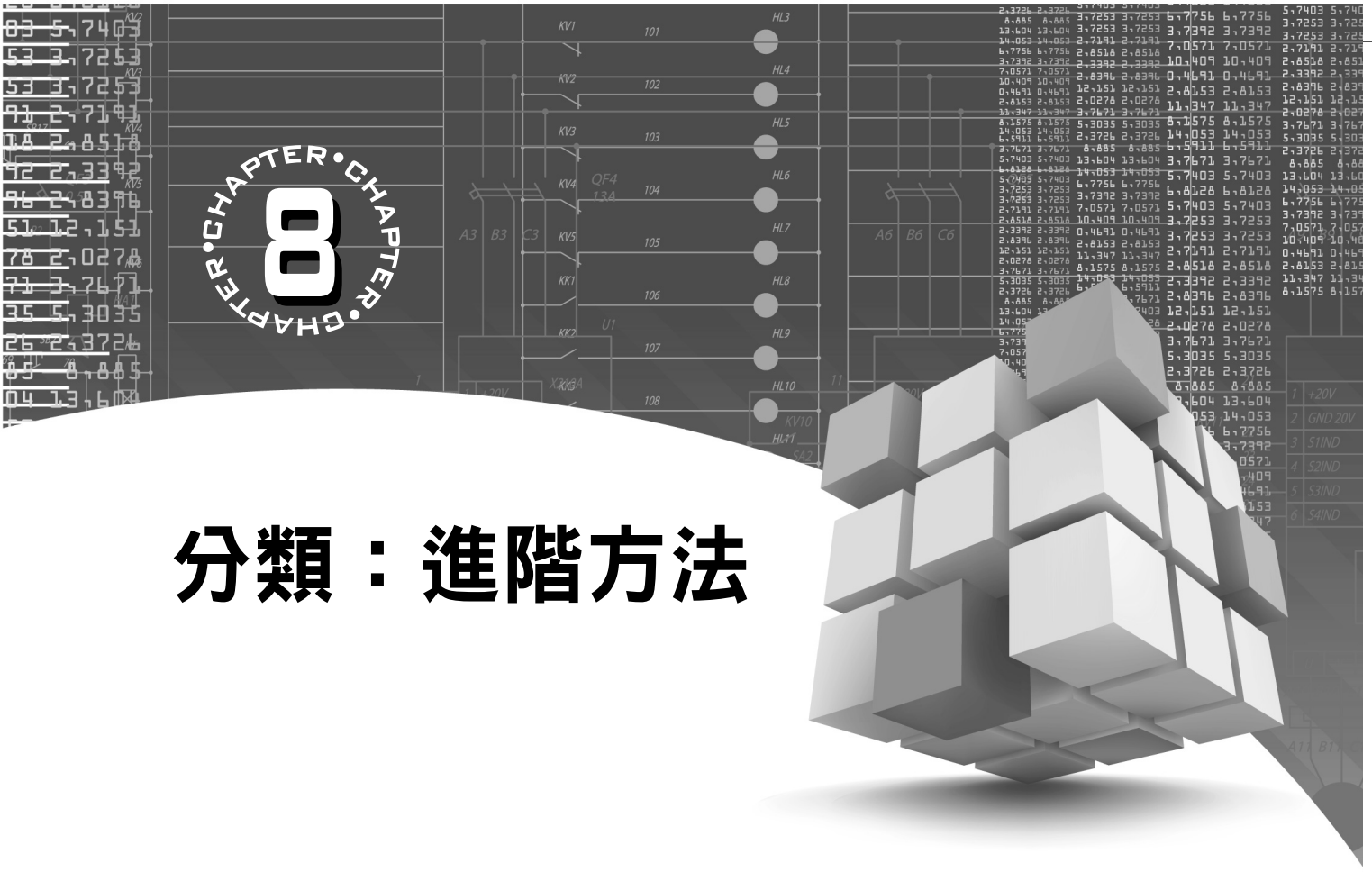


CHAPTER 8

分類：進階方法

在本章，讀者將學習資料分類的進階技術，我們首先從**貝式信念網路** (Bayesian belief network) 開始，不同於樸素貝式分類器，貝式信念網路並不假設類別條件獨立是成立的。**倒傳遞** (backpropagation) 類神經網路演算法將在 8.2 節介紹，類神經網路 (neural network) 是由一組相互連結的輸入 / 輸出單元所組成的，單元與單元的連結上皆對應一個權重值，在學習階段，藉由調整網路連結上的權重值來幫助類神經網路正確地預測輸入資料值組的類別標籤。最近熱門的分類技術 —— **支持向量機** (support vector machine) 將在 8.3 節介紹，支持向量機將訓練資料投影到一個高維度的特徵空間，並找出一個最佳的超平面 (hyperplane) 來切割兩個類別的資料，此超平面使用訓練資料中重要的值組 (稱為支持向量) 來建構的。第 8.4 節介紹使用**頻繁樣式的分類法**，此方法基於探勘出頻繁出現的資料中，其屬性與屬性值配對之間的關聯性，以進行資料分類，此方法建構在第 5 與 6 章所介紹的頻繁樣式探勘的研究基礎上。

Chapter 8

第 8.5 節介紹**懶惰學習法** (lazy learner) 或稱**例證式** (instance-based) 學習的分類法，例如最鄰近分類法 (nearest neighbor classifier，也稱作最近鄰居分類法) 與案例式推理分類 (case-based reasoning classifier)，它們將所有訓練資料值組儲存在樣式空間中，並且靜待著測試值組的出現以進行分類推理。

8.1 貝氏信念網路

延續在第 7 章所介紹貝式定理與樸素貝式分類器，本節我們探討的貝式信念網路 (Bayesian belief network) 是一種機率圖形模型 (probabilistic graphical model)，不同於樸素貝式分類器，它允許屬性間出現相依關係，並可用來分類資料。8.1.1 節介紹貝式信念網路的基本概念，在 8.1.2 節將會學習如何訓練貝式信念網路的機率圖形模型。

8.1.1 概念與機制

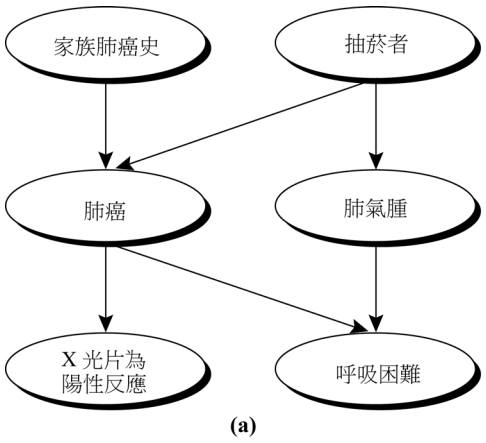
樸素貝氏分類建立類別條件獨立 (class conditional independence) 的假設，也就是說，給定一個對應類別標籤的資料值組，其屬性的值與其它屬性的值假設為條件獨立的，此假設簡化了計算的複雜度，如果這個假設是成立的，則樸素貝氏分類法會是最正確的分類法，然而，實際上變數間存在著相互依賴的關係。**貝氏信念網路**明確指定聯合條件機率分佈 (joint conditional probability distribution)，它允許我們定義變數間的類別條件相依的關係，並提供一個可學習的圖形式模型來表達這種因果關係，而訓練過後的貝氏信念網路可以用來進行分類。貝氏信念網路也稱為**信念網路** (belief networks)、**貝氏網路** (Bayesian networks) 或**機率網路** (probabilistic networks)，為了敘述簡便，這裡我們使用信念網路這個名詞。

信念網路包含兩個元件：一個有向非循環圖 (directed acyclic graph) 與一組條件機率表格 (conditional probability tables)，如圖 8.1 所示，在此機率圖形模型中，每個節點代表一個隨機變數，此變數可以是離散值或連

續值的，它們可以對應到資料中真實的屬性，或著是相信具有某種關係的“隱藏變數”(hidden variable)，以醫學資料為例，隱藏變數可以表示某疾病的特殊症狀與併發症。每個箭頭代表機率相依的關係，如果有一個箭頭從節點 Y 指向節點 Z，則稱 Y 為 Z 的**父節點** (parent) 或**立即先行者** (immediate predecessor)，而稱 Z 為 Y 的**子孫節點** (descendant)，在給定它的父節點的條件下，每一個節點是條件獨立於它所有的非子孫節點。

圖 8.1 是一個簡單的信念網路，圖中 6 個節點對應到 6 個布林變數，箭頭代表已知的因果關係，舉例來說，是否罹患肺癌會受到此人的「家族肺癌史」所影響，同時也會受到此人是否為「抽菸者」所影響。請注意，給定我們得知病人罹患肺癌的條件下，那變數「X 光片為陽性反應」是獨立於「家族肺癌史」或「抽菸者」的，換句話說，當得知「肺癌」變數的結果，那麼「家族肺癌史」或是「抽菸者」並不會為變數「X 光片為陽性反應」提供額外的訊息。依據這些箭頭，我們也可知道在給定「肺癌」變數的父節點「家族肺癌史」或「抽菸者」的條件下，「肺癌」是條件獨立於「肺氣腫」。

信念網路中每一個變數皆附屬一個**條件機率表** (conditional probability table, CPT)，變數 Y 的 CPT 明確指示條件分佈 $P(Y \mid Parents(Y))$，其中



	FH, S	*FH, ~S*	*~FH, S*	*~FH, ~S*
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

(b)

圖 8.1 一個簡單的貝氏信念網路：(a) 用有向非循環圖代表所提出的因果模型；(b) 此條件機率表顯示變數“肺癌史” (*LC*) 與他的父節點 “家族肺癌史”(*FH*) 與“抽菸者”(*S*) 之間，在所有可能的排列組合上的條件機率值。

Parents(*Y*) 為 *Y* 的父節點。圖 8.1 (b) 顯示「肺癌」變數的 CPT，「肺癌」變數在給定父節點「家族肺癌史」或「抽菸者」所有可能的排列組合下的條件機率值皆顯示在 CPT 中。舉例來說，CPT 左上角與右下角儲存格告訴我們：

$$P(\text{肺癌} = \text{是} \mid \text{個人家族肺癌史} = \text{是}，\text{抽菸者} = \text{是}) = 0.8$$
$$P(\text{肺癌} = \text{否} \mid \text{個人家族肺癌史} = \text{否}，\text{抽菸者} = \text{否}) = 0.9$$

令 $X = (x_1, \ldots, x_n)$ 為由 n 個屬性 (變數) $Y_1, \ldots, Y_n$ 所描述的資料值組。回顧在給定它的父節點的條件下，每一個節點是條件獨立於它所有的非子孫節點，這性質使得信念網路得以使用下面公式來計算聯合條件機率分佈：

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid Parents(Y_i)) \qquad \textbf{(8.1)}$$

其中 $P(x_1, \ldots, x_n)$ 為 X 在特定的值時之機率值，$P(x_i \mid Parents(Y_i))$ 的條件機率值則可以查詢 Y_i 的 CPT 得知。

網路中的一個節點可選定作為輸出節點，它代表類別屬性標籤 (輸出節點的數目也可以超過一個)。信念網路在分類時，它不是僅傳回資料的類別標籤，它回傳一個機率分佈函數來指定資料是屬於各個類別的機率值。

8.1.2 貝式信念網路的訓練方法

「如何訓練信念網路？」在學習信念網路時有許多情況可能發生。網路拓樸結構 (topology，網路中節點與連結線的佈局) 可以由人類專家建立，或是從資料推論得到。網路中的變數分為在訓練資料中可明確觀察到的 (observable) 或是隱藏的 (hidden)，這種隱藏的資料被稱為遺失值 (missing values) 或不完全資料 (incomplete data)。

如果網路拓樸為未知的，而所有變數為明確時，有許多種方法可在給定網路中變數的情況下學習信念網路的拓樸結構，這是一種離散最佳化

(discrete optimization) 的問題。人類專家通常會對被分析的領域中特定變數間的條件相依性有很好的理解，這可以幫助網路拓樸的設計。領域專家必須明確指明有參與直接相依的變數間的條件機率值，這些機率值可以用來計算剩餘的機率值。

如果網路拓樸為已知的，且所有變數皆為明確時，則信念網路的訓練是相當直接明確的，它僅需要估計 CPT 儲存格中的那些機率值，就如同樸素貝式分類器中估計機率值的做法一樣。

如果網路拓樸為已知的，但其中有某些變數為隱藏 (hidden) 時，許多種方法可用來訓練信念網路，我們將介紹其中最有作為的梯度降低 (gradient descent) 演算法。對那些沒有進階數學知識的讀者，此方法所用繁複的微分公式可能看來怪嚇人的，不過已有套裝軟體來幫忙求解這些方程式，而且其基本的想法是滿容易理解的。

令 D 為訓練資料集，其中包含的值組有 $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_{|D|}$，訓練信念網路亦即代表我們必須學習到 CPT 表中每個儲存格內的機率值。令 w_{ijk} 為變數 $Y_i = y_{ij}$ 的 CPT 中，對應到 Y_i 的父節點變數為 $U_i = u_{ik}$ 的 CPT 儲存格內的機率值，其中 $w_{ijk} = P(Y_i = y_{ij} \mid U_i = u_{ik})$。舉例來說，假設 w_{ijk} 為圖 8.1 (b) 中的 CPT 的左上角項目，則 Y_i 為變數「肺癌」，而 y_{ij} 為變數的值 $\{yes\}$。U_i 為 Y_i 的父節點變數，共包含 { 家族肺癌史，抽菸者 }，而 u_{ik} 為父節點變數的值 {yes, yes}。w_{ijk} 也可被視為權重值，如同在 8.2 節介紹的類神經網路中隱藏節點的權重一樣。這些權重值整體可用權重矩陣 $\boldsymbol{W}$ 表示。初始時，這些權重值為隨機設定的機率值，梯度降低 (gradient descent) 演算法使用貪婪式登山法 (greedy hill-climbing) 的策略，在每一次疊代中，我們更新權重值，並且最終會收斂至一個區域最佳解 (local optimum solution)。

梯度降低法可用來尋找最佳的 w_{ijk} 值以建構最符合給定資料的機率模型，它疊代式地往準則函數 (criterion function) 梯度為負的方向 (亦即，逐步降梯度) 尋找新的解，也就是說，我們想找到一組 W 能夠最大化準則函數。初始時，權重被設定為隨機的機率值，接下來使用貪婪式登山法的策略，在每一步 (疊代) 它往目前看似最好的解去移動，每一次疊代

皆會更新權重值，最終它會收斂至一個區域最佳解。

在訓練信念網路時，我們想要最大化 $P_w(D)=\prod_{d=1}^{|D|}P_w(\boldsymbol{X}_d)$，它可借由隨著 $\ln P_w(D)$ 函數梯度為負的方向來完成，梯度降低法的執行步驟如下：

1. **計算梯度**：對每個 i, j, k 計算

$$\frac{\partial \ln P_w(D)}{\partial w_{ijk}}=\sum_{d=1}^{|D|}\frac{P(Y_i=y_{ij}, U_i=u_{ik}\mid \boldsymbol{X}_d)}{w_{ijk}} \qquad \textbf{(8.2)}$$

方程式 (8.2) 的右半部會對資料集 D 中的每個值組 $\boldsymbol{X}_d$ 計算其機率值，為了簡化描述，我們使用 p 代表其機率值。當 Y_i 與 U_i 所代表的變數對值組 $\boldsymbol{X}_d$ 為隱藏時，機率值 p 可以使用貝氏網路推論的標準方法來算，例如使用商用套裝軟體 HUGIN (*http://www.hugin.dk*)。

2. **往梯度為負的方向移動一小步**：權重藉由下公式更新

$$w_{ijk}\leftarrow w_{ijk}+(l)\frac{\partial \ln P_w}{\partial w_{ijk}} \qquad \textbf{(8.3)}$$

其中 l 代表**學習速率** (learning rate)，代表每一次移動的步伐大小 (step size)，$\partial \ln P_w(D)/\partial w_{ijk}$ 是由公式 (8.2) 計算得知，學習速率一般設定為一個小的常數值，以幫助穩定收斂於最佳解。

3. **重新正規化權重**：由於權重 w_{ijk} 代表機率值，它們必須在 0.0 到 1.0 之間，而且 $\Sigma_j w_{ijk}=1$ 於所有的 i, k，為了符合這些限制，在使用公式 (8.3) 更新權重值之後，要再對這些權重值進行正規化 (normalization) 的動作。

應用此演算法學習的稱為適應式機率網路 (Adaptive Probabilistic Networks)。使用信賴網路來預測分類，會需要大量的數值計算。由於信賴網路詳盡地表達了事件間的因果結構，所以訓練信念網路時，可由人類專家提供她對該領域寶貴的知識，作為決定網路的拓樸結構與條件機率值的參考，如此可以明顯地改善學習品質。

8.2 倒傳遞類神經網路分類法

倒傳遞 (backpropagation) 是一種類神經網路 (neural network) 的學習演算法，類神經網路的研究風潮最早是由心理學家與神經學家所激起的，他們想要發展與神經元 (neurons) 類似的計算單元。概略地說，**類神經網路** (neural network) 是由一組互相連結的輸入 / 輸出單元 (input / output units) 所組成，每一個聯結上面都對應一個權重 (weight)。在學習階段中，類神經網路學習調整聯結上的權重值以能夠正確預測輸入值組的類別標籤，因此，類神經網路也被稱為聯結式學習 (connectionist learning)。

類神經網路需要較長的訓練時間，因此它比較適合可接受長時間訓練的應用領域，而且它當中有許多參數必須要憑經驗才能設定最好的數值，例如網路的拓樸結構 (network topology)，類神經網路另一個飽受批評的地方，是它的可解讀性很低，舉例來說，使用者很難去理解學習過的權重值與網路中隱藏單元背後的意義，這些特性使得類神經網路較不適合應用於資料探勘。

但是類神經網路也有許多優點，例如它對雜訊值的容許度較高，以及對學習階段沒有遇到的未知資料，也具有能力正確地分類，當我們對屬性與類別間的關係所知極少的時候，這是很有用的。而且與絕大多數決策樹演算法不同，它非常適用於連續值的資料。類神經網路已成功應用於許多領域，例如手寫字辨識、臨床病理檢驗、語音辨識與處理。類神經網路的架構與生俱來的便是平行式 (parallel) 處理，平行式運算可以提升類神經網路的計算效能。除此之外，最近有提出一些新方法來從類神經網路中萃取出規則，這些吸引人的因素增加類神經網路應用於資料探勘的分類與數值預測的實用性。

有許多種不同的類神經網路架構與學習演算法，最普遍的是 1980 年代提出的倒傳遞法，在 8.2.1 節將介紹多層前饋式類神經網路，它的學習演算法即是倒傳遞法。8.2.2 節討論如何定義類神經網路的拓樸結構，8.2.3 節介

紹倒傳遞演算法，8.2.4 節介紹如何從類神經網路中萃取規則。

8.2.1 多層前饋式類神經網路

倒傳遞演算法是**多層前饋式類神經網路** (multilayer feed-forward networks) 的學習演算法，它疊代式地學習調整權重值來預測資料值組的類別標籤。多層前饋式類神經網路由一個輸入層 (input layer)、一個或多個隱藏層 (hidden layer) 以及一個輸出層 (output layer) 所組成，圖 8.2 是一個多層前饋式類神經網路的範例。

類神經網路中的每一層是由許多運算單元所組成，類神經網路的輸入為資料值組的屬性值，這些輸入同時地餵入 (feed) **輸入層**中的運算單元，通過輸入層的輸入值經過加權計算並同時地餵入第二層的“類似神經” (neurolike) 的運算單元，第二層被稱為**隱藏層**。隱藏層的輸出將會餵入至下一個隱藏層，依此類推。隱藏層的數目可以是任意的，不過實際上，通常只需要一層隱藏層就足夠了。最後一層的隱藏層將其輸入經過加權計算後，便將計算結果餵入至**輸出層**中的運算單元，而輸出層所輸出的結果，即是對輸入的資料值組所預測的結果。

