算力。確切地說，神經網路結構搜索通常涉及學習像層（通常稱爲 “單元” ）之類的東西，它們可以組裝成一堆重複的單元來創建神經網路。

在 2017 年 5 月舉行的 Google I / O 大會上，Google AI 研究人員發布了他們的研究成果，AutoML 這一術語迅速成為主流。神經結構搜索的學術論文很多，不在此探討。

神經網路結構搜索是 AutoML 領域的一部分，該領域關注的核心問題是：我們如何將模型選擇和超參數優化過程自動化？ 然而，自動化忽視了人類參與的重要作用。

但是，選擇模型只是構建機器學習產品複雜過程中的一部分。在大多數情況下，網路結構選擇遠不是最難，最耗時或最重要的問題。目前，沒有證據能夠證明每個問題最好用它自己獨特的網路結構來建模，大多數從業者都認爲不太可能會出現這種情況。

Google 等組織致力於網路結構設計，並與其他人共用他們發現的網路結構。這樣的服務是非常重要且有用的。 然而，只有那些致力於基礎神經結構設計的小部分研究人員才需要基礎架構搜索方法。 我們其他人可以通過遷移學習來利用他們找到的網路結構。

人類和電腦如何協同工作才能使機器學習效率更高？ 增強機器的重點在於弄清楚人和機器應如何更好地協同工作以發揮他們的不同優勢。增強機器學習的一個例子是 Leslie Smith 的學習率查詢器，它可在 fastai 庫（在 PyTorch 之上運行的高級 API）中實現。 學習率是一個超參數，可以確定模型訓練的速度，甚至可以確定模型是否訓練成功。 學習速率查詢器允許人類通過查看生成的圖表中找到合適的學習速率。它比 AutoML 更快地解決了這一問題，增強了資料科學家對訓練過程的理解，並鼓勵採用更強大的多步驟方法來訓練模型。

專注於自動化超參數選擇存在的另一個問題是：它忽視了某些類型的模型可能適用性更廣，需要調整的超參數更少以及對超參數選擇不太敏感的情況。例如，隨機森林優於梯度提升機器（Grandient Boosting Machine, GBM）的重要一點是隨機森林更加穩定，而 GBM 往往對超參數的微小變化相當敏感。因此，隨機森林在業內得到廣泛應用。研究有效刪除超參數的方法（通過更智慧的預設值或通過新模型）將產生巨大的影響。

編者：鄭志偉，AI Office，20181015

i. **資料來源：AutoML 和神經架構搜索初探**

37. 什麼是遷移學習 Transfer Learning?

遷移學習利用預訓練模型，可以讓人們用少量資料集或者較少的計算力得到頂尖的結果，是一種非常強大的技術。預訓練模型此前會在相似的、更大的資料集上進行訓練。由於模型無需從零開始學習，它可以比那些用更少數據和計算時間的模型得到更精確的結果。

遷移學習是項重要的技術，很多世界 500 強公司都用到了這種技術。雖然遷移學習看起來不如神經架構搜索那麼“性感”，但是它卻創造過很多學術界進步的成果，例如 Jeremy Howard 和 Sebastian Ruder 將遷移學習應用到 NLP 中，在 6 個資料集上達到了最佳分類效果，同時也成爲了 OpenAI 在這一領域的研究基礎。

神經架構搜索 vs 遷移學習：兩種不同的方法。遷移學習之下的基礎理念是，神經網路結構會對相同種類的問題進行泛化：例如，很多圖片都有基礎特徵（比如角度、圓圈、狗狗的臉、車輪等等），這些特徵構成了圖片的多樣性。相比之下，神經架構搜索解決問題的基本理念恰恰相反：每個資料集都有一個獨特的，高度專門化的架構。

圖 37-1 圖像分類器習得的 4 個特徵：角、圓圈、狗臉和輪子

當神經架構搜索發現了一種新結構，你必須從零開始學習該結構的權重。但是有了遷移學習，你可以從預訓練模型上已有的權重開始訓練。這也意味著你無法在同一個問題上同時使用遷移學習和神經架構搜索：如果你要學習一種新的結構，你可能需要為此訓練一個新權重；但如果你用遷移學習，可能無需對結構進行實質性改變。

當然，你可以將遷移學習運用到一個經過神經架構搜索的結構上。這只需要幾個研究者用神經架構搜索和開源的模型即可。如果可以用遷移學習，並不是所有機器學習從業者都要在問題上使用神經架構搜索，然而，Jeff Dean、Sundar Pichai 以及 Google 和媒體的報導都表示：每個人都應該直接用神經架構搜索。

編者：鄭志偉，AI Office，20181015

i. **資料來源：AutoML 和神經架構搜索初探**