在輸入層的運算單元稱**輸入單元** (input units)，而在隱藏層與輸出層的運算單元，因為它們類似神經元的運算基礎，也稱為**神經單元**

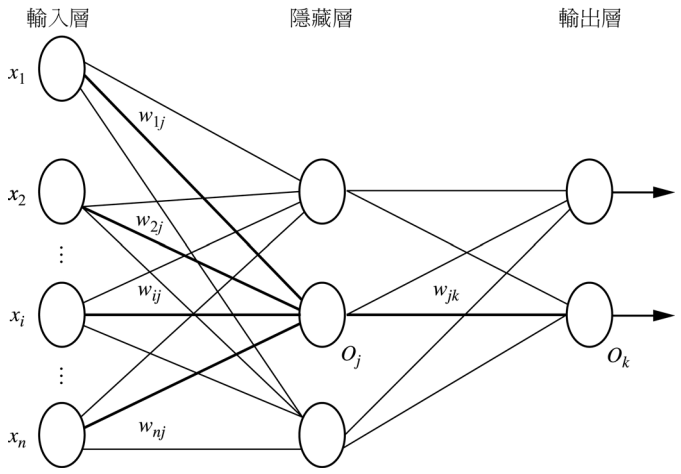


圖 8.2 多層前饋式類神經網路。

(neurodes)。圖 8.2 中的多層前饋式類神經網路架構共有兩層的神經單元，所以，我們稱它為**兩層式類神經網路** (two-layer neural network)。我們並沒有將輸入層納入計算，那是因為輸入層僅僅將資料傳送到下一層。相同地，包含兩層隱藏層的類神經網路，則稱為三層式類神經網路，依此類推。它被稱為**前饋式** (feed-forward)，是因為沒有網路聯結會繞回至輸入層或是前面的隱藏層。此外，每一個提供輸入的運算單元都與他下一層中的單元是**完全連接**。

每一個神經元接收其上一層中單元輸出值的加權總合，做為此神經元的輸入值 (如稍後的圖 8.4 所示)，然後對此加權後的輸入值套用一個非線性激發 (activation) 函數，多層前饋式類神經網路能夠根據輸入值的非線性組合來建立類別預測模型，從統計的觀點來看，它其實是在執行一個非線性迴歸。而多層前饋式類神經網路吸引人的魔力是：只要給予足夠多的隱藏層單元與訓練樣本，它可以近似估計任何函數。

8.2.2 設定類神經網路的拓樸結構

在訓練類神經網路之前，使用者必須事先決定網路的拓樸結構 (topology)，包含定義輸入層內的單元數目、隱藏層的數目 (如果超過一層的話)、每一個隱藏層內的單元數目，以及在輸出層內的單元數目。

對輸入值組中的每一個屬性值進行正規化 (normalizing)，能有助於提升訓練的速度，一般是將連續值屬性正規化至 0.0 至 1.0 的區間內，對於離散值屬性，則是將它編碼成一個輸入單元對應該屬性中一個屬性值，舉例來說，如果屬性 A 共包含三個種可能的屬性值 $\{a_0, a_1, a_2\}$，我們可以指派三個輸入單元 I_0, I_1, I_2 來代表 A 的值，如果 $A = a_1$，則 I_1 設定為 1，而其餘的 I_0 與 I_2 則設定為 0，依此類推。

類神經網路可以用於分類 (預測輸入值組的類別標籤) 與數值預測 (預測一個連續值的輸出)，對於分類問題，一個輸出單元可以用來代表兩個類別，輸出值為 1 代表一個類別，輸出值為 0 則代表另一個類別，當類別數目超過兩個以上時，則是用一個輸出單元對應一個類別。

並沒有明確的規則告訴我們如何決定隱藏層內的“最佳”的單元數目，網路拓樸結構設定通常採用嘗試錯誤 (trail-and-error) 的方法，而且網路拓樸結構會影響網路訓練後的正確率，此外，網路聯結上的權重初始值也會影響網路訓練後的正確率。當訓練後的類神經網路的預測正確率是無法接受時，通常會設定不同的拓樸結構與權重初始值，然後再重新開始訓練。交叉驗證 (cross-validation) 可用來估計類神經網路的正確率，以決定是否已找到一個可接受的網路。近年來，有數種自動式找出“最好”的網路拓樸架構的技術已被提出來，它們通常是使用爬坡法 (hill-climbing) 來對初始結構進行有選擇性的修改。

8.2.3 倒傳遞演算法

倒傳遞 (backpropagation) 演算法是一種疊代式的訓練方法，每一次疊代時，它比對每一個資料值組的預測結果與真實的目標值 (target value) 之間的誤差，目標值可以是此訓練資料值組的類別標籤 (對分類問題) 或是一個連續數值 (對數值預測問題)，對每一個訓練資料值組，我們藉由調整網路聯結上的權重值來最小化網路預測結果與真正目標值間的均方誤差值 (mean squared error)，它以向後倒傳遞的方向來修正這些權重值 (因此稱為倒傳遞)，也就是說，它從輸出層開始，往後饋方向經過每一個隱藏層，直到第一個隱藏層為止，雖然無法證明，但這些權重值通常最終都會收斂，然後演算法停止。圖 8.3 顯示倒傳遞演算法的執行步驟，第一次看到類神經網路的學習演算法，妳可能會覺得這些步驟很棘手，不過妳一但熟悉之後，就會發現這些步驟的概念其實是很簡單的，底下我們介紹這些步驟的內容。

1. **初始權重** (initialize the weights)：網路聯結上的權重初始值為小的隨機數值 (例如，從 −1.0 至 1.0 或從 −0.5 至 0.5 間的隨機數值)，網路上每個運算單元也都會對應一個偏移量 (bias)，這些偏移量同樣亦會設為小的隨機數值，我們會在稍後介紹偏移量。

   對每一個訓練資料值組，我們執行下列步驟

運算法：倒傳遞法是多層前饋式類神經網路應用於分類與數值預測時的學習演算法。

輸入：

- D 為資料集，它包含訓練資料值組與其對應的目標值；
- l 為學習速率；
- *network* 為多層前饋式網路。

輸出：一個訓練過的類神經網路。

方法：

(1) 設定所有 *network* 中的聯結權重與單元偏移量的初始值；

(2) **while** 結束條件未滿足{

(3) **for** D 中每個資料值組 X {

(4) // 向前傳遞輸入值

(5) **for** 輸入層中的每個單元 j {

(6) $O_j = I_j$; // 在輸入層中的單元，其輸出值便是它的輸入值

(7) **for** 每個隱藏層或輸出層中單元 j {

(8) $I_j = \sum_i w_{ij} O_i + \theta_j$; //根據前一層的輸出值來計算單元 j 的輸入值

(9) $O_j = \frac{1}{1+e^{-I_j}}$; } // 計算單元 j 的輸出值

(10) // 倒傳遞錯誤值

(11) **for** 輸出層中每個單元 j

(12) $Err_j = O_j(1-O_j)(T_j - O_j)$; // 計算錯誤值

(13) **for** 每個隱藏層中的每個單元 j，從最後一個隱藏層到第一個隱藏層

(14) $Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$; // 根據上一個隱藏層 k 來計算錯誤

(15) **for** *network* 中的每個聯結權重 w_{ij} {

(16) $\Delta w_{ij} = (l) Err_j O_i$; //計算權重修正值

(17) $w_{ij} = w_{ij} + \Delta w_{ij}$; } //權重更新

(18) **for** *network* 中每個偏移量 θ_j {

(19) $\Delta \theta_j = (l) Err_j$; // 計算偏移量修正值

(20) $\theta_j = \theta_j + \Delta \theta_j$; } // 偏移量更新

(21) }}

圖 8.3 倒傳遞演算法。

2. **往前傳遞輸入值** (propagate the inputs forward)：一開始，將訓練資料值組餵入 (fed) 輸入層，輸入值通過輸入層時維持不變，也就是說，對於輸入單元 j，它的輸入值 I_j 與輸出值 O_j 是相同的。接下來，計算隱藏層與輸出層中每個運算單元的輸入值與輸出值，運算單元的輸入值為它之輸入的線性組合，為了幫助理解這一點，我們使用圖 8.4 來說明。每一個單元都有許多輸入，這是由它上一層中與它連結的單元之輸出的值，每一個網路聯結都對應一個權重，所有連結至此單元的輸入，乘上網路聯結上的權重值，最後加總起來，便是餵進此單元的輸入值。對於隱藏層 (或輸出層) 中的單元 j，其輸入值 I_j 為

$$I_j = \sum_i w_{ij} O_i + \theta_j \qquad \textbf{(8.4)}$$

w_{ij} 代表從上一層單元 i 連結至單元 j 的聯結權重，O_i 代表上一層單元 i 的輸出值，而 θ_j 代表單元 j 的**偏移量** (bias)，偏移量作為改變此神經單元激發狀態之門檻值 (threshold)。隱藏層 (或輸出層) 中每一個單元將它的輸入值代入至一個**激發函數** (activation function)，如圖 8.4 所示，激發函數象徵此神經元的激發狀態，常用的激發函數有**邏輯斯** (logistic) 與 *S* **型** (sigmoid) 函數。假設單元 j 的輸入值為 I_j，則其輸出值 O_j 為

$$O_j = \frac{1}{1+e^{-I_j}} \qquad \textbf{(8.5)}$$

激發函數也稱為壓扁函數 (squashing function)，因為它把大的輸入範圍對應至 0 到 1 的小區間內。邏輯斯函數是非線性與可微分的，這使得倒傳遞演算法能對線性不可分割的資料集建立分類模型。

從第一個隱藏層往上，直到最後的輸出層，我們對每一個隱藏層與輸出層內的單元計算其輸出值 O_j，輸出層的輸出值結果即為類神經網路的預測結果，實際執行時，由於所有單元的輸出值在之後執行倒傳遞錯誤值時會再用到，因此將這些輸出值暫時儲存起來以節省計算負擔是很好的運算技巧。

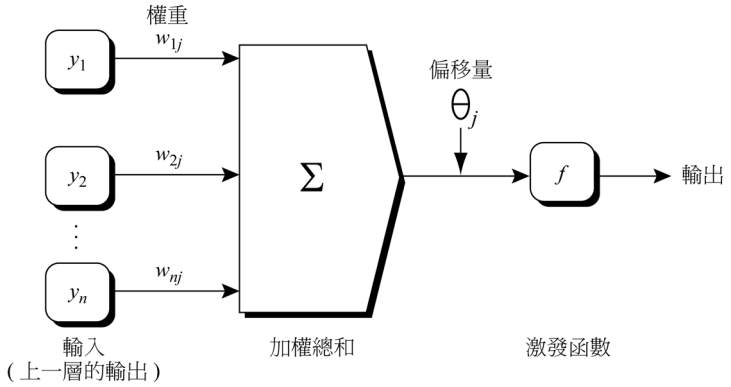


圖 8.4 隱藏層或輸出層中的單元 *j*：此單元 *j* 的輸入是前一層的輸出值。這些輸入值與相對應的聯結權重相乘後並進行加總，然後再加上偏移量。最終，將計算結果代入激發函數 *f*，即為此單元 *j* 的輸出值。

3. **倒傳遞錯誤值** (backpropagate the error)：類神經網路預測的錯誤值會向後傳遞，並根據這些倒傳遞的錯誤值來修正網路連結上的權重值以及網路單元上的偏移量。對於輸出層內的單元 j，其錯誤值 Err_j 的計算公式如下：

$$Err_j = O_j(1-O_j)(T_j - O_j) \tag{8.6}$$

其中 O_j 為單元 j 的輸出值，T_j 為輸入資料值組的目標值，而 $O_j(1-O_j)$ 為邏輯斯函數的微分結果。

要計算隱藏層內單元 j 的倒傳遞錯誤值，則必須考慮與此單元相連結的上一層單元的錯誤值的加權總和，隱藏層內單元 j 的錯誤值計算公式如下：

$$Err_j = O_j(1-O_j)\sum_k Err_k w_{jk} \tag{8.7}$$

其中 Err_k 為上一層中單元 k 的錯誤值，w_{jk} 為連結單元 j 與單元 k 的聯結之權重，網路中的聯結權重與單元的偏移量則是根據傳遞的錯誤值來進行更新，權重的更新公式如下，其中 Δw_{ij} 代表權重 w_{ij} 的修正值：

$$\Delta w_{ij} = (l) Err_j O_i \tag{8.8}$$

$$w_{ij} = w_{ij} + \Delta w_{ij} \quad \textbf{(8.9)}$$

變數 l 稱為**學習速率**，通常設定為 0 至 1 之間的一個小常數值，倒傳遞演算法利用梯度降低法來蒐尋能讓網路預測的結果與真實的目標值之間的均方誤差能夠最小化的權重值[1]。學習速率能夠幫助避免陷入區域最小值的狀況 (區域最小值是指權重雖然收斂了，但並不是找到最佳的權重值)，並且促進找到全域最小值 (global minimum)。如果學習速率設定的太小，則會使用非常緩慢的步伐來學習；但如果學習速率設定的太大，則學習會在不適當的解之間擺盪。經驗法則是將學習速率設定為 $1/t$ ，其中 t 為疊代的次數。

偏移量更新的公式如下，其中 $\Delta\theta_j$ 代表偏移量 θ_j 的修正值：

$$\Delta\theta_j = (l)Err_j \quad \textbf{(8.10)}$$

$$\theta_j = \theta_j + \Delta\theta_j \quad \textbf{(8.11)}$$

請注意，這裡我們會在餵入每一個資料值組之後，進行權重與偏移量的更新，這種方法也稱為**個案更新** (case updating)。除此之外，另一種方式則是每餵入一筆資料值組，便將公式 (8.8) 與 (8.10) 這些修正值進行累加之後儲存在變數中，在全部的資料值組皆餵入之後，再進行權重與偏移量的更新，這種方法稱為**批次更新** (epoch updating)，其中將全部訓練資料值組餵入網路中運算過一次的疊代稱為一個**批次** (epoch)。理論上，使用微分推導的倒傳遞公式使用批次更新，但在實際上執行時，個案更新更為普遍，因為它能獲得更佳的正確率。

4. **終止條件** (terminating condition)：當以下條件發生時，則訓練停止
   - 在上一批次中，所有的 Δw_{ij} 值都比一個使用者預設的門檻值來得小。
   - 在上一批次中，訓練值組分類錯誤的比率低於使用者預設的門檻值。
   - 訓練進行的批次數目超過使用者指定的最上限。

[1] 在 8.1.2 節中曾介紹使用梯度降低法來訓練貝氏信念網路。

實際執行時，倒傳遞演算法可能需要執行數十萬批次才能讓權重值收斂，倒傳遞計算效能如何呢？假設有 $|D|$ 個訓練資料值組與 w 個權重，每一批次則需 $O(|D|\times w)$ 時間，然而，在最差的情況下，訓練進行的批次數目可能為 n 的冪次方，其中 n 是輸入變數的數目。實際上，權重收斂所需要的時間有很大的變異性，有許多學者提出不同的技術來縮減訓練的時間，例如模擬退火法 (simulated annealing) 可用來確保能夠收斂到全域最佳解。

範例 8.1 倒傳遞學習演算法的計算範例

圖 8.5 為一個多層前饋式類神經網路的範例，假設學習速率為 0.9，初始的權重值及偏移量顯示在表格 8.1 中。第一個餵入的訓練資料值組為 $X=(1,0,1)$，它對應的類別標籤為 1。

此範例說明當此值組餵入網路後，倒傳遞演算法詳細的計算步驟，首先，計算網路中每一個單元的輸入值與輸出值，這些輸入 / 輸出數值的計算顯示在表格 8.2 中，接著，計算每一個單元的錯誤值，並將錯誤值向後傳遞，這些錯誤值的計算顯示在表格 8.3 中，最後，進行權重值與偏移量的更新，其計算過程顯示在表格 8.4 中。

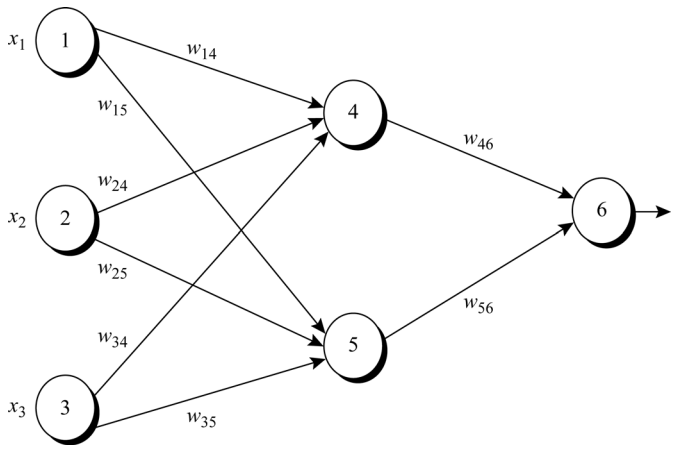


圖 8.5 多層前饋式類神經網路的範例。

表 8.1 輸入、初始權重值與偏移量

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	−0.3	0.4	0.1	−0.5	0.2	−0.3	−0.2	−0.4	0.2	0.1

表 8.2 輸入值與輸出值的計算

單元 j	輸入 I_j	輸出 O_j
4	$0.2+0-0.5-0.4=-0.7$	$1/(1+e^{0.7})=0.332$
5	$-0.3+0+0.2+0.2=0.1$	$1/(1+e^{-0.1})=0.525$
6	$(-0.3)(0.332)-(0.2)(0.525)+0.1=-0.105$	$1/(1+e^{0.105})=0.474$

表 8.3 計算每個單元的錯誤值

單元 j	錯誤值 Err_j
6	$(0.474)(1-0.474)(1-0.474)=0.1311$
5	$(0.525)(1-0.525)(0.1311)(-0.2)=-0.0065$
4	$(0.332)(1-0.332)(0.1311)(-0.3)=-0.0087$

表 8.4 權重值與偏移量更新的計算

權重或偏移量	新值
w_{46}	$-0.3+(0.9)(0.1311)(0.332)=-0.261$
w_{56}	$-0.2+(0.9)(0.1311)(0.525)=-0.138$
w_{14}	$0.2+(0.9)(-0.0087)(1)=0.192$
w_{15}	$-0.3+(0.9)(-0.0065)(1)=-0.306$
w_{24}	$0.4+(0.9)(-0.0087)(0)=0.4$
w_{25}	$0.1+(0.9)(-0.0065)(0)=0.1$
w_{34}	$-0.5+(0.9)(-0.0087)(1)=-0.508$
w_{35}	$0.2+(0.9)(-0.0065)(1)=0.194$
θ_6	$0.1+(0.9)(0.1311)=0.218$
θ_5	$0.2+(0.9)(-0.0065)=0.194$
θ_4	$-0.4+(0.9)(-0.0087)=-0.408$

「如何利用訓練好的類神經網路來分類輸入資料值組呢？」為了分類一個未知的資料值組 X，將此值組餵入訓練好的類神經網路，然後計

算每一個單元的輸入值與輸出值 (不需要倒傳遞錯誤值)，如果是一個輸出單元 (節點) 對應一個類別標籤，則輸出值最高的單元所對應的類別標籤，即為類神經網路預測此資料值組 X 的類別標籤。如果只有一個輸出單元 (節點)，此時輸出值大於 0.5 便代表資料值組 X 屬於正類別，若輸出值小於 0.5 則代表資料值組屬於負類別。

有數種改良式倒傳遞演算法被提出，包含能夠動態調整網路拓樸結構、學習速率以及其他參數，以及使用不同的錯誤函數。

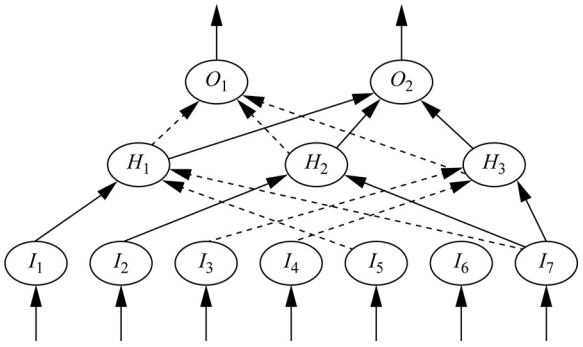
8.2.4 黑箱：倒傳遞類神經網路的可解讀性

類神經網路就像一個黑箱，我們如何理解倒傳遞類神經網路學習到什麼知識呢？類神經網路最大的缺點是知識的表達方式，對於網路中互相連結的單元，從這些聯結上的權重值取得的知識是很難被人類所理解的，這項特質激勵學者研究如何萃取出鑲嵌在訓練過類神經網路上的知識，並且符號化的表達這些知識，這些方法包含規則萃取與敏感度分析 (sensitivity analysis)。已有數種不同的規則萃取方法被提出來，這些方法一般會對網路的學習程序、拓樸結構或離散化輸入值加上一些限制。

完全連接的網路是很難清楚理解的，因此，通常從網路中萃取規則的第一個步驟便是**網路修剪** (network pruning)，它是概念是藉由移除那些影響力最小的網路聯結來簡化網路，如何評估一條網路聯結的影響力呢？舉例來說，如果移除一條網路聯結並不會降低網路分類預測的正確率，則這條網路聯結的影響力便是很小，所以可以刪除掉它。

一旦類神經網路修剪過後，有些規則萃取的方法會對網路聯結、單元、或是激發值進行群集分析 (clustering)，舉例來說，其中一個方法是對每一個隱藏層單元找出具有共同激發值的群組集合，以圖 8.6 中的兩層類神經網路來說明，我們對各個隱藏層單元的激發值的組合進行分析，再依據激發值的組合與對應的輸出單元值的聯繫來推導出規則，同樣地，探討輸入值與隱藏層激發值的聯繫來推導出輸入層與隱藏層的關聯性，最後，這兩類規則被合併成為 IF-THEN 規則的型式。還有些演算法會萃取出不同

型式的規則，包括 M-of-N 規則 (在 N 個前項條件中有 M 個成立才能產生結論)、使用 M-of-N 屬性測試的決策樹、模糊規則 (fuzzy rules) 與有限自動機 (finite automata)。



對每個隱藏節點 H_i ，找出共同的激發值集合 for H_1 : (−1, 0, 1) for H_2 : (0, 1) for H_3 : (−1, 0.24, 1)
依據共同激發值與輸出節點 O_j 的聯繫來推導出規則 IF ($H_2 = 0$ AND $H_3 = -1$) OR ($H_1 = -1$ AND $H_2 = 1$ AND $H_3 = -1$) OR ($H_1 = -1$ AND $H_2 = 0$ AND $H_3 = 0.24$) OR THEN $O_1 = 1, O_2 = 0$ EISE $O_1 = 0, O_2 = 1$

依據輸入節點 I_j 與輸出節點 O_j 的聯繫來推導出規則
IF ($I_2 = 0$ AND $I_7 = 0$) THEN $H_2 = 0$
IF ($I_4 = 1$ AND $I_6 = 1$) THEN $H_3 = -1$
IF ($I_5 = 0$) THEN $H_3 = -1$

獲得輸入與輸出類別聯繫的規則
IF ($I_2 = 0$ AND $I_7 = 0$ AND $I_4 = 1$ AND
$I_6 = 1$) THEN 類別 = 1
IF ($I_2 = 0$ AND $I_7 = 0$ AND $I_5 = 0$) THEN
類別 = 1

圖 8.6 從訓練好的類神經網路中擷取規則。

敏感度分析 (sensitivity analysis) 是用來估計一個輸入變數對於類神經網路輸出結果的影響，首先，我們將其他輸入變數的值固定住，然後監控此輸入變數的值改變時，網路輸出的值有何改變，敏感度分析方式萃取得到的知識可表達成 "IF *X* 降低 5% THEN *Y* 增加 8%" 規則表示法。

8.3 支持向量機

支持向量機 (support vector machine, SVM) 是一種熱門的線性 / 非線性分類方法，他的作法概略如下，首先，它使用一個非線性轉換 (nonlinear transformation) 將原始資料映射 (mapping) 至較高維度的特徵空間 (feature space) 中，然後在高維度特徵空間中，它找出一個最佳的線性分割超平面 (linear optimal separating hyperplane) 來將這兩個類別的資料值組分割開來，此分割超平面也可稱為決策分界線 (decision boundary)。藉由挑選適當的非線性轉換，以及將資料映射至足夠高維度的特徵空間，這兩個類別的資料值組在高維度特徵空間中必定能被一個超平面分隔開來，支持向量機利用重要的資料值組作為建構分割超平面的支持向量 (support vector)，而最佳的超平面係指邊界 (margin) 最大化的超平面，詳細的概念我們會在後續介紹。

第一篇有關支持向量機的文章是由 Vapnik、Boser 與 Guyon 在 1992 年所提出，不過它最早在 1960 年代就由 Vapnik 與 Chervonenkis 所提出統計學習理論 (statistical learning theory) 與 Vapnik-Chervonenkis dimension 奠定研究的基礎，雖然支持向量機的訓練時間可能會很久，但是由於它能夠建構複雜的非線性決策分界線，支持向量機具有極高的正確率。而且相較於其他的分類方法，支持向量機較沒有過度學習 (overfitting) 的問題，同時它能提供一個更簡潔的分類模型。支持向量機可應用於分類與數值預測的問題，例如在手寫字辨識、物件辨識、語音辨識與時間序列預測等應用領域，支持向量機都有相當優異的表現。

8.3.1 資料為線性可分割的情況

要揭開支持向量機 (以下簡稱為 SVM) 的神秘面紗，我們從最簡單的例子開始：兩個類別的分類問題，而且這兩類別的資料是線性可分割的。假設資料集 D 包含 $(\boldsymbol{X}_1, y_1), (\boldsymbol{X}_2, y_2), \ldots, (\boldsymbol{X}_{|D|}, y_{|D|})$，其中 $\boldsymbol{X}_i$ 代表訓練資料值組，其對應的類別標籤為 y_i，每一個類別標籤的值為 +1 或是 −1，亦即 $y_i \in \{-1, +1\}$，以購買電腦分類問題為例，y_i 為 +1 對應於 { 購買電腦=是 } 的類別，y_i 為 −1 對應於 {購買電腦=否} 的類別，為了方便視覺化說明支持向量機的基本概念，讓我們考慮僅包含兩個輸入屬性 A_1 與 A_2 的範例，如圖 8.7 所示，我們可以看到此 2D 資料集是線性可分割的，換句話說，我們可以劃出一條直線來分隔開所有類別 +1 與類別 −1 的資料值組，然而，能夠分割這兩個類別的直線有無窮多條，我們希望找出其中“最佳”的一條分割直線，亦即，它對於之前未見的資料值組，能夠具有最小的分類錯誤率，我們該如何找出“最佳”的分割直線呢？請注意如果是 3D 資料 (亦即有 3 個屬性)，我們要找的是最佳分割平面 (plane)，延伸到 n 維度，我們要找的是最佳超平面 (hyperplane)，在本節，不管輸入屬性的數目如何，我們將使用超平面這個名詞代表我們所要尋找的決策分界線 (decision boundary)。

該如何尋找最佳的超平面呢？SVM 尋找具有**最大邊界的超平面** (maximum marginal hyperplane)。圖 8.8 中顯示兩個可能的分割超平面與其對應的邊界 (margin)。在我們詳細定義何謂邊界之前，先讓我們直覺地比較圖中的兩個超平面，它們都可以正確地分類所有給定資料，然而，直覺地，我們預期有較大邊界的超平面比邊界較小的超平面，更能夠正確地分類那些未知的資料值組。這也是為何 SVM 在訓練階段尋找邊界最大的超平面，其對應的邊界能對這兩個類別提供最大的分割度。

現在，我們正式地定義何謂邊界，一個分割超平面可表示為

$$\boldsymbol{W} \cdot \boldsymbol{X} + b = 0 \qquad \textbf{(8.12)}$$

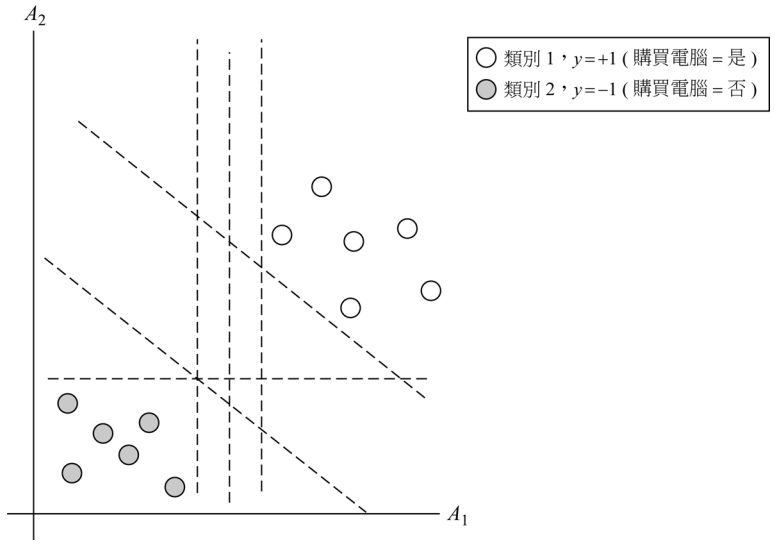


圖 8.7 此 2D 資料集為線性可分割的，能正確分類給定二個類別的分割超平面 (或決策分界線) 有無窮多種可能，其中一些我們在圖上用虛線表示，哪一條分割超平面是最好的？

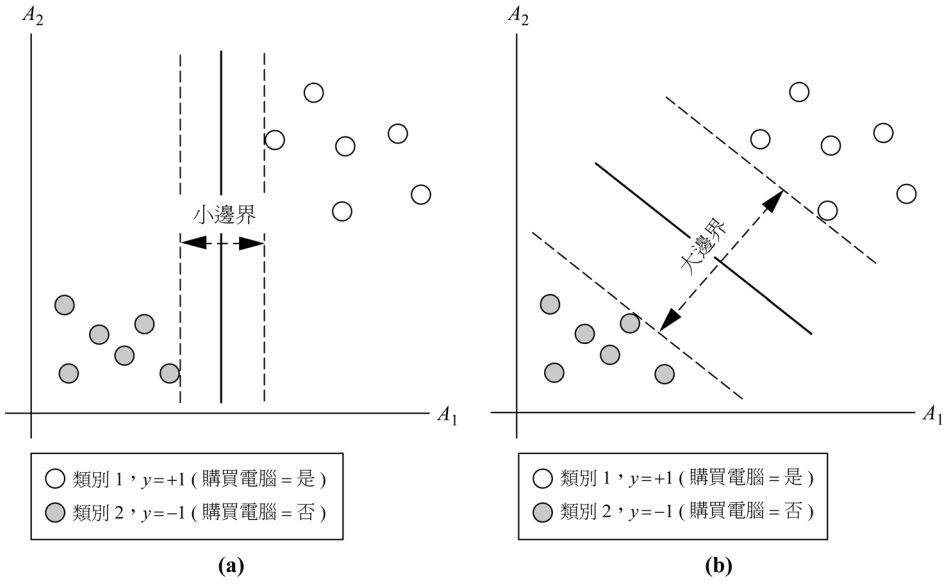


圖 8.8 這裡我們顯示兩個可能的分割超平面，以及其對應的邊界，哪一個分割超平面會比較好呢？擁有較大邊界的超平面 (b) 應有較高的推理能力，正確率也更佳。

其中 $\boldsymbol{W}=\{w_1, w_2, \ldots, w_n\}$ 代表權重向量，n 為屬性的數目，b 為一個純量，通常也稱為偏移量。為了方便視覺化介紹邊界的概念，讓我們考慮 2D 的訓練資料值組 $\boldsymbol{X}=(x_1, x_2)$，其中 x_1 與 x_2 為屬性 A_1 與 A_2 的值，如果我們把 b 視為額外的權重 w_0，則公式 (8.12) 的分割超平面可改寫成

$$w_0 + w_1 x_1 + w_2 x_2 = 0 \qquad \textbf{(8.13)}$$

任何落在此超平面上半部的資料點滿足

$$w_0 + w_1 x_1 + w_2 x_2 > 0 \qquad \textbf{(8.14)}$$

同樣地，任何落在此超平面下半部的資料點滿足

$$w_0 + w_1 x_1 + w_2 x_2 < 0 \qquad \textbf{(8.15)}$$

我們定義超平面的分割邊界對這兩個類別的“邊緣”為

$$H_1 : w_0 + w_1 x_1 + w_2 x_2 \geq 1 \quad \text{for} \quad y_i = +1 \qquad \textbf{(8.16)}$$

$$H_2 : w_0 + w_1 x_1 + w_2 x_2 \leq 1 \quad \text{for} \quad y_i = -1 \qquad \textbf{(8.17)}$$

也就是說，任何落在邊緣 H_1 上或其上半部的資料值組皆屬於正類別 {+1}；任何落在邊緣 H_2 上或其下半部的資料值組則屬於負類別 {–1}。將公式 (8.16) 與 (8.17) 合併，我們可以得到

$$y_i(w_0 + w_1 x_1 + w_2 x_2) \geq, \forall i \qquad \textbf{(8.18)}$$

落在 H_1 與 H_2 上的資料值組稱為**支持向量** (support vectors)，也就是說，它們落在邊界的邊緣上。圖 8.9 中的支持向量用粗邊圓圈表示，支持向量是最難被分類的資料值組，也因此提供最多的資訊來幫助分類。

根據 H_1 與 H_2 的定義，我們可得知計算邊界尺寸的公式，由分割超平面到 H_1 的距離為 $1/\|\boldsymbol{W}\|$，其中 $\|\boldsymbol{W}\|$ 為權重向量 $\boldsymbol{W}$ 的範數 (norm)，也就是 $\sqrt{\boldsymbol{W}\cdot\boldsymbol{W}}$。而分割超平面到 H_2 的距離也是如此，因此，此分割超平面的邊界的大小為 $2/\|\boldsymbol{W}\|$。

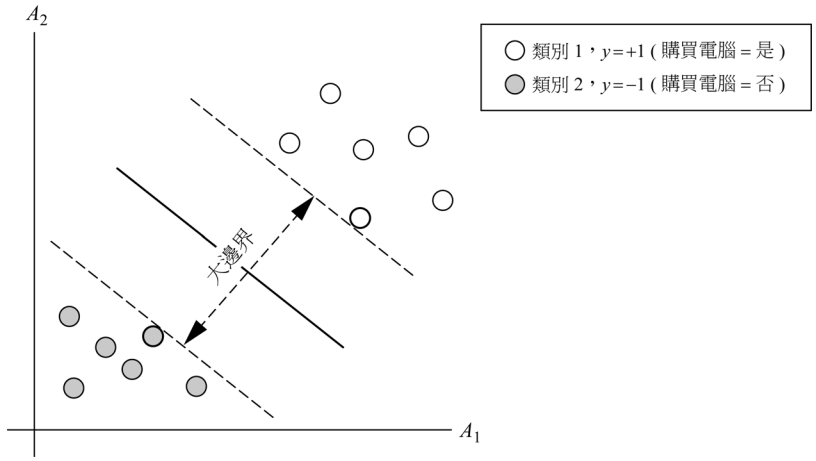


圖 8.9 SVM 尋找具有最大邊界的分割超平面，也就是說，它與最鄰近的訓練資料值組的距離是最遠的，支持向量用粗邊的圓圈表示。

藉由精妙的數學技巧，我們可以將方程式 (8.18) 改寫成具有限制條件的凸面二次最佳化的問題 (constrained quadratic optimization problem)。此精妙的數學技巧已超出本書的範圍，有興趣的讀者可以參閱相關文獻。

當資料量很小時 (小於 2000 筆)，一般具有求最佳解的套裝軟體，都可以找出支持向量與最大邊界的超平面，對於資料量極大時，則必須使用特殊且有效率的演算法來訓練 SVM，此演算法已超出本書的範圍。請注意，是由支持向量建構出分割超平面，一旦我們找到支持向量與最大邊界超平面 (MMH)，則此支持向量機就已經訓練好了。由於最大邊界的超平面 (MMH) 是一個線性分界線，可用來分類線性可分割的資料，我們稱此訓練好的 SVM 分類器為線性 SVM (linear SVM)。

根據拉格朗日公式，最大邊界的超平面 (MMH) 為

$$d(\boldsymbol{X}^T)=\sum_{i=1}^{l} y_i\alpha_i \boldsymbol{X}_i \boldsymbol{X}^T + b_0 \qquad \textbf{(8.19)}$$

其中 y_i 為支持向量 $\boldsymbol{X}_i$ 的類別標籤，$\boldsymbol{X}^T$ 為測試值組，α_i 與 b_0 為 SVM 求解最佳化問題所得到的參數，l 為支持向量的個數。α_i 為拉格朗日乘數 (Lagrangian multipliers)，支持向量為訓練資料值組的子集合。

當輸入測試值組 $\boldsymbol{X}^T$ 時，我們將 $\boldsymbol{X}^T$ 代入公式 (8.19) 中，並檢驗結

果的正負號，它告訴我們測試資料值組落在超平面的哪一面上。如果結果為正號，代表 $\boldsymbol{X}^T$ 落在超平面的上半部，則 SVM 預測 $\boldsymbol{X}^T$ 屬於正類別 {+1}，也就是我們例子中，對應到 { 購買電腦 = 是 } 的類別。如果結果為負號，則 $\boldsymbol{X}^T$ 在超平面的下半部，則 SVM 預測 $\boldsymbol{X}^T$ 屬於負類別 {–1}，亦即對應到 { 購買電腦 = 否 } 的類別。

請注意公式 (8.19) 中包含向量 $\boldsymbol{X}_i$ 與 $\boldsymbol{X}^T$ 的內積 (dot product) 計算。它對於在線性不可分隔的情形下，尋找最大邊界超平面 (MMH) 與支持向量是非常有用的。

在討論線性不可分割的情況之前，有幾點重要的觀念值得注意。第一，SVM 所學習得到的分類器，其複雜度取決於支持向量的數目，而不是資料的維度，因此 SVM 比較不會有過度學習的問題。其次，支持向量是關鍵重要的訓練資料值組，因為它們是距離超平面 (MMH) 最接近，也是最難被分類正確的訓練值組，如果能把它們分類正確，那麼其餘的訓練值組也都可以被正確地分類，換句話說，如果我們移除支持向量以外所有的訓練值組，並且再重新訓練 SVM，我們依然會得到“相同”的超平面。更進一步，支持向量的數目可用來計算 SVM 預期分類錯誤率的上界，而此錯誤率的上界依舊與資料維度無關。一個支持向量數量稀少的 SVM 分類器，即使面對資料的維度十分高時，其推理能力 (generalization) 仍然很好。

8.3.2 資料為線性不可分割的情況

在 8.3.1 節我們學到線性 SVM，但是當資料為線性不可分割的情況，例如圖 8.10 所舉的例子，找不到一個直線能分割開這兩個類別，此時線性 SVM 在訓練時找不到一個可行解，我們該怎麼辦呢？解決的方法就是將線性 SVM 延伸為非線性 SVM，並用來分類線性不可分隔的資料，非線性 SVM 可以在輸入空間 (input space) 中找到一個非線性決策超曲面 (decision hypersurface)。

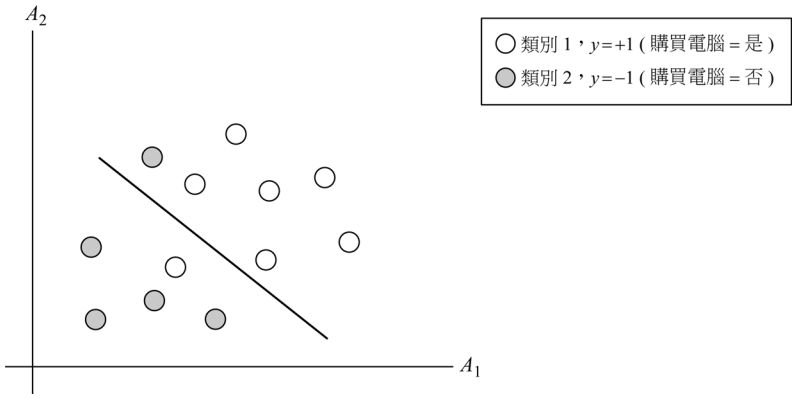


圖 8.10 一個簡單的線性不可分隔的 2D 資料集，不同於圖 8.7，我們無法找到一條直線來分割這兩個類別，相反地，它的決策分界線必須是非線性的曲線。

要將線性 SVM 延伸至非線性 SVM，有兩個主要步驟。第一步，我們使用一個非線性轉換將原始資料映射至較高維度的特徵空間 (feature space)，此步驟中，有許多種常見的非線性轉換可供選用。下一個步驟，我們在高維度特徵空間中找出最佳的線性分割超平面。在此處我們同樣使用線性 SVM 找出最大邊界超平面所用的二次最佳化問題 (quadratic optimization problem)，而我們在高維度特徵空間中找到的最大邊界超平面 (MMH)，對應至原來空間就是一個非線性分割超曲面 (nonlinear separating hypersurface)。

範例 8.2 利用非線性轉換將原始資料映射至較高維度的特徵空間

考慮以下情況，使用一個非線性轉換將 3-D 輸入向量 $\boldsymbol{X}=(x_1, x_2, x_3)$ 映射至 6-D 的特徵空間 $\boldsymbol{Z}$，映射方式為 $\Phi_1(\boldsymbol{X})=x_1$，$\Phi_2(\boldsymbol{X})=x_2$，$\Phi_3(\boldsymbol{X})=x_3$，$\Phi_4(\boldsymbol{X})=(x_1)^2$，$\Phi_5(\boldsymbol{X})=x_1x_2$，$\Phi_6(\boldsymbol{X})=x_1x_3$。在特徵空間的決策超平面為 $d(\boldsymbol{Z})=\boldsymbol{W}\boldsymbol{Z}+b$，其中 $\boldsymbol{W}$ 與 $\boldsymbol{Z}$ 為向量，它在特徵空間中為線性超平面，當我們使用二次最佳化問題求解出最佳的 $\boldsymbol{W}$ 與 b，並將其代入特徵空間 $\boldsymbol{Z}$ 的決策超平面，它會對應到原始 3-D 空間的一個非線性二次多項式決策函數

$$d(\boldsymbol{Z}) = w_1x_1 + w_2x_2 + w_3x_3 + w_4(x_1)^2 + w_5x_1x_2 + w_6x_1x_3 + b$$
$$= w_1z_1 + w_2z_2 + w_3z_3 + w_4x_4 + w_5z_5 + w_6z_6 + b$$

現在的問題是我們該如何選擇非線性轉換，另一個問題是這裡會用到的計算十分花成本，回顧公式 (8.19)，為了分類測試值組 $\boldsymbol{X}^T$，我們必須計算它與所有支持向量的內積 (dot product)[2]。同樣地，在進行 SVM 訓練時，我們也必須使用大量的內積計算來找到最大邊界超平面 (MMH)，由於在高維度空間中計算內積的代價是非常昂貴的，因此我們需要其它的技巧。

幸運的是我們發現在解二次最佳化問題時 (也就是在高維度特徵空間中尋找線性 SVM 時)，訓練值組僅會以內積 $\Phi(\boldsymbol{X}_i)\cdot\Phi(\boldsymbol{X}_j)$ 的形式出現，其中 $\Phi(\boldsymbol{X})$ 是將訓練值組經過非線性轉換 Φ 映射至高維度特徵空間的結果，除了直接計算映射後值組的內積外，我們可以應用核心函數 (kernel function) 來代表內積的計算

$$\boldsymbol{K}(\boldsymbol{X}_i, \boldsymbol{X}_j) = \Phi(\boldsymbol{X}_i)\cdot\Phi(\boldsymbol{X}_j) \qquad \textbf{(8.20)}$$

換句話說，在訓練過程所有的 $\Phi(\boldsymbol{X}_i)\cdot\Phi(\boldsymbol{X}_j)$，都可以用 $\boldsymbol{K}(\boldsymbol{X}_i, \boldsymbol{X}_j)$ 取代，利用核心函數的技巧，所有的計算都可以在原始空間完成，原始空間的維度小了很多，計算也更有效率，而且我們也可避免真正的把資料映射到較高維度空間，透過核心函數，我們甚至不用知道映射函數的詳細內容是什麼。稍後，我們會介紹有哪些種類的核心函數可供我們使用。

藉由核心函數的技巧，我們可以使用 8.3.1 節介紹的線性 SVM 訓練程序來找出最大邊界超平面，其中不同的地方是非線性 SVM 會加入一個使用者設定的參數 C，參數 C 控制拉格朗日乘數 α_i 的上界，最佳的 C 值通常是由實驗測試來決定的。

[2] 兩個向量 $\boldsymbol{X}^T = (x_1^T, x_2^T, \ldots, x_n^T)$ 與 $\boldsymbol{X}_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ 的內積為 $x_1^T x_{i1} + x_2^T x_{i2} + \cdots + x_n^T x_{in}$。請注意，它對每一個維度都必須執行一次乘法與加法的運算。

核心函數可用來取代內積的計算，常用的核心函數有下列三種：

Polynomial kernel of degree h： $\boldsymbol{K}(\boldsymbol{X}_i, \boldsymbol{X}_j) = (\boldsymbol{X}_i, \boldsymbol{X}_j + 1)^h$

Gaussian radial basis function kernel： $\boldsymbol{K}(\boldsymbol{X}_i, \boldsymbol{X}_j) = e^{-\|X_i - X_j\|^2 / 2\sigma^2}$

Sigmoid kernel： $\boldsymbol{K}(\boldsymbol{X}_i, \boldsymbol{X}_j) = \tanh(kX_i \cdot X_j - \delta)$

不同的核心函數的訓練結果，會對應到輸入空間上不同類型的非線性分類器，類神經網路愛用者會有興趣注意到，非線性 SVM 所找到的分類模型，與某些著名的類神經網路所用的分類模型是一模一樣的。例如，SVM 使用高斯 RBF 核心函數 (Gaussian radial basis function) 所找到的分類器與 RBF 類神經網路相同；SVM 使用 *S* 型核心函數 (sigmoid kernel) 所找到的分類器等同於簡單的兩層類神經網路。

並沒有金科玉律告訴我們該使用哪種核心函數才會得到正確率最高的 SVM 分類器，幸好實際上，挑選不同的核心函數對於分類正確率的結果並不會產生很大的差異。不同於類神經網路所使用的倒傳遞演算法通常會收斂至區域最佳解，SVM 一定會找到全域最佳解 (global solution)。

到目前為止，我們僅介紹二元 (對兩個類別) 的線性與非線性 SVM 分類器，延伸到 *m* 個類別，我們可以組合數個二元 SVM 分類器來解決多類別分類問題，例如對一個類別訓練一個分類器，或是使用錯誤更正碼 (error-correcting codes) 的技巧。

SVM 的重要研究目標是如何改善它的訓練與測試速度，使得在解決極大型資料集的分類問題時，SVM 能夠變成一個更可行的選項。SVM 其他的研究課題包含如何決定最好的核心函數，以及對多類別分類問題尋找更有效率的解法。

8.4 使用頻繁樣式來進行分類

頻繁樣式 (frequent patterns) 顯示集料集合中頻繁發生的屬性與值配對的有趣關聯關係，舉例來說，我們可能發現 AllElectronics 公司內有購買電腦的客戶中，有 20% 的顧客資料出現 { 年齡 = 青年 } 與 { 信用評等 =

通過 } 的屬性與值的配對關係，我們可以將每一個屬性與值的配對視為一個項目 (item)，並透過頻繁樣式探勘或是頻繁項目集探勘找出這些頻繁樣式。在課本第 5 與 6 章節中，我們看到透過頻繁樣式推導出關連規則，而這些關聯性通常用來分析顧客在大賣場中的購買行為，這些分析對於物品擺放、型錄設計與交叉行銷等決策的制定都是非常有用的。

在本節中，我們探討如何使用頻繁樣式來進行分類，8.4.1 節介紹**關聯分類** (associative classification)，它利用頻繁樣式推導出關聯規則來進行分類，其概略觀念是我們可以尋找頻繁樣式與類別標籤之間的強關聯性。8.4.2 節介紹**鑑別性頻繁樣式基礎的分類法** (discriminative frequent pattern-based classification)，此時頻繁樣式視為特徵的組合，而在建立分類模型時，除了考慮單一特徵之外，也會考慮這些特徵的組合。由於頻繁樣式探索數個屬性之間具有高度信賴性的關聯性，鑑別性頻繁樣式基礎的分類法可以克服決策樹分類法中的一些限制，例如決策樹分類判別時一次只能考慮一個屬性，研究顯示頻繁樣式基礎的分類法相較於傳統的決策樹分類法有更佳的正確率與可擴充性。

8.4.1 關聯分類法

在本節中，我們學習 CBA, CMAR 與 CPAR 等關聯分類法。在開始敘述這些方法之前，讓我們重新檢視關聯規則探勘的程序，關聯規則探勘分為兩個步驟，首先探勘出頻繁項目集，緊接著由頻繁項目集產生關聯規則。第一個步驟尋找在資料集中頻繁重複出現的樣式 (屬性與值的配對)，而其中每一個屬性與值的配對被視為一個項目 (item)，最終所得到的屬性與值的配對集合成為頻繁項目集，也稱為頻繁樣式集。在第二步驟中我們分析頻繁項目集來產生關聯規則，所有的關聯規則必須滿足特定的標準，包含它的正確率 (亦即它的信賴度) 以及在資料集中出現的比率 (亦即他的支持度) 必須滿足特定的標準，舉例來說，下列為一個關聯規則的例子

$$\begin{aligned}&\text{年齡} = \text{青年} \wedge \text{信用評等} = \text{通過}\\ &\Rightarrow \text{購買電腦} = \text{是}\,[\,\text{支持度} = 20\%，\ \text{信賴度} = 93\%]\end{aligned} \tag{8.21}$$

其中 ∧ 代表邏輯 AND (而且)，關於支持度與信賴度後續會再做說明。

更正式地說，令 D 為資料集合，其中每個資料值組使用 n 屬性 $A_1, A_2, \ldots, A_n$ 來描述，另包括一個類別標籤屬性 A_{class}，所有連續屬性均先進行離散化 (discretized)，並且以類別屬性 (或稱名目屬性) 來處理。一個項目 p 為一個屬性與值的配對，並以 (A_i, v) 來表示，(A_i, v) 代表屬性 A_i 的屬性值為 v。一個資料值組 $\boldsymbol{X} = (x_1, x_2, \ldots, x_n)$ 滿足項目 p，若且唯若 $x_i = v$，其中 x_i 為值組 $\boldsymbol{X}$ 中第 i 個屬性的值。關聯規則左端的前提部分 (antecedent) 可以包含任意數目的項目 (item)，同樣地，關聯規則右端的結論部分 (consequent) 也可以包含任意數目的項目。不過，當利用關聯規則於分類問題時，我們僅對於使用以下形式 $p_1 \wedge p_2 \wedge \ldots \wedge p_l \Rightarrow A_{class} = C$ 的關聯規則有興趣，其中，關聯規則前提部分中項目 $p_1, p_2, \ldots, p_l (l \le n)$ 是以邏輯 AND 連結，並且關聯到一個類別標籤 C。

對於關聯規則 R，其信賴度為資料集 D 內滿足規則 R 前提部分的那些資料值組中，同時亦滿足對應類別標籤的資料值組所佔之比率。從分類的角度觀看，信賴度等同於規則的正確率。舉例來說，關聯規則 (8.21) 的信賴度為 93%，表示顧客屬性為 { 年齡 = 青年 } 而且 { 信用評等 = 通過 } 的顧客中，有 93% 的顧客屬於 { 購買電腦 = 是 } 的類別。關聯規則 R 的支持度為資料集 D 內同時滿足 R 的前提部分以及類別標籤為 C 的資料值組所佔之比率，關聯規則 (8.21) 的支持度為 20%，這說明有 20% 的顧客是 { 年齡 = 青年 } 且 { 信用評等 = 通過 } 以及 { 購買電腦 = 是 } 的類別。

一般而言，關聯式分類包含下列步驟：

1. 探勘資料中的頻繁項目集，也就是說，找出經常出現的屬性與值的配對。
2. 分析頻繁項目集，為每一個類別產生對應的關聯規則，而且這些關聯規則滿足特定的信賴度與支持度標準。

3. 組織那些關聯規則來產生一個規則式分類器。

關聯分類與先前介紹的頻繁樣式探勘的不同，主要在於如何分析與推演這些規則，以及如何使用這些規則來分類資料。我們現在介紹不同的關聯分類方法。

CBA (Classification-Based Association) 是最早期也是最簡單的關聯分類方法是。CBA 使用類似於 5.2.1 節中介紹的 Apriori 的方法進行多層次尋找頻繁樣式集，它疊代式利用當前推演的項目集來產生長度更長的樣式集並且測試它，一般而言，尋找的次數等同於最長規則的長度。所有找出的滿足最小信賴度與支持度門檻值的規則將被分析並納入到分類器中，CBA 利用啟發式方法來建立分類器，這些規則將按照其信賴度 (confidence) 與支持度 (support) 由高至低排序。如果有一群規則具有相同的前提部分，將使用他們當中信賴度最高的規則來代表這群規則。當要分類一個新值組時，將使用第一個滿足這個值組的規則來分類這個值組。此外，CBA 分類器亦包含一個預設規則 (default rule)，它被擺放在規則列表的最後面，當輸入值組並沒有符合任何規則時，預設規則指定了一個預設類別，透過這個方式，這些分類器包含的規則建構一個決策清單 (decision list)，一般來說，CBA 在實證中比 C4.5 有更高的正確率。

CMRA (Classification based on Multiple Association Rules) 與 CBA 的不同在於探勘頻繁項目集與建立分類器的策略，它透過樹狀結構儲存規則，能夠更有效率的存取規則，以及對規則進行修剪。CMAR 利用 FP-growth 修正演算法找出全部滿足最小支持度與信賴度門檻的規則，FP-growth 演算法在 5.2.4 節介紹過，它使用一個樹狀結構 (稱為 FP tree) 來紀錄資料集 *D* 中所有頻繁項目集的資訊，它僅需要掃描資料集 D 二次，之後便從 FP tree 中探勘出頻繁項目集，CMAR 使用增強版 FP tree，對每一個頻繁項目，它記錄所有滿足此頻繁項目的資料值組的類別標籤之分佈情形，透過此方式，能夠跟將探勘頻繁項目集與產生規則合併在一個步驟內完成。

CMAR 使用另一種樹狀結構來存取規則，並且利用信賴度、相互關係 (correlation) 與資料庫涵蓋率 (coverage) 來進行規則修剪。規則修剪的觸發時機點是當有新的規則要插入到樹狀結構時，舉例來說，給定兩個規則

$R1$ 與 $R2$，如果規則 $R1$ 的前提部分比 $R2$ 更通用，同時 $conf(R1) \geq conf(R2)$，則規則 $R2$ 應該給予刪除，其理由是如果存在一個更通用且信賴度更高的規則，則高特殊性與信賴度較低的規則可以被刪除掉。CMAR 同時根據 χ^2 檢定統計顯著性，來刪除掉規則前提與類別標籤沒有正相關的規則，

「如果有一條以上規則被觸發了，我們該使用哪條規則呢？」作為分類器，CMAR 的執行步驟與 CBA 並不相同，假設我們想要分類一個資料值組 X，而且只有一個規則被此值組觸發[3]，則此情況很簡單明確，我們指定此規則的類別標籤給該資料值組即可。然而相反地，假設有超過一條以上的規則被此資料值組觸發時，令這些規則組成集合 S，則該使用哪一條規則來決定資料值組 X 的類別標籤呢？CBA 會使用 S 中信賴度最高的規則來判定資料值組 X 的類別標籤，而相反地，CMAR 則使用多條規則來進行分類預測。它首先依據這些規則的類別標籤將這群規則分組，同一組內的規則有相同的類別標籤，而不同組的規則有不同的類別標籤。

CMAR 使用加權式的 χ^2 量測 (weighted χ^2 measure) 計算同一分組內規則的統計相關性，並找出強度最大的規則分組，然後將強度最大的分組的類別標籤指定給該資料值組。按照此做法，在分類新資料值組時，它考慮數條規則，而非只考慮信賴度最高的單一條規則。在實驗中發現 CMAR 的平均正確率比 CBA 略高，而且它的執行時間、可擴充性以及記憶體空間使用也更有效率。

有沒有辦法降低產生規則的數目呢？CBA 與 CMAR 都是應用頻繁項目探勘的方法來產生候選的關聯規則，它找出所有滿足最小支持度的項目集 (由 AND 聯結的屬性與值的配對 (項目))，這些規則接著被檢驗，並選出其中一小部分來建構分類器，然而這些方法會產生相當大數量的規則。CPAR (Classification based on Predictive Association Rules) 採用不同的方式來建立規則，它根據 7.4.3 節介紹的 FOIL 方法，FOIL 建構出能區隔

[3] 如果一個規則的前提滿足或符合 X，則我們稱這個規則滿足 X，或說 X 觸發此規則。

正值組（例如，屬於｛購買電腦 = 是｝類別的值組）與負值組（例如，屬於｛購買電腦 = 否｝類別的值組）的規則。對於多類別的分類問題，FOIL 方法會應用到每一個類別，也就是說，對於類別 *C*，所有屬於類別 *C* 的資料值組被視為正值組，而剩餘的資料值組被視為負值組，FOIL 產生能夠區隔正值組與負值組的規則，每當產生一條新規則時，滿足此新規則的正值組（也就是說，被此規則涵蓋的正值組）將被移除，直到所有的正值組都被涵蓋時便停止，使用此方法所產生的規則數量可以較少。CPAR 放寬此步驟，它允許被涵蓋的正值組仍被考慮，但是降低它們的權重。此步驟重複對每一個類別執行，而所得到的規則將合併起來建構分類器。

在進行分類時，CPAR 使用與 CMAR 稍微不同的多規則策略，如果新資料值組 *X* 滿足超過一條以上的規則，則將這些規則根據類別來分組（與 CMAR 一樣），然而，CPAR 使用每一分組中預期正確率最好的前 *k* 條規則，而不考慮分組中全部的規則，藉由這樣，可以避免排名較低的規則的影響。實驗顯示在許多資料集上 CPAR 的正確率十分接近 CMAR，然而，由於 CPAR 建立的規則數目遠小於 CMAR，所以它在大型資料集的訓練效率更高。

總結來說，關聯規則由資料集中頻繁出現的屬性與值配對的聯接來建立規則，並提供一種全新的分類機制。

8.4.2 鑑別性頻繁樣式基礎的分類法

藉由關聯分類的進展，我們看見頻繁樣式反映出資料集中屬性與值的配對（項目）的強關聯性，這對分類是很有幫助的。然而，「如何知道用來分類的頻繁樣式，它的鑑別性 (discriminative) 如何呢？」頻繁樣式也可視為特徵的組合，讓我們比較頻繁樣式與個別特徵的鑑別能力，圖 8.11 顯示 UCI 的三個資料集內的頻繁樣式與個別特徵（長度為 1 的樣式）的資訊獲利分佈情形，其中某些頻繁樣式的鑑別能力比個別特徵還要高，這是因為頻繁樣式將資料映射到更高維度的空間，它們擷取到更多資

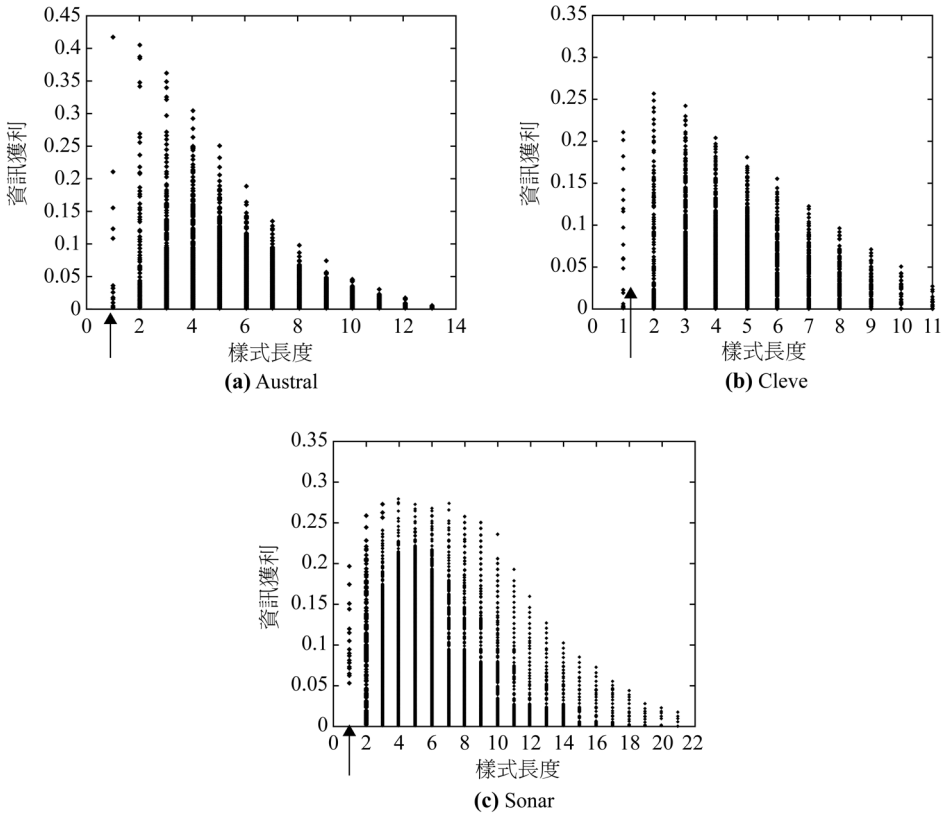


圖 8.11 個別屬性 vs 頻繁樣式的資訊獲利：個別屬性（長度等於 1 的樣式，由箭頭指示）與頻繁樣式（特徵的組合）對於 UCI 中三個資料集上的資訊獲利情形。

料內部的意義，所以比個別特徵具有更好的表達能力。

頻繁樣式基礎的分類法其概念是在由個別屬性與頻繁樣式所建構的特徵空間中學習分類模型，使用此方式，我們將原始特徵空間轉換成維度更大的空間，它能增加找到重要特徵的機會。

讓我們回到之前的問題，頻繁樣式的鑑別能力如何？許多透過頻繁項目集探勘演算法找到的頻繁樣式並沒有鑑別能力，這是因為它們僅考慮支持度，而未考慮到預測能力。也就是說，依照定義必須要滿足使用者定義的最小支持度門檻值 *min_sup* 的樣式，才能算是頻繁樣式，舉例來說，如果 min_sup = 5%，代表一個樣式必須出現在資料值組中 5%，才能算是頻繁樣式，考慮圖 8.12 所示資訊獲利 (information gain) 與樣式頻繁度（支

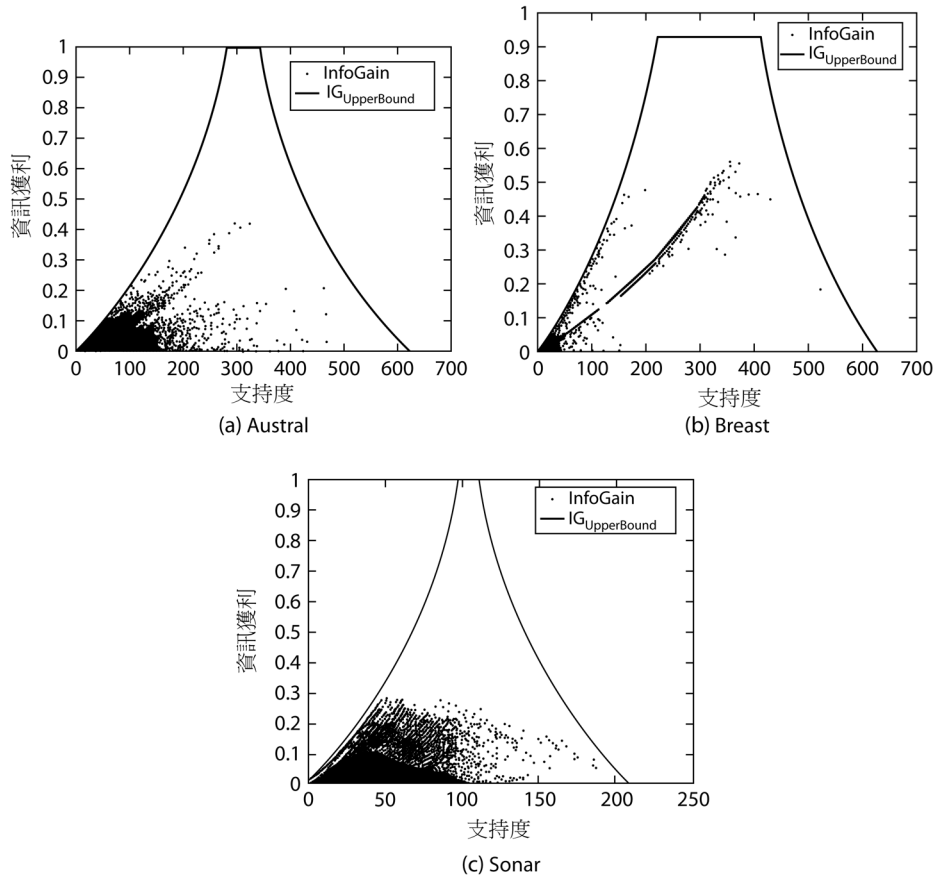


圖 8.12 樣式的資訊獲利 vs 頻繁度 (支持度) 的關係，以 UCI 中三個資料集為例子。

持度) 之間的關係，此處以資訊獲利來估量鑑別能力，其中，資訊獲利的理論上界也同時顯示在圖上，此圖顯示了支持度低的樣式的鑑別能力上界為一個小的數值，這是因為此樣式僅涵蓋極小部分的資料集，同樣地，支持度高的樣式的鑑別能力上界亦為一個小的數值，這是因為它們在資料集中太常見到了，樣式的資訊獲利的上界是跟它頻繁度相關的函數，資訊獲利的上界隨增樣式頻繁度增加而遞增，支持度中高的樣式 (例如，在圖 8.12(a) 中支持度 = 300) 可能有鑑別能力，也可能沒有，所以，並非所有的頻繁樣式都是對分類有幫助的。

如果我們將所有的頻繁樣式加入特徵空間，這會讓特徵空間的維度非常巨大，這會拖緩分類器訓練的速度，也會導致分類器因為過度學習而正

確率降低，由於許多的樣式都是多餘的，因此，使用特徵選取 (feature selection) 的方式來刪除掉鑑別力低的樣式，防止這些多餘的頻繁樣式作為特徵，會是很好的作法。一般而言，鑑別性頻繁樣式基礎分類法的架構如下

1. **特徵產生**：此步驟先將資料集 D 依據類別標籤進行分組，然後對各個分組使用頻繁樣式探勘演算法找出滿足最小支持度門檻值的頻繁樣式，這些頻繁樣式 F 即為候選特徵集合。
2. **特徵選取**：本步驟對候選特徵 F 集合進行特徵選取，以找出更有鑑別能力的頻繁樣式 FS。資訊獲利 (Information gain)、費雪分數 (Fisher score) 或是其他的評量鑑別能力計量都可在此步驟中使用，此外，亦可在本步驟加入關聯性檢查方法來淘汰掉多餘的樣式。接著，將此資料集 D 轉換為 D'，此時特徵空間除了包含個別特徵外，也包含這些選取的頻繁樣式 FS。
3. **學習分類模型**：在資料集 D' 上建立分類器，此步驟可使用任何學習演算法來建立分類器。

鑑別性頻繁樣式基礎分類法的架構摘要在圖 8.13(a)，其中具有鑑別能力的樣式用實心圓圈表示，雖然此作法很直接，但是我們會遭遇到一些計算的瓶頸，因為我們必須找出所有的頻繁樣式，然後逐一分析這些頻繁樣式來選取出具有鑑別能力者，這些頻繁樣式的數目可能十分龐大，因為這些項目的排列組合非常眾多。

要提高鑑別性頻繁樣式基礎分類法的計算效率，可考慮將步驟一和二濃縮成為單一步驟，也就是說，僅探勘出有高鑑別能力的頻繁樣式，而非找出全部的頻繁樣式，此方法也稱為直接鑑別樣式探勘 (direct discriminative pattern mining)，圖 8.13(b) 所示的 DDPMine 演算法便是採用此方式，它首先將訓練資料轉換成一個樹狀結構，也稱為 FP tree (5.2.4 節介紹)，FPtree 保留所有屬性與值配對 (項目) 的關聯資訊，接著便在 FPtree 上搜尋有鑑別能力的樣式，此方法避免了產生大量無鑑別能力的樣式。我們可以進一步藉由移除訓練資料值組來縮減 FPtree，讓探勘過程更為加快。

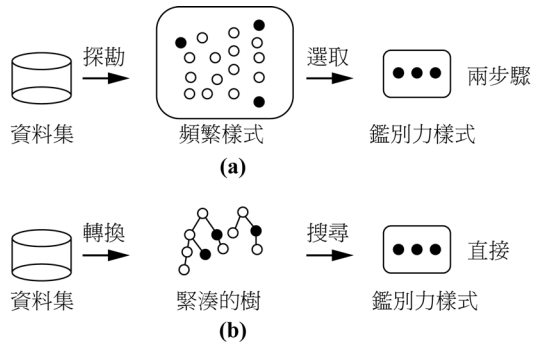


圖 8.13 具鑑別力頻繁樣式為基礎的分類器的架構：(a) 一般使用兩步驟的做法 (b) 使用直接步驟的 DDPMine。

藉由將原始資料轉換成 FPtree，DDPMine 可避免產生多餘的樣式，這是因為 FPtree 只會儲存封閉 (closed) 頻繁樣式，依據定義，封閉樣式 α 的子樣式 β 對於 α 而言是多餘的。DDPMine 直接探勘出有鑑別能力的樣式，並且在探勘過程中整合特徵選取法，而資訊獲利的理論上界可以用來加速分支界限搜尋法 (branch and bound serach)，因為它大幅減少了搜尋空間，實驗結果顯示，相較於分成二個步驟執行，DDPMine 能夠大幅提升速度而不會降低分類正確率，DDPMine 也在效能與正確率方面勝過最先進的關聯分類法。

8.5 偷懶學習 —— 從你的鄰居學習

在本節之前所提的分類方法，例如決策樹、貝式分類器、規則式分類、倒傳遞類神經網路、支持向量機、關聯分類等，全都歸類於是**積極學習** (eager learner)，所謂積極學習表示先利用訓練資料值組建立一個分類模型，往後遇到新 (測試) 值組時，便以此分類器進行預測。我們可以想像此學習者很積極的建立分類器，並且急迫的想要分類未知資料。

想像如果使用一個偷懶的方法，此學習者在遇到未知資料前的最後一刻，它什麼事情都不做，也不預先建立分類器，也就是說，當給定訓練資料時，**偷懶學習** (lazy learner) 僅簡單地把它們儲存起來 (或只對訓練資

料做一些前處理)，只有當它遇到未知資料時，它才根據未知資料與儲存訓練資料的相似度，來推演未知資料的類別。與積極學習不同，偷懶學習在訓練階段做的工作很少，然而在分類或數值預測階段則做很多工作。由於偷懶學習僅儲存訓練資料值組 (也稱為事例 (instance))，所以它也被稱為**事例學習** (instance-based learner)。

偷懶學習在分類或數值預測階段需要大量的計算工作，它需要有效率的儲存技術而且很適合於平行系統架構上執行，它對於資料結構並沒有太多著墨，然而偷懶學習很自然地支持遞增式學習 (incremental learning)，它也能夠建構出複雜的決策空間 (具有超多邊形 (hyperpolygonal))，而這是其他分類方法不能夠輕鬆完成的 (例如決策樹僅能建立超立方體 (hyperrectangular) 形狀的決策空間)。在本節，我們介紹兩種偷懶學習的方法，包含 *k*-最近鄰居法 (*k*-nearest-neighbor) 與案例式推理 (case-based reasoning) 的分類方法。

8.5.1 *k*-最近鄰居分類法

k-最近鄰居分類法最早在 1950 代年初被提出，由於它在大型訓練資料集上需要大量的運算，因此一開始並沒有受到重視，直到 1960 年代電腦的運算能力增加，*k*-最近鄰居分類法才受到注意，如今它已廣泛應用於圖訊辨識領域上。

最近鄰居分類法是根據類推式學習，也就是說，它藉由比對測試資料與跟他相似的訓練資料值組來進行預測，這些訓練值組是由 n 個屬性來描述，我們可以把這些訓練資料值組看作 n-維空間上的一個資料點，利用這種方式，我們可以將所有的訓練資料值組儲存在一個 n-維空間中。當給定一個未知的資料值組時，***k*-最近鄰居分類法** (k-nearest-neighbor classifiers) 在樣式空間中尋找與未知資料值組距離最接近的 k 個訓練資料值組。這 k 個訓練資料值組即為與未知資料值組最接近的 k 個鄰居。

近似性可依據距離來衡量，例如使用歐基里德距離 (Euclidean distance)，給定兩個資料點 $\boldsymbol{X}_1 = (x_{11}, x_{12}, \ldots, x_{1n})$ 與 $\boldsymbol{X}_2 = (x_{21}, x_{22}, \ldots, x_{2n})$ ，

它們之間的歐基里德距離為

$$dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n}(x_{1i} - x_{2i})^2} \quad \textbf{(8.22)}$$

換句話說，對每個數值屬性，我們計算資料值組 $\boldsymbol{X}_1$ 與 $\boldsymbol{X}_2$ 中對應屬性值相差的平方總和再取平方根，一般來說，我們會先對每一個屬性進行正規化 (normalization) 後再根據公式 (8.22) 計算距離，這樣可防止讓屬性值範圍偏大的屬性 (例如收入) 過度影響結果，而輕忽了屬性值範圍較小的屬性 (例如二元屬性) 的影響。例如，可以用 min-max 正規化方法來將數值屬性 A 的值 v 轉換至範圍 [0,1] 中的值 v'，其轉換公式如下：

$$v' = \frac{v - min_A}{max_A - min_A} \quad \textbf{(8.23)}$$

min_A 與 max_A 為數值屬性 A 的最小值與最大值，第 3 章的資料轉換中有介紹其他的正規化方法。

在 k-最近鄰居分類法中，它將未知值組分類至其 k 個最接近鄰居中最常見 (比率最高) 的類別中，當 $k=1$ 時，便指派此未知值組的類別為它最接近的鄰居的類別。最近鄰居法也可應用於數值預測問題中，在此情形中，此預測模型傳回與未知值組最接近的 k 個鄰居所對應的實數值標籤的平均值，作為數值預測的結果。

之前所考慮的皆假設所有屬性均是數值屬性，如果值組中有些屬性不是數值屬性，而是名目屬性 (或類別屬性)，例如顏色屬性，這時該如何計算距離呢？對於名目屬性，最簡單的方式就是當兩個屬性值相同時 (例如資料值組 $\boldsymbol{X}_1$ 與 $\boldsymbol{X}_2$ 都有相同的顏色)，則兩者的差距為 0，如果屬性值不同 (例如資料值組 $\boldsymbol{X}_1$ 的顏色為藍色，而 $\boldsymbol{X}_2$ 的顏色為紅色)，則兩者的差距為 1。我們也可以套用更高階的評估差距的方式，例如藍色與白色的差距比藍色與黑色的差距來得大。

「如果屬性發生遺失值時該怎麼辦？」一般來說，如果資料值組 $\boldsymbol{X}_1$ 與 / 或 $\boldsymbol{X}_2$ 中屬性 A 的值遺失了，我們則假設其差距為最大可能的差距值。舉例來說，假設所有屬性值的範圍都映射在 [0,1] 區間中，如果屬

性 A 為名目屬性，則 $\boldsymbol{X}_1$ 與 $\boldsymbol{X}_2$ 的差距為 1。如果屬性 A 為數值屬性，假如 $\boldsymbol{X}_1$ 或 $\boldsymbol{X}_2$ 的屬性值均為遺失了，則令其差距值為 1。如果 $\boldsymbol{X}_1$ 與 $\boldsymbol{X}_2$ 之中只有一個的屬性值遺漏，而另一個的屬性值為 v'，則令其差距值為 $|1-v'|$ 或 $|0-v'|$ 中的最大值。

「如何決定最佳的 k 值（也就是鄰居的數目）呢？」它可透過實驗來決定，一開始設定 $k=1$，我們使用一個測試資料集來評估分類器的錯誤率，接著我們每次疊代均把 k 的值增加 1，也就是多增加一個鄰居，能夠擁有最小錯誤率的 k 值，便是最佳的 k 值。一般來說，訓練資料集越大，則最佳的 k 值也會相對的變大。假如訓練值組的數目為無限大，則當 $k=1$ 時，k-最近鄰居分類法的錯誤率不會超過貝氏錯誤率（Bayes error rate，分類錯誤率理論上的最小值）的兩倍，而當 k 趨近無限大時，k-最近鄰居法的錯誤率上界會趨近貝氏錯誤率。

最近鄰居分類法使用距離來進行比對，並且把每個屬性視為一樣重要，因此當遇到雜訊或是無關的屬性時，他必須承受正確率較差的困擾。我們可以藉由給每個屬性不同的權重值並且刪除資料集中的雜訊來修正此問題。除此之外，選擇恰當的距離度量法 (distance metric) 也是非常重要的，曼哈頓距離（Manhattan distance，2.4.4 節）或其他的距離量度法也可以應用在最近鄰居分類法中。

最近鄰居法在分類測試值組時是非常緩慢的，如果 D 為訓練資料集，其中包含 $|D|$ 個值組，並且設定 $k=1$，要分類一個測試值組需要執行 $O(|D|)$ 的比對，藉由預先排序並把值組編排成搜尋樹 (search tree) 的結構，我們可以將比對的次數降低至 $O(\log(|D|))$。如果使用平行處理技術，我們可以將執行時間降低至常數 $O(1)$，此時執行時間與值組數目 $|D|$ 無關。

其他增加最近鄰居分類速度的技術包含使用部分距離 (partial distance) 與編輯 (editing) 儲存值組。在部分距離方法部分，我們使用 n 個屬性的子集合來計算距離，如果部分距離超過一個門檻值，我們則停止該值組後續的計算，並前往處理下一個儲存的值組。而編輯儲存值組方法是移除不需要的訓練值組，它也被稱為**刪除** (pruning) 或**濃縮** (condensing) 方法，因

為它會降低全部儲存值組的數目。

8.5.2 案例式推理

案例式推理 (case-based reasoning, CBR) 利用一個資料庫儲存過去問題的解決方案，再根據類比推演的方法，由類似的舊問題解決方案中嘗試解決新的問題。不同於最近鄰居分類法將資料值組儲存成為歐基里德空間(Euclidean space)中的一個資料點，CBR 將問題解決方案的值組 (或稱案例) 儲存成更複雜的符號化敘述。CBR 在商業上的應用如顧客服務，此時案例為描述與產品相關的診斷問題；CBR 也可應用於工程與法律問題上，此時案例分別為技術設計與法律裁決；醫療看護也是 CBR 另一個應用的領域，此時過去病患診斷與治療的歷史案例可以用來幫助治療新病患。

當要分類一個新給定的案例，CBR 首先檢查是否有一模一樣的歷史 (訓練) 案例存在，如果找到了，那麼此訓練案例所伴隨的解決方案將會傳回。如果找不到一模一樣的訓練案例，則 CBR 會去尋找與新案例有某些部分相似的訓練案例，這些訓練案例可視為新案例的鄰居。如果案例是使用圖形結構表達，那即表示要尋找出跟新案例有相同子圖形 (sub-graph) 結構的訓練案例。CBR 會嘗試整合類似的 (鄰居) 訓練案例們的解決方案，並提供該解決方案給新的案例。如果那些訓練案例伴隨的解決方案並不一致，此時 CBR 必須能夠回溯再去尋找其他的解決方案，CBR 可以套用背景知識與問題解決策略來提供合理的解決方案。

CBR 的挑戰在於找出好的相似度測量尺度 (similarity metric)、合適的解決方案整合方法，CBR 其它的挑戰也包含如何選擇顯著的特徵來將訓練案例進行索引 (index)，以及發展有效率的索引技術。當儲存的案例數量變得非常多時，CBR 也必須面臨正確率與執行效率之間的取捨。當案例儲存數目增加，CBR 推理變得更有智慧，但相對的，一但儲存案例數量過多，CBR 也會花更多的時間來搜尋與處理相關的案例，而導致執行效率變差。如同最近鄰居分類法，其中一個解決方案是編輯 (濃縮) 案例資料庫，我們可以將多餘與無用的案例加以刪除，以增加執行效率。然而，如何自

動地決定哪些案例應該予以刪除，仍是案例式推理研究的議題。

8.6 其他分類方法

這一節我們概略地介紹其他分類的方法，包含基因演算法、粗糙集合與模糊集合等方法。一般來說，與之前所介紹的分類法相比，這些方法較少應用於商業資料探勘系統，然而這些方法在特定應用上有其優勢，因此仍值得在此節介紹它們。

8.6.1 基因演算法

基因演算法 (genetic algorithms) 嘗試融入自然進化 (evolution) 的概念，一般來說，基因演算法的執行步驟如下：隨機產生一組規則來建立初始**族群** (population)，每條規則皆用位元字串 (string of bits) 表示。舉一個簡單的例子，假設訓練資料集中的樣本皆由兩個布林屬性 A_1 與 A_2 來描述，而且有兩個類別 C_1 與 C_2，規則 "*IF* A_1 *AND NOT* A_2 *THEN* C_2" 可以編碼成位元字串 100，最左邊兩個位元分別代表屬性 A_1 與 A_2，而最右邊的位元則代表類別。相同地，規則 "*IF NOT* A_1 *AND NOT* A_2 *THEN* C_1" 可以編碼成位元字串 001。如果一個屬性有 k 個值而且 $k > 2$，則可以用 k 個位元來編碼此屬性的值。類別部分也可以使用相同的編碼技巧。

根據適者生存的自然法則，目前族群中最合適的規則，以及它們的後代 (off-spring)，將繁衍成新的族群。一般而言，一條規則的**合適度** (fitness) 通常是由它在訓練資料分類的正確率來估計。後代是經由**交配** (crossover) 與**突變** (mutation) 等演化運算來產生。在交配過程中，我們將一對規則中的子字串進行交換，然後重組成一對新的規則。而在突變過程中，則是隨機選取規則中的任一個位元進行反相 (invert)。持續重複此步驟，不斷地根據上一代族群繁衍出更合適族群，直到新一代族群的合適度滿足使用者預設的門檻值為止。

基因演算法很容易可以擴充到平行化運算，它可應用於分類或其它最

佳化的問題，在資料探勘上，它也可用於評估其他演算法的合適度。

8.6.2 粗糙集合

粗糙集合理論 (rough set theory) 可用來發掘出不精確與雜訊資料集中的結構以幫助分類工作，它僅適用於離散值屬性，連續值屬性資料在使用前必須先離散化。

粗糙集合理論的基礎是在訓練資料集上建立**等價類別** (equivalence classes)，在等價類別中的所有資料值組是不可被分辨的，也就是說，這些資料值組的屬性值皆為相同。給定現實資料集，常會發現有某些類別無法由可用的屬性予以分辨，此時便可使用粗糙集合來粗略地定義 (近似) 這些類別。類別 *C* 的粗糙集合的定義是由兩個集合來逼近的：類別 *C* 的**下界近似集** (lower approximation) 與類別 *C* 的**上界近似集** (upper approximation)。類別 *C* 的下界近似集包含那些基於對屬性的知識，確認必定會屬於類別 *C* 的所有資料值組，而類別 *C* 的上界近似集包含那些基於對屬性的知識，無法認定會不屬於類別 *C* 的資料值組 (亦即有可能會屬於類別 *C* 的所有資料值組)，類別 *C* 的下界與上界近似集如圖 8.14 所示，其中每一個長方形代表一個等價類別。我們可以為每一個類別建立決策規則，通常我們會使用決策表 (decision table) 來表達這些規則。

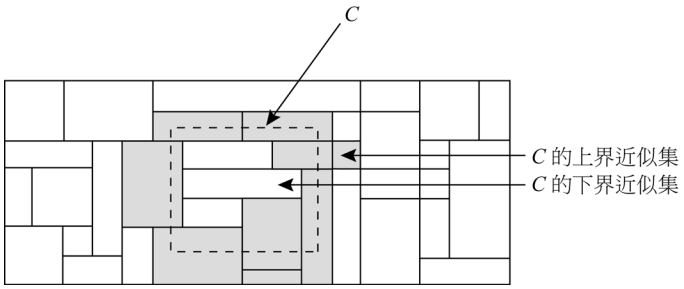


圖 8.14 利用類別 C 的下界與上界近似集來定義類別 C 的粗糙集合，其中每一個長方形區域代表一個等價類別。

粗糙集合也可以幫助屬性選取 (亦即特徵刪減 (feature reduction)，把那些對於分類任務沒有貢獻的屬性予以辨認出，並加以刪除) 與相關性分析 (分析估計每一個屬性對於分類任務的貢獻程度與重要性)，找出能描述給定資料集中所有概念的最小 (簡約) 屬性集合是一個 NP-hard 的問題，然而，仍有能夠減低它的計算複雜度的演算法被提出來。其中一個方法是使用**差別矩陣** (discernibility matrix) 來儲存每一對資料值組間的屬性值的差異值，不用搜尋全部資料集合，我們可以使用差別矩陣來代替以偵測出多餘的屬性。

8.6.3 模糊集合

規則式分類系統的缺點是它對於連續值屬性採取明確尖銳的截斷 (cutoff)，以顧客信用卡申請應用為例子，下列規則說明顧客申請信用卡通過的門檻，就是工作年資要有兩年或兩年以上，並且要有高收入 (收入最少 5 萬元)。

$$IF\ (\text{工作年資} \geq 2)\ AND\ (\text{收入} \geq 50{,}000)\ THEN\ \text{信用卡} = \text{通過} \quad \textbf{(8.24)}$$

依據規則 (8.24)，一個年資兩年以上的顧客並且薪水超過 50,000 者將能申請到信用卡，反之薪水為 49,000 的則無法通過申請，如此嚴苛的門檻似乎並不公平。

相反地，我們可以將連續值屬性 { 收入 } 分割成不同類別，例如 { 低收入，中收入，高收入 }，然後使用模糊集合 (fuzzy set) 理論來允許這些類別之間存在有模糊的門檻 (或是邊界)，如圖 8.15 所示。相對於非 0 即 1 的明確截斷，模糊集合允許使用 0.0 到 1.0 之間的數值來表示它隸屬於某一集合的歸屬程度 (membership degree)，這些集合便稱為模糊集合，因此，使用模糊集合理論，收入 49,000 雖然不及 50,000，但我們可以將收入 49,000 視為“些許”高的收入。模糊邏輯系統通常會提供圖形化工具來幫助將連續值屬性轉換成模糊集合。

模糊集合理論也稱為**可能性理論** (possibility theory)，它是由 Lotfi Zadeh 在 1965 年提出，來取代傳統的布林邏輯與機率理論，它提供讓我們

能夠在更高層抽象概念下面去處理不精確的資料的工具，更重要的是，它讓我們能夠處理現實世界下曖昧的與不精確的現象，舉例來說，屬於高收入本身就是不精確的，如果收入超過 50,000 才算是高收入，那麼收入 49,000 算高收入嗎？收入 48,000 算高收入嗎？不同於傳統的明確 (crisp) 集合，元素只有屬於或不屬於這兩種情形，模糊集合理論允許一個元素屬於一個以上的模糊集合，舉例來說，收入 49,000 同時屬於高收入與中收入這兩個模糊集合，只是隸屬程度並不一樣，使用圖 8.15 所定義的模糊集合，我們可以看到

$$m_{\text{中收入}}(\$49{,}000) = 0.15 \text{ 與 } m_{\text{高收入}}(\$49{,}000) = 0.96$$

m 代表中收入與高收入這兩個模糊集合的歸屬函數 (membership function)。在模糊集合理論中，一個元素 x (例如收入 49,000) 隸屬於各個模糊集合的歸屬程度值加總不需要等於 1，這是與一般機率理論不同的地方。

模糊集合理論對於規則式分類非常有用，它提供模糊集合的運算與模糊邏輯推演的工具。假設除了收入之外，我們將年資屬性區分為 { 資淺員工，資深員工 } 這兩個模糊集合。現今有一條規則，在前提部分 (IF 部分) 對高收入與資深員工進行測試，並把測試的結果用 AND 連結起來，則它屬於高收入的歸屬程度與屬於資深員工的歸屬程度，會取最小值，做為此規則前項部分成立的歸屬程度，換句話說

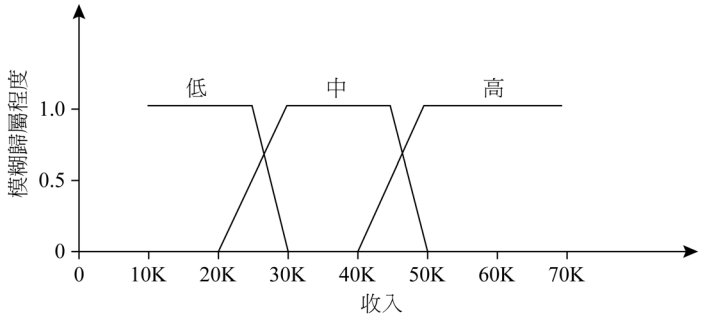


圖 8.15 使用歸屬函數來定義收入隸屬於 { 低收入，中收入，高收入 } 這三個模糊集合的歸屬程度。請注意，給定一個收入值 x，它可能隸屬於一個以上的模糊集合，而且它屬於各個模糊集合的歸屬函數值的加總並不需要等於 1。

$$m_{(\text{高收入 } AND \text{ 資深員工})}(x) = min(m_{\text{高收入}}(x), m_{\text{資深員工}}(x))$$

相反的，如果是把測試的結果用 OR 連結起來，則是取最大值做為此規則前項部分成立的歸屬程度。

$$m_{(\text{高收入 } OR \text{ 資深員工})}(x) = max(m_{\text{高收入}}(x), m_{\text{資深員工}}(x))$$

當要分類一個資料值組時，我們可以代入數條模糊規則，每一條規則皆會貢獻出它預測此值組屬於某類別的歸屬程度，這些歸屬程度會加總起來合併成為一個模糊的輸出結果，這輸出的結果也是一個模糊集合，有數種技術可以將模糊輸出結果進行解模糊化 (defuzzified) 動作，還原成為一個明確值並回傳給系統。模糊系統已經應用在市場研究、財務金融、健康照顧與環境工程等分類問題上。

8.7 其他分類議題

大多數的分類演算法能夠處理多類別分類的問題，但少數演算法，例如支持向量機，只能夠處理二元分類的問題，該如何擴展它們去處理多類別分類呢？我們將在 8.7.1 節介紹多類別分類。

另外，如果我們只有少數的資料擁有類別標籤，而絕大多數的資料並沒有，我們該如何利用這些資料來建立分類器呢？文件分類、語音辨識與資訊擷取皆是含有大量的無類別標籤資料的範例，以文件分類為例，假設我們想要建立一個自動化的文件分類器，能夠分類文章或網頁等資料，好比分類曲棍球與足球這兩類別的文件，我們手邊有龐大數量的文件，但都沒有標記出類別標籤，回顧監督式學習 (如分類) 需要訓練資料，也就是具有類別標籤的資料值組集合，而用人工的方式去檢視與標記每一個文件的類別標籤以建立訓練資料集，是耗時且成本昂貴的工作。8.7.2 節介紹半監督式分類演算法，它能在同時混合有類別標籤與無類別標籤的資料集合中，建立分類器，因此適用於處理含有大量無類別標籤資料的情況。

8.7.1 多類別分類

某些分類演算法，例如支持向量機，是專門設計來處理二元分類的問題，我們該如何延伸這些演算法來處理多類別分類的問題呢？

一個簡單的方式是**一對多** (one-versus-all, OVA)，給定 m 個類別，我們訓練 m 個二元分類器，每一個分類器都是針對一個類別來訓練，分類器 j 的訓練資料集是將所有屬於類別 j 的值組視為正類別，而剩餘的值組視為負類別。要分類一個未知資料 $\boldsymbol{X}$ 時，每一個分類器投票決定此資料屬於哪一個類別，舉例來說，如果分類器 j 預測 $\boldsymbol{X}$ 屬於正類別，則類別 j 獲得一票，反之，若分類器預測 X 屬於負類別，則除了類別 j 之外的所有類別都獲得一票，最終得票數最高的類別，便是我們預測資料 $\boldsymbol{X}$ 所歸屬的類別。

另一種方式是**多對多** (all-verus-all, AOA)，我們為每一對的類別來建立分類器，給定 m 個類別，我們則建立 $m(m-1)/2$ 個分類器，每一個分類器都是要區分一對不同的類別，要分類一個未知值組，則每一個分類器都進行投票，得票數最高的類別便是此值組歸屬的類別。一般來說，多對多法的分類效果會比一對多法更優異。

上述方式的缺點是對二元分類器的錯誤是很敏感的，如果任一個分類器預測錯誤了，那會影響投票表決的結果。

錯誤更正碼 (error-correcting code) 可以提升多類別分類的正確率，不僅對如支持向量機等二元分類器可以使用，一般的分類器一樣可以適用。錯誤更正碼最初的設計，是用來解決在通訊問題中，更正資料在傳輸過程中發生的錯誤，對此問題，我們將要傳輸的資料增加額外的更正碼，以便即使在傳輸的過程中，因為通道的雜訊而使傳輸資料發生一些錯誤，接收端還是能接收到正確的資料。對多類別分類問題而言，使用錯誤更正碼能夠幫助我們在即使少數二元分類器發生預測錯誤的情況下，依然能正確地預測未知值組的類別標籤。

要應用錯誤更正碼於多類別分類問題中，我們首先為每一個類別指定一個錯誤更正碼，錯誤更正碼是一個位元向量 (bit vector)，圖 8.16 顯示一

個使用錯誤更正碼的例子，我們對 C_1, C_2, C_3 與 C_4 四個類別分別指定一個 7 位元的代碼 (錯誤更正碼)，對每一個位元我們建立一個對應的二元分類器，因此在這個例子中，我們將訓練 7 個分類器，如果其中一個分類器預測失誤了，根據這些額外的位元所得到的資訊，我們依然有很好的機會能正確預測未知值組的類別，這裡我們使用一種計算位元向量距離的技術，稱為漢明距離 (Hamming distance)，並在錯誤發生時，預測它是屬於最“接近”的類別，詳細的作法在範例 8.3 中說明。

範例 8.3 **應用錯誤更正碼於多類別分類**

考慮指派 7 位元的代碼 (錯誤更正碼) 給 C_1 至 C_4 四個類別，如圖 8.16 所示，假設給定一個未知的值組，訓練好的 7 個分類器預測的結果為 0001010，此代碼並未符合任一個類別的代碼，很明顯地，有分類器發生預測錯誤了，但是我們可以依據漢明距離猜測它最可能是哪一個類別，漢明距離是二個位元向量之間擁有不同位元的個數，分類器輸出的代碼與類別 C_1 的代碼之間的漢明距離為 5，因為它們有 5 個位元是不一樣的，分別為第 1, 2, 3, 5 與 7 個位元，同樣地，分類器輸出的代碼與類別 C_2 的代碼之間的漢明距離為 3，與 C_3 的漢明距離為 3，與 C_4 的漢明距離為 1，注意分類器輸入的代碼與 C_4 的代碼是最接近的，因為它們之間的漢明距離最小，所以我們預測未知值組是屬於類別 C_4 。

類別	錯誤更正碼 (代碼)
C_1	1 1 1 1 1 1 1
C_2	0 0 0 0 1 1 1
C_3	0 0 1 1 0 0 1
C_4	0 1 0 1 0 1 0

圖 8.16 使用錯誤更正碼來解決多類別 (四個類別) 的分類問題。

錯誤更正碼最多能更正 $(h-1)/h$ 個位元的錯誤，其中 h 是任兩個代碼之間漢明距離的最小值，如果我們設計的代碼是一個位元對應一個類別的話，則它就等同於一對多法 (one-verous-all)，而此代碼是沒有自我更正 (self-correct) 的能力。使用錯誤更正碼來解決多類別問題，各個類別所指派的代碼之間的漢明距離是越大越好，漢明距離越大代表有越多的錯誤可以被更正。

8.7.2 半監督式分類法

半監督式分類法 (semi-supervised classification) 使用有標記 (類別標籤) 與無標記的資料來建立分類器，令 $X_l = \{(x_1, y_1, \ldots, (x_l, y_l))\}$ 為有標記類別標籤的資料集，$X_u = \{x_{l+1}, \ldots, x_n\}$ 為無標記類別標籤的資料集，這裡我們介紹幾種作法

自我學習 (self-training) 是半監督式分類法中最簡單的方法，它首先利用有標記類別標籤的資料集來建立分類器，此分類器接著嘗試預測無標記資料值組的類別標籤，預測結果最為信賴的值組，便將該值組與對應的預測類別標籤，一起加入到有標記類別標籤的資料集中，然後繼續重複上述的步驟 (如圖 8.17 所示)，雖然此方法很簡單明瞭，但它的缺點是可能會擴散錯誤。

協同學習 (cotraining) 是另一種半監督式的分類法，它使用二個或更多個分類器來彼此教導，每一個分類器使用值組內不同與互相獨立的特徵集合來學習，以網頁資料為例，描述網頁上圖片的屬性可視為一組特徵集合，而與網頁文字內容相關的屬性可構成另一組特徵集合 (在同樣的網頁資料上)，每一組特徵集合都足夠能訓練出好的分類器。假設我們將特徵集合切割成二個部分，分別用來訓練分類器 f_1 與 f_2，接著，分類器 f_1 與 f_2 可用來預測無標記資料值組的類別標籤，f_1 將它預測結果最為信賴的值組連同其預測的類別標籤，加入到 f_2 有標記類別標籤的資料集中，同樣地，f_2 將它預測結果最為信賴的值組連同類別標籤，加入到 f_1 有標記類別標籤的資料集中，透過這種方式，這兩個分類器彼此教導，也彼此學

自我學習

1. 選擇一個學習演算法，例如說貝式分類器，使用有標記類別標籤的資料集 X_l 來建立分類器。
2. 使用訓練好的分類器來預測未標記資料集 X_u 中資料值組的類別標籤。
3. 選擇信賴度最高的值組 $x \in X_u$，將它與預測的類別標籤加入到 X_l 中。
4. 重覆上步驟 (亦即，使用擴充後的有標記資料集重新訓練分類器)。

協同學習

1. 定義有標記資料集 X_l 上兩個不重疊的特徵集合。
2. 使用有標記資料集訓練兩個分類器，f_1 是使用其中一個特徵集合來訓練，f_2 則是使用另一個特徵集合來訓練。
3. 使用 f_1 與 f_2 來預測無標記資料集 X_u 中值組的類別標籤。
4. 把 f_1 預測信賴度最高的 $(x, f_1(x))$ 加入到 f_2 所使用的有標記訓練資料集中，其中 $x \in X_u$，同樣地，把 f_2 預測信賴度最高的 $(x, f_2(x))$ 加入 f_1 所使用的有標記訓練資料集中。
5. 重複上步驟 (亦即，f_1 與 f_2 分別使用其擴充後的有標記資料集重新訓練分類器)。

圖 8.17　半監督式分類中的自我學習法與協同學習法。

習，詳細的步驟如圖 8.17 所示。相較於自我學習，協同學習對於錯誤較不敏感，它困難的地方在於它的假設不見得會成立，也就是說，有可能無法將特徵集合切割成互斥且彼此類別條件獨立的群組。

還有其他的半監督式學習方法，舉例來說，我們可以建構特徵與類別標籤的聯合機率分佈 (joint probability distribution) 模型，對於無標記類別標籤的資料，其類別標籤可視為遺失值處理，EM 演算法 (第 10 章) 可用來最大化概似 (likelihood) 估計此機率模型，此外，也有學者提出使用支持向量機的方法。

8.8 模型評估與選取

現在你已經建構好一個分類模型，你腦海中可能會浮現出許多問題，舉例而言，假設你使用以前的銷售資料建構好一個分類器，並用它來預測

顧客購買行為，你希望評估此分類器預測未來顧客購買行為（也就是未用來訓練分類器的顧客資料）的正確率，你甚至可能嘗試基於不同方法來建構數個分類器，並希望比較它們的正確率。但是，正確率是什麼？該如何估計它？是否有某些分類器的正確率評估量測比其他量測更為適合？我們該如何得到可靠的正確率估計？這些問題都將在這章節討論。

8.8.1 節介紹數種預測分類器正確率的評估度量，保留與隨機子取樣（8.8.2 節），交叉驗證（8.8.3 節），與 Bootstrap 法（8.8.4 節）都是基於對給定資料集執行隨機分割採樣，來估計正確率的常用方法。假設我們有數個分類器，並想要選取最好的一個，這問題稱為**模型選取** (model selection)，最後兩個小節將討論此議題，8.8.5 節探討如何使用統計顯著性來檢驗兩個分類器之間正確率的差異是否純屬偶然，8.8.6 呈現如何根據成本收益 (cost-benefit) 與接收者操作特徵曲線（receiver operating characteristic curve，或叫 ROC 曲線）來比較分類器的效能。

8.8.1 評估分類器效能的度量

此章節呈現估計你的分類器在預測資料值組的類別標籤的效能有多麼好（或多麼“正確”），我們將考慮不同類別的值組是大致均勻分佈，或是類別不平衡的狀況（例如，有興趣的重要類別的值組數量稀少，好比醫療檢測資料），本章節呈現的分類器評估量測摘要在圖 8.18 中，它們包含正確率（accuracy，也稱為辨識率 (recognition rate)）、敏感度（sensitivity，也稱招回率 (recall)）、特異性 (specificity)、精確率 (precision)、F_1 與 F_β，請注意，雖然正確率意旨特定的度量，但是“正確率”這個名詞也可用來表示分類器的預測能力的一般術語。

使用訓練資料來推導分類器，並且用它們來估計學習得到的模型的正確率，可能會導致過分樂觀而令人誤導的估計結果（我們將在稍後探討此議題），相對地，分類器的正確率最好是使用包含類別標籤的測試資料集（未用來訓練分類模型）來估計。

量測	公式
正確率 (accuracy)，辨識率 (recognition rate)	$\frac{TP+TN}{P+N}$
錯誤率 (error rate)，分類錯誤率 (misclassification rate)	$\frac{FP+FN}{P+N}$
敏感度 (sensitivity)，真陽率 (true positive rate)，招回率 (recall)	$\frac{TP}{P}$
特異性 (specificity)，真陰率 (true negative rate)	$\frac{TN}{N}$
精確率 (precision)	$\frac{TP}{TP+FP}$
F, F_1, F-分數，精確率與招回率的調和均值	$\frac{2\times precision\times recall}{precision+recall}$
F_β 其中 β 是非負實數	$\frac{(1+\beta^2)\times precision\times recall}{\beta^2\times precision+recall}$

圖 8.18 評估量測，注意某些量測中的符號 *TP*、*TN*、*FP*、*FN* 分別代表真陽性、真陰性、偽陽性、偽陰性的值組數目 (見文章解釋)。

在深入探討各種量測之前，我們必須熟悉一些專門術語，回顧我們之前所談論的**正值組** (所感興趣的主要類別的值組) 與**負值組** (其他值組)[4]，舉例來說，給定兩個類別，我們可以稱屬於「購買電腦=是」的為正值組，而負值組是對應於「購買電腦 = 否」，假設在具有類別標籤的測試值組集合中，P 為正值組的數目，而 N 為負值組的數目，對每一個測試值組，我們比較分類器預測的類別標籤與真實的類別標籤。

另外還有四個重要的術語必須知道，它們是計算許多評估量測所需要的“構件”，理解它們能有助於洞悉各種量測的含意：

- **真陽性** (true positives, TP)：係指被分類器正確預測的正值組，令 TP 代表真陽性的值組數目。
- **真陰性** (true negatives, TN)：係指被分類器正確預測的負值組，令 TN 代

[4] 在機器學習與圖訊識別文獻中，它們分別稱為正樣本 (positive sample) 與負樣本 (negative sample)。

表真陰性的值組數目。

- **偽陽性** (false positives, FP)：它們是負值組，但錯誤地被分類器預測為正值組，(例如，屬於「購買電腦 = 否」的值組，被分類器預測為「購買電腦 = 是」)，令 FP 代表偽陽性的值組數目。
- **偽陰性** (false negatives, FN)：它們是正值組，但錯誤地被分類器預測為負值組，(例如，屬於「購買電腦 = 是」的值組，被分類器預測為「購買電腦 = 否」)，令 FN 代表偽陰性的值組數目。

這些術語摘要在圖 8.19 的**混淆矩陣** (confusion matrix) 中。

對於分析你的分類器對辨識不同類別的值組的效果有多麼好，混淆矩陣是很有幫助的，*TP* 與 *TN* 告訴我們分類器何時判斷正確，而 *FP* 與 *FN* 則是說明分類器何時判斷錯誤。給定 m 個類別 ($m \geq 2$)，混淆矩陣是一個至少為 $m \times m$ 的表格，而前 m 行與前 m 列中的儲存格單元 CM_{ij}，代表第 i 類別的值組被分類器預測為第 j 類別的數目，對於擁有良好正確率的分類器，理想地，絕大多數的值組會出現在混淆矩陣的對角線上，從 $CM_{1,1}$ 至 $CM_{m,m}$，而其餘的儲存格單元之值是零，或接近於零，也就是說，理想地，*FP* 與 *FN* 是趨近零。

此表格有額外的列與行，來提供總計的資訊，例如在圖 8.19 中，最後一行顯示 P 與 N，而最底端列的 P' 為分類器預測為正類別的值組總數 ($TP + FP$)，而 N'為分類器預測為負類別的值組總數 ($TN + FN$)，全部值組的總數為 $TP + FP + TN + FN$，或 $P + N$ 或 $P' + N'$，請注意，雖然這裡顯示的混淆矩陣是針對二元分類，混淆矩陣可以使用相似的方法，輕易的延伸到多類別分類的問題上。

		預測類別		
		是	否	總計
真實類別	是	*TP*	*FN*	*P*
	否	*FP*	*TN*	*N*
	總　計	*P′*	*N′*	*P* + *N*

圖 8.19　混淆矩陣，顯示正值組與負值組的總合。

現在，讓我們從正確率開始，來逐一檢視這些評估量測。分類器的正確率 (accuracy) 是在給定的測試值組集合中，被此分類器正確分類的值組所佔的百分比，也就是說

$$accuracy = \frac{TP + TN}{P + N} \tag{8.25}$$

在圖訊識別的文獻中，它也稱為分類器的辨識率 (recognition rate)，即它反映出分類器正確辨識各個類別的效能，對應於兩個類別「購買電腦=是」與「購買電腦 = 否」的混淆矩陣範例顯示在圖 8.20 中，總共顯示對每一個類別與整體的辨識率，藉由審視混淆矩陣，可以輕易的一眼看出分類器是否混淆了這兩個類別。

舉例來說，我們可以看到 412 個“否”類別的值組被誤判為“是”類別了，當類別分佈是大致平衡時，正確率是最有效的評估量測。

我們也可簡單地說分類器 M 的錯誤率 (或分類錯誤率) 即為 1-*accuracy* (M)，其中 *accuracy* (M) 為分類器 M 的正確率，它也可由下式計算出來

$$error\ rate = \frac{FP + FN}{P + N} \tag{8.26}$$

如果我們使用訓練資料集 (而非測試資料集) 來評估分類模型的正確率，此度量值也稱為**重新代入誤差** (resubstitution error)，此誤差估計對於真實的錯誤率是過度樂觀的 (同樣地，其對應的正確率估計也是過度樂觀的)，因為此分類模型不是由它前所未見的資料樣本來評估的。

類別	購買電腦 = 是	購買電腦 = 否	總計	辨識率 (%)
購買電腦 = 是	**6954**	**46**	7000	99.34
購買電腦 = 否	**412**	**2588**	3000	86.27
總計	7366	2634	10,000	95.42

圖 8.20 對應於兩個類別「購買電腦 = 是」與「購買電腦 = 否」的混淆矩陣，其中第 *i* 列第 *j* 行的單元顯示，第 *i* 類別的值組被分類器預測為第 *j* 類別的數目，理想上，非對角線的單元應該為零，或接近於零。

我們現在考慮**類別分佈不平衡** (class imbalance) 的問題，其中我們所感興趣的主要類別內的資料值組數目是極稀少的。舉例來說，在詐欺偵測問題中，我們所感興趣的類別 (即正類別) 為“詐欺”，它比負類別 (非詐欺的) 出現的頻率罕見許多。在醫學資料集中，對應於“癌症”類別的資料是很稀少的，假設你想要訓練一個分類器來判別醫學資料，其中類別標籤屬性是“癌症”，而其可能的值為“是”或“否”，分類器的正確率為 97%，可說是相當正確的，但是如果只有 3%的資料值組屬於真正罹患癌症時會怎樣呢？明顯地，正確率 97% 可能是不被接受的，舉例來說，此分類器可能僅正確地分類“非癌症”的值組，但是把所有“癌症”的值組都錯誤分類了，有鑑於此，我們需要其他的量測，能夠評估正值組 (癌症 = 是) 辨識的有多良好，而負值組 (癌症 = 否) 又辨識的有多良好。

為了達到此目的，我們可以分別使用敏感度 (sensitivity) 與特異性 (specificity) 這兩個量測，敏感度對應於真陽 (辨識) 率，亦即正樣本被正確辨識的比率，而特異性對應於真陰 (辨識) 率，亦即負樣本被正確辨識的比率，這些量測定義為

$$sensitivity = \frac{TP}{P} \tag{8.27}$$

$$specificity = \frac{TN}{N} \tag{8.28}$$

可以證明正確率是敏感度與特異性所組成的函數

$$accuracy = sensitivity\frac{P}{P+N} + specificity\frac{N}{P+N} \tag{8.29}$$

範例 8.4 **敏感度與特異性**

圖 8.21 顯示對於醫學資料的混淆矩陣，其中類別標籤屬性為“癌症”，類別值為“是”與“否”，分類器的敏感度是 $90/300 = 30.00\%$，特異性是 $9560/9700 = 98.56\%$，整體正確率為 $9650/10{,}000 = 96.50\%$，

因此，我們可以注意到，雖然分類器擁有高正確率，但是它正確地分類正類別的能力是很差勁的，因為它的敏感度很低。它擁有高特異性，代表它能正確地分類負值組。

類別	是	否	總計	辨識率 (%)
是	**90**	**210**	300	30.00
否	**140**	**9560**	9700	98.56
總計	230	9770	10,000	96.50

圖 8.21 對於「癌症 = 是」與「癌症 = 否」的混淆矩陣。

精確率與招回率也廣泛用於分類量測中，精確率 (precision) 可視為**精準度**的量測 (判別為正類別的值組中，真實是正類別所佔的比率)，而招回率 (recall) 可視為**完整性**的量測 (真實是正類別的值組中，被判別為正類別所佔的比率)，如果你覺得招回率似曾相識，那是因為它與敏感度 (或真陽率) 是一樣的，這些量測的計算公式為

$$precision = \frac{TP}{TP + FP} \tag{8.30}$$

$$recall = \frac{TP}{TP + FN} = \frac{TP}{P} \tag{8.31}$$

範例 8.5 精確率與招回率

在圖 8.21 中，對於“是”類別的精確率是 $90/230 = 39.13\%$，而招回率是 $90/300 = 30.00\%$，它與範例 8.4 中的敏感度計算是一樣的。

對於類別 C 而言，完美的精確率是 1.0，這代表每一個被分類器判別是類別 C 的值組，真正都是屬於類別 C 的，但是，它沒有告訴我們，真正是類別 C 的值組中，有多少個是被分類器錯誤判別的。對於類別 C，完美

的招回率 1.0 代表每一個類別 C 的值組，都被分類器正確地判別為類別 C，但是，它沒有告訴我們，不屬於類別 C 的值組中，有多少個是被錯誤地判別為類別 C 的。精確率與招回率傾向於有反向的關係，提升其中一者的代價，將會降低另一者。舉例來說，我們的醫療分類器可以藉由將特定方式表達的值組判別為癌症，而提升了精確率，但是如果此分類器對很多癌症值組判別錯誤，則可能會有較低的招回率。精確率與招回率分數通常一起使用，其中對招回率固定來比較精確率，或反之亦然。舉例而言，我們可以在招回率固定為 0.75 時，比較精確率。

另一個替代方式是將精確率與招回率合併在一起，成為單一量測，這便是 F-量測（也稱為 F_1 分數或 F-分數）與 F_β 量測的作法，它們定義為

$$F = \frac{2 \times precision \times recall}{precision + recall} \tag{8.32}$$

$$F_\beta = \frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall} \tag{8.33}$$

其中 β 為非負實數，F-量測是精確率與招回率的調和均值，它對精確率與招回率給予相同的權重，F_β 量測是精確率與招回率的加權量測，它指定招回率的權重是精確率的 β 倍，常用的 F_β 量測是 F_2（招回率的權重是精確率的 2 倍）與 $F_{0.5}$（精確率的權重是招回率的 2 倍）。

「正確率是否還有在某些狀況是不適當的嗎？」在分類問題中，通常會假設所有資料值組都是唯一可分類的，也就是說，每一個資料值組僅會屬於單一個類別，然而，由於大型資料庫內資料分佈的多樣性，假設所有值組皆是唯一可分類的，並不是永遠合理的。相反地，假設每個資料可以屬於多個類別，會更為恰當。但是，這樣該如何量測分類器的正確率呢？傳統正確率量測在此處並不適當，因為它沒有考慮到值組屬於多個類別的可能性。

相對於僅傳回單一類別標籤，回傳資料值組可能屬於各個類別的機率分佈，是更為有用的，正確率量測可以使用**二次猜測** (second guess) 方

案，如果預測的類別與最可能或是次一可能類別是一致的，則評斷此次的分類是正確的。雖然它在某種程度上，是有顧慮到值組非唯一分類的問題，但它並不是完整解。

除了正確率為主的量測外，分類器仍應就底下各方面來比較其效能：

- **速度** (speed)：係指建構與使用分類器所需的計算成本。
- **強健性** (robustness)：係指在給定雜訊資料與遺失值的情況下，分類器能做出正確預測的能力。通常，是給定一系列的人造資料集，代表增加雜訊與遺失值的程度，來估計分類器的強健性。
- **可擴充性** (scalability)：係指在給定龐大數量的資料之情況下，能有效率建構分類器的能力。通常，是給定一系列的資料集，並逐步增加資料的數目，來評估分類器的可擴充性。
- **可解讀性** (interpretability)：係指分類器 (或預測器) 可提供的理解與洞悉的層級，可解讀性是主觀的，所以不易評估，決策樹與分類規則可能較容易解讀，但是隨著它們變得更複雜，易解讀的優點也會消失。我們在 8.2.4 節中所探討的，從類神經網路的黑箱中萃取出分類規則，便是這領域的課題。

總結來說，我們提出許多評估分類模型的量測，當類別是較為均衡的分佈時，正確率的效果最好，其他的量測，例如敏感度 (亦即招回率)、特異性、精確率、F 與 F_β 量測，則較適用於類別分佈不平衡的情況下，亦即其中感興趣的主要類別是較為稀少的。剩餘的章節將聚焦於如何得到可靠的分類正確率估計。

8.8.2 保留與隨機子取樣

保留法 (holdout) 是迄今我們討論正確率時，所暗指的方法。在此方法中，給定的資料集被隨機分割成兩個獨立的集合：訓練集 (training set) 與測試集 (test set)。通常，2/3 的資料分配至訓練集中，而剩餘的 1/3 的資料則配置給測試集，訓練集是用來推導分類模型，而分類模型的正確率則是在測試集上作評估 (如圖 8.22)。此估計是悲觀的，因為初始資料中，

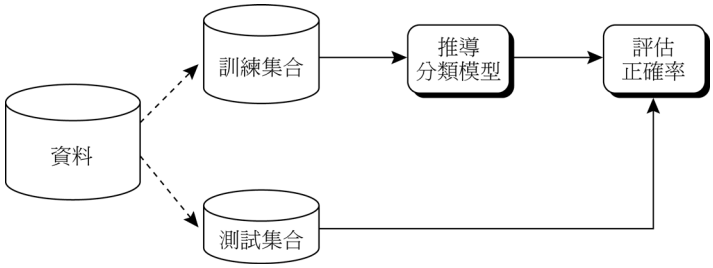


圖 8.22　使用保留法來估計正確率。

只有一部分是用推導分類模型。

隨機子取樣法 (random subsampling) 是保留法的變異，它重複執行保留法 k 次，並將每一次疊代得到的正確率加總之後取平均值，作為整體的正確率估計。

8.8.3　交叉驗證

k-折交叉驗證 (k-fold cross-validation) 將初始資料集隨機的分割成 k 個互斥的子集合 (也稱為“折 (fold)”)，這 k 個子集合 $D_1, D_2, \ldots, D_k$ 的大小是近似相等的。訓練與測試會執行 k 次，在第 i 次疊代，子分割 D_i 被當作測試集，而剩餘的分割被收集整合起來，用來訓練分類模型。也就是說，在第 1 次疊代，子集合 $D_2, D_3, \ldots, D_k$ 被合併起來做為訓練集，用它們來訓練分類模型，並在子集合 D_1 上面測試，到第 2 次疊代，在 $D_1, D_3, \ldots, D_k$ 上面訓練分類模型，並在 D_2 上面測試，依此類推。不同於保留法與隨機子取樣法，在交叉驗證中，每一個資料樣本用作訓練的次數是一樣的，而且有一次是用來測試。對於分類問題，整體的正確率估計是將 k 次疊代中正確分類的數量除以初始資料集中值組的數量而得到。

留一驗證 (leave-one-out) 是 k-折交叉驗證的特例，其中 k 是初始值組的數目，也就是說，每一次它只“保留”一個資料樣本作為測試集。在**分層交叉驗證** (stratified cross-validation) 中，每一“折”被分層，使得每一折中的值組的類別分佈，是與初始資料集中值組的類別分佈是近似相同的。

一般而言，較為建議採用分層 10-折交叉驗證來估計正確率 (即便有足夠的計算能力使用更多的“折”)，因為它具有相對較小的偏差與變異量。

8.8.4 Bootstrap 法

不同於之前提及的正確率估計方法，bootstrap 法對於給定的訓練資料值組，採取可放回的均勻採樣策略，也就是說，每一次選取一個資料值組時，此值組有相同的機率會再被選取到，並再次添加入訓練集中。舉例來說，想像隨機的選取值組進入訓練集中，在可放回採樣 (sampling with replacement) 策略中，我們允許相同的值組被選取數次。

有許多種 bootstrap 方法，最常用的是 **.632 bootstrap** 法，它的執行流程如下，假設給定一個擁有 d 個值組的資料集，此資料集被採樣 d 次 (可放回)，得到 bootstrap 採樣集 (即擁有 d 個值組的訓練集)，非常有可能某些原始資料值組，將在採樣集中出現一次以上，而那些最終沒有添加入訓練集的值組，即被收集起來組成測試集。假設我們執行這樣的採樣策略數次，結果會有平均 63.2% 的原始資料值組，會添加進入 bootstrap 採樣集，而剩餘的 36.8% 的值組會組成測試集 (因此，它命名為 .632 bootstrap 法)。

「63.2% 的數字是從何而來的呢？」每一個值組被選取的機率是 $1/d$ ，所以沒有被選取到的機率是 $(1-1/d)$，我們必須選取 d 次，所以一個值組最終沒有被選取到的機率是 $(1-1/d)^d$ ，如果 d 值很大，它的機率會接近於 $e^{-1}=0.368$，所以有 36.8% 的值組不會被選取到訓練集中，最終做為測試集，而剩餘的 63.2% 的值組組成訓練集。

我們可以重複此採樣程序 k 次，在每一次疊代中，我們使用當前的測試集，來估計從當前的 bootstrap 採樣集建構的分類模型的正確率，則分類模型 M 的整體正確率為

$$Acc(M)=\frac{1}{k}\sum_{i=1}^{k}(0.632\times Acc(M_i)_{test\_set}+0.368\times Acc(M_i)_{train\_set}) \qquad \textbf{(8.34)}$$

其中 $Acc(M_i)_{test\_set}$ 是從第 i 個 bootstrap 採樣集建構的分類器在第 i 個測試集上的正確率，而 $Acc(M_i)_{train\_set}$ 是從第 i 個 bootstrap 採樣集建構的分類器在原始資料集上的正確率。Bootstrap 法傾向於非常樂觀，而且在小型資料集上執行的非常良好。

8.8.5 使用統計顯著性檢定來模型選取

假設我們已經建構好兩個分類模型 M_1 與 M_2，我們也已經執行 10-折交叉驗證來得到平均錯誤率。我們該如何決定哪一個分類模型是最好的？看來最直覺的方法，是選取錯誤率最低的分類模型，然而，平均錯誤率僅估計在真實分佈的未來資料上的錯誤，在 10-折交叉驗證實驗中所得到的錯誤率，可能存有相當可觀的變異量，雖然 M_1 與 M_2 得到的平均錯誤率可能不同，但是其差異可能不是統計顯著的，如果它們的差異只是純屬偶然，那該怎麼辦？本章節將探討這些問題。

為了決定兩個分類模型的平均錯誤率是否“真正”是有差異的，我們必須執行統計顯著性檢測，此外，我們希望獲得平均錯誤率的信賴界限 (confidence limit)，使得我們可以提出以下的論述「對於未來的樣本，則 95% 的時間裡面，觀察到的平均值不會超過正負兩個標準差」，或者是「一個分類模型比另一個分類模型好，其誤差幅度為 ±4%」。

我們需要什麼以執行統計檢測？假設對每一個分類模型，我們執行 10 折交叉驗證，共執行 10 次，每一次我們使用不同的 10 折資料分割，每一次分割都是獨立抽取，我們可以分別為 M_1 與 M_2 的 10 個錯誤率取平均值，以得到每一個分類模型的平均錯誤率。對一個給定的分類模型，它由交叉驗證計算得來的每一個別錯誤率，可以視為來自一個機率分佈中的不同的獨立樣本，一般而言，它們符合具有 $k-1$ 自由度的 t- 分佈，此處 $k=10$ (此分佈看來很像常態或高斯分佈，即便它們兩者的分佈函數很不類似，它們都是單峰的、對稱的、與鐘形的)。這允許我們執行統計檢定，其顯著性檢驗可以使用 ***t*-檢定**或**學生 *t*-檢定** (student t-test)，我們的假說是這兩個分類模型是一樣的，或換句話說，兩者在平均錯誤率的差異

為零。如果我們可以否定此假說 (稱為虛無假說 (null hypothesis))，則我們可以斷定這兩個分類模型的差異是統計顯著的，在此情況下，我們應該選取錯誤率較低分類模型。

在實行資料探勘時，我們通常使用單一測試集，也就是說，M_1 與 M_2 使用相同的測試集，在此情況下，我可以在每一輪的 10 折交叉驗證中，對這兩個模型執行**逐對比較** (pairwise comparison)，也就是說，在 10 折交叉驗證的第 i 輪，我們對 M_1 與 M_2 使用相同的交叉驗證分割資料，來得到它們的錯誤率，令 $err(M_1)_i$ (或 $err(M_2)_i$) 為分類模型 M_1 (或 M_2) 在第 i 輪的錯誤率，對 M_1 的錯誤率取平均值，得到 M_1 的平均錯誤率，記為 $\overline{err}(M_1)$，同樣地，我們可以得到 $\overline{err}(M_2)$，兩個模型的差異的變異量標記為 $var(M_1 - M_2)$，t-檢定從 k 個樣本計算具有 $k-1$ 自由度的 t-統計量。在我們範例中，我們有 $k=10$，因為此處的 k 個樣本，是我們對每一個模型執行 10 折交叉驗證所得到的錯誤率，逐對比對的 t-統計量計算方式如下

$$t = \frac{\overline{err}(M_1) - \overline{err}(M_2)}{\sqrt{var(M_1 - M_2)/k}} \qquad \textbf{(8.35)}$$

其中

$$var(M_1 - M_2) = \frac{1}{k}\sum_{i=1}^{k}\left[err(M_1)_i - err(M_2)_i - (\overline{err}(M_1) - \overline{err}(M_2)) \right] \qquad \textbf{(8.36)}$$

要決定 M_1 與 M_2 的差異是否為顯著的，我們計算 t-統計量並選取顯著水準 (significance level) sig。在實作中，通常會使用 5% 或 1% 為顯著水準，我們接著查閱 t-分佈表格 (標準統計教科書皆會提供)，此表格通常以不同列來表示各個自由度，不同行來表示各個顯著水準，當我們要確定 M_1 與 M_2 差異的顯著不同是否有 95% 時，亦即 $sig = 5\%$ 或 0.05，我們需要找出表格中對應 $k-1$ 自由度的 t-分佈值 (在我們範例中，自由度為 9)。然而，由於 t-分佈是對稱的，通常只顯示上半部的百分比點，因此，我們尋找 $z = sig/2$ 的表格值，在此例子中為 0.025，其中 z 又稱為**信賴界**

限 (confidence limit)，如果 $t > z$ 或 $t < -z$ ，則 t 值落在拒絕區域，亦即在分佈的尾端，這代表我們可以拒絕代表 M_1 與 M_2 是相同的虛無假說，並斷定說這兩個分類模型的差異是統計顯著的。否則，如果我們不能否定虛無假說，則代表 M_1 與 M_2 的差異可能歸因於偶然。

如果有兩個測試集可以使用，而非是只有一個，則使用非逐對版本的 t-檢定，其中兩個模型的平均值的差異之變異量可由下式估計

$$var(M_1 - M_2) = \sqrt{\frac{var(M_1)}{k_1} + \frac{var(M_2)}{k_2}} \qquad \textbf{(8.37)}$$

其中 k_1 與 k_2 分別是 M_1 與 M_2 所使用的交叉驗證的樣本數 (在我們例子中，是 10-折交叉驗證的回合數)。這也稱為**兩樣本的 *t*-檢定** (two sample t-test)，當查閱 t-分佈的表格時，所使用的自由度，是這兩個模型的自由度的最小者。

8.8.6 根據成本收益與 ROC 曲線來比較分類器

真陽率 (TP)、真陰率 (TN)、偽陽率 (FP) 與偽陰率 (FN) 也可用於估計分類模型的成本與收益 (或風險與報酬)，對應於偽陰率的成本 (例如錯誤地預測癌症病患為健康的) 是遠大於偽陽率的成本 (錯誤地預測健康病患為罹患癌症的)，在此情況下，我們可以藉由指派不同的錯誤類型有不同的成本，而給予某類型的錯誤有更多的權重。這些成本可能是考慮對病患的危害程度、治療的開銷以及醫院的其他成本。同樣地，對應於真陽率的成本，也可能與真陰率的不同。至今為止，要計算分類器的正確率，我們僅假設每一個錯誤類型有相同的成本，而只將真陽例與真陰例的總合除以測試值組的總數而已。

相對於此，我們可以融入成本與收益的概念，轉而計算這些預測決定的平均成本 (或獲益)。其它涉及成本與獲益的範例，還包含貸款申請決策與廣告信件目標行銷，舉例來說，貸款給一個會拖欠債務者所要付出的成本，遠超過拒絕貸款給一個不會拖欠債務者，所導致遺失商機的代價。同樣地，寄廣告信件給那些不會回覆的住家所付出得成本，可能大於不寄

廣告信件給那些會回覆的住家，所導致的商機遺失。在整體分析中，其它要考慮的成本，還包含蒐集資料與發展分類器所要付出的成本。

接收者操作特徵曲線 (receiver operating characteristic curve，或叫 ROC **曲線**) 是比較兩個分類模型的有用的視覺化工具，ROC 曲線源於信號偵測理論，是在二次世界大戰時為了分析雷達圖像而開發的。對於給定的分類模型，ROC 曲線顯示在真陽率 (true positive rate, *TPR*) 與偽陽率 (false positive rate, *FPR*) 之間的權衡，給定測試集與分類模型，*TPR* 是正 (或“yes”) 值組被模型正確預測的比率，*FPR* 是負 (或“no”) 值組被模型錯誤預測為正值組的比率，令 *TP* 、 *FP* 、 *P* 與 *N* 分別為真陽、偽陽、正與負值組的數目，我們有 $TPR = TP/P$ ，它也是敏感度 (sensitivity)，此外， $FPR = FP/N$ ，它也等同於 1- sensitivity。

對於雙類別分類問題，ROC 曲線允許我們視覺化地檢視測試集的不同部分，分類模型能正確辨識正樣本的比率對應它錯誤判別負樣本為正樣本的比率之間的權衡，*TPR* 增加的代價是增加 *FPR* ，在 ROC 曲線下的面積也可用來量測分類模型的正確率。

要畫出給定分類模型 *M* 的 ROC 曲線，對於每一個測試值組，該模型必須要能夠回傳其預測類別的機率分佈。使用此資訊，我們可以對測試值組進行評次與排序，使得最有可能屬於正類別 (或“yes”類別) 的值組，排序在序列的頂端，而最不可能屬於正類別的值組，則位在序列的尾端。樸素貝氏分類與倒傳遞類神經網路分類器能回傳每一次預測值組的類別時，它預測的類別的機率分佈，因此很適合於 ROC 曲線，至於其他的分類器，例如決策樹，可以輕易的修改來使得他能回傳預測的類別機率值。假設一個機率式分類器對給定值組 $\boldsymbol{X}$ 所回傳的值是 $f(\boldsymbol{X}) \to [0,1]$，對於二元分類問題，通常會選取一門檻值 t ，使得具有 $f(\boldsymbol{X}) \geq t$ 的值組將被考慮為屬於正類別，而其餘的值組即考慮為負類別，請注意，真陽與偽陽值組的數目都是隨 t 而改變的函數，因此可以把他們寫成 $TP(t)$ 與 $FP(t)$ ，這兩者都是單調遞減的函數。

我們首先描述繪製 ROC 曲線背後的概念，並伴隨一個例子說明，ROC 曲線的垂直軸代表 *TPR*，水平軸代表 *FPR*，要繪製分類模型 *M* 的 ROC 曲

線，我們的步驟如下，從左下角開始 (此處 $TPR = FPR = 0$)，我們檢查在序列頂端的值組的真正類別標籤，如果它是真陽值組 (亦即，被正確分類的正值組)，則 *TP* 與 *TPR* 增加，在圖形上，我們往上移動，並且繪製一點；如果相反地，它是偽陽值組 (亦即，分類器將負值組錯誤預測為正值組)，則 *FP* 與 *FPR* 將增加，在圖形上，我們往右移動，並且繪製一點。對於排序後的每一個測試值組，此步驟根據每一個測試值組的排次順序不斷重複，每一次，若為真陽則往上移動，若是偽陽則往右移動。

範例 8.6 繪製一個 ROC 曲線

圖 8.23 顯示機率式分類器對測試集中的 10 個值組所回傳的機率值 (在第 3 欄)，他們以機率值遞減的順序來排列，第 1 欄是值組的編號，它可幫助我們解釋，第 2 欄是此值組真正的類別標籤，這裡有 5 個正值組與 5 個負值組，因此，$P = 5$ 與 $N = 5$，隨著我們檢驗此值組的類別標籤，我們能夠決定其餘欄位上的 *TP*、*FP*、*TN*、*FN*、*TPR* 與 *FPR* 之值。我們從值組 1 開始，它具有最高的機率分數，並把門檻值設定為 $t = 0.9$，則此分類器將值組 1 分類為正類別，而其餘的值組被判別為負類別。由於值組 1 它真正的類別標籤為 "*P*"，故它為 *TP*，因此 $TP = 1$ 與 $FP = 0$，至於其它的 9 個值組，全都被分類成負類別，其中真正有 5 個是負類別 (所以 $TN = 5$)，而剩餘的 4 個實際上是正類別，所以 $FN = 4$。因此，我們可以計算 $TPR = TP / P = 1/5 = 0.2$，而 $FPR = 0$，所以我們在 ROC 曲線上繪製一點 $(0.2, 0)$。

接下來，門檻值 *t* 設定為 0.8，它是值組 2 的機率值，所以此值組現在被判別為正類別，而值組 3 至 10 被分類為負類別，值組 2 真正的類別標籤為 "*P*"，所以現在 $TP = 2$，剩餘欄的值可以輕易計算出，所以得到 $(0.4, 0)$ 這一點。接著，我們檢驗值組 3 的真正類別標籤，並且設定 $t = 0.7$，它是分類器對值組 3 所回傳的值，因此值組 3 被判別為正類別，但值組 3 真正的類別標籤是 "*N*"，故它是偽陽的，所以 *TP* 維持不變，但 *FP* 增加，而得到 $FP = 1$，剩餘欄位上的值可以輕易計算出，故

得到 (0.4 , 0.2) 此點。隨著檢驗每一個值組，可以得到 ROC 圖形，它是一個鋸齒狀的圖形，如圖 8.24 所示。

有許多方式可以從這些點來取得曲線，最常用的是凸包 (convex hull)。為了方便比較，圖形中也顯示一條對角線，在此模型中每遇到一個真陽值組，皆恰好遇到一個偽陽值組，它代表著隨機猜測。

值組編號	類別	機率	*TP*	*FP*	*TN*	*FN*	*TPR*	*FPR*
1	*P*	0.90	1	0	5	4	0.2	0
2	*P*	0.80	2	0	5	3	0.4	0
3	*N*	0.70	2	1	4	3	0.4	0.2
4	*P*	0.60	3	1	4	2	0.6	0.2
5	*P*	0.55	4	1	4	1	0.8	0.2
6	*N*	0.54	4	2	3	1	0.8	0.4
7	*N*	0.53	4	3	2	1	0.8	0.6
8	*N*	0.51	4	4	1	1	0.8	0.8
9	*P*	0.50	5	4	1	0	1.0	0.8
10	*N*	0.40	5	5	0	0	1.0	1.0

圖 8.23　根據機率分數遞減排序的值組，其分數是由機率式分類器的回傳值。

圖 8.25 顯示兩個分類模型的 ROC 曲線，代表隨機猜測的對角線也顯示在其中，所以，如果分類模型的 ROC 曲線越接近對角線，此模型的正確率越差。如果此分類模型非常好，初始時，隨著從序列向下移動，我們很有可能遭遇真陽值組，所以此曲線從零點開始陡峭地上升，之後，我們遇到的真陽值組越來越少，而遇到越來越多的偽陽值組，此曲線上升平緩，變得更為水平。

為了評估分類模型的正確率，我們量測曲線底下包含的區域之面積，有許多軟體套件可以幫助我們執行此計算，此區域的面積越接近 0.5，其對應的分類模型正確率越差，具有完美正確率的分類模型，它的區域面積會為 1.0。

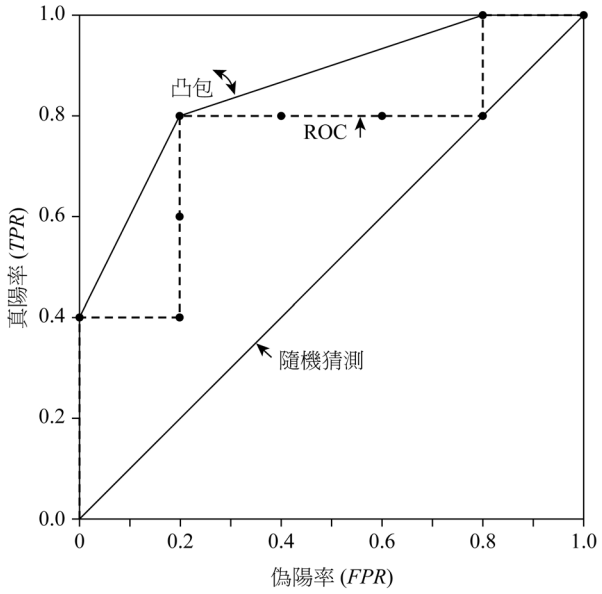


圖 8.24 圖 8.23 內展示資料的 ROC 曲線。

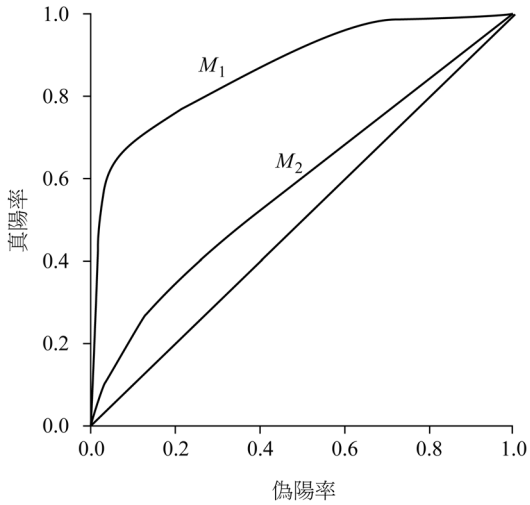


圖 8.25 兩個分類模型 M_1 與 M_2 的 ROC 曲線，此對角線顯示對每一個真陽值組，都有相同的可能遭遇一個偽陽值組，分類模型的 ROC 曲線越接近對角線，它的正確率越差勁，因此，此處 M_1 的正確率較高。

8.9 總結

- 不同於樸素貝氏分類器 (naïve Bayesian classifier) 假設類別條件獨立，**貝氏信念網路** (Bayesian belief networks) 允許我們定義變數之間類別條件相依的關係，並提供一個可學習的圖形式模型來表達這種因果關係，訓練過的貝氏信念網路可以用來分類資料。
- **倒傳遞** (backpropagation) 是類神經網路的學習演算法，它利用梯度降低法來調整網路聯結上的權重，以最小化網路預測結果與真正目標值間的均方誤差值。我們可以從訓練過後的類神經網路中萃取出規則，來提升類神經網路的可解讀性。
- **支持向量機** (support vector machine) 可解決線性可分割與線性不可分割的分類問題，它將原始資料映射到更高維度的特徵空間，並且在特徵空間上使用重要的訓練值組 (稱為支持向量) 來建構最大邊界的超平面以分割兩個類別。
- 頻繁樣式反應了屬性與值的配對 (也稱為項目 (item)) 存在有強關聯性，**頻繁樣式基礎的分類法**便是透過頻繁樣式來分類資料，它包括關聯分類法與鑑別性頻繁樣式基礎的分類法，**關聯分類法** (associative classification) 透過頻繁樣式所產生的關聯規則來建構分類器；**鑑別性頻繁樣式基礎的分類法** (discriminative frequent pattern-based classification) 則是將頻繁樣式視為特徵的合成，並將它們與個別特徵一起用來建構分類器。
- 決策樹分類器、貝式分類器、倒傳遞類神經網路、支持向量機、關聯分類等全都屬於**積極學習** (eager learner)，它們使用訓練資料建立分類模型，並準備好分類未知資料。與它相反的是**偷懶學習** (lazy learner)，也稱**事例學習** (instance-based learner)，例如**最近鄰居分類法** (nearest neighbor classifier) 與**案例式推理** (case-based reasoning, CBR)，它們僅將所有訓練資料儲存起來，便偷懶等待，直到要被分類的值組出現才進行

預測工作，因此，偷懶學習需要對儲存資料更有效率的資料索引技術。

- 在**基因演算法** (genetic algorithm) 中，上一代族群藉由交配與突變兩種演化運算不斷地進化成新的族群，直到族群內所有分類規則的合適度都符合使用者設定門檻值才停止；**粗糙集合理論** (Rough set theory) 可用來近似定義無法被屬性予以分辨的類別；**模糊集合理論** (fuzzy set theory) 使用歸屬函數來取代傳統利用門檻值將連續值屬性進行尖銳明確的截斷。
- 二元分類器 (例如支持向量機) 也可以延伸到解決多類別分類的問題，它的作法便是將數個二元分類器整合在一起，此外，錯誤更正碼 (error-correcting code) 可用來增加多類別分類的正確率。
- **半監督式分類** (semi-supervised classification) 在含有大量無標記類別標籤的資料存在時，是很有用的，它同時使用有標記與無標記 (類別標籤) 的資料集來建立分類器，半監督式學習的範例包含**自我學習** (self-training) 與**協同學習** (cotraining)。
- **混淆矩陣** (confusion matrix) 可用來評估分類器的品質，對於兩類別分類問題，它顯示了真陽率、真陰率、偽陽率與偽陰率，可以用來衡量分類器預測能力的量測包含：正確率、敏感度 (也稱為招回率)、特異性、精確率、F 與 F_β 量測，過度依賴正確率可能受到蒙蔽，尤其當主要感興趣的類別僅占少數時。
- 欲建構與評估分類器，我們必須將擁有類別標籤的資料值組分割成兩部分：訓練集合與測試集合，**保留** (holdout) 與**隨機子取樣** (random sampling)、**交叉驗證** (cross-validation) 與 Bootstrap **法**是典型的資料值組分割方法。
- 顯著性檢定與 ROC 曲線可用於模型選取，**顯著性檢定** (significance test) 可以用來估計兩個分類器的正確率的差距，是否歸因於偶然。ROC **曲線**可以繪製出單一或數個分類器的真陽率 (或敏感度) 對應於與偽陽率 (或 1-specificity) 之間的權衡。

8.1 下列表格包含員工的訓練資料，資料使用廣義方式表示，例如，年齡“31..35”代表年齡落在 31 到 35 之間，而每一列的數目欄位，代表符合該列中部門、狀態、年齡與薪資值的員工數目，令「狀態」為類別標籤屬性。

部　門	狀　態	年　齡	薪　資	數　目
Sales	senior	31...35	46 千...50 千	30
Sales	junior	26...30	26 千...30 千	40
Sales	junior	31...35	31 千...35 千	40
systems	junior	21...25	46 千...50 千	20
systems	senior	31...35	66 千...70 千	5
systems	junior	26...30	46 千...50 千	3
systems	senior	41...45	66 千...70 千	3
marketing	senior	36...40	46 千...50 千	10
marketing	junior	31...35	41 千...45 千	4
secretary	senior	46...50	36 千...40 千	4
secretary	junior	26...30	26 千...30 千	6

(a) 為此資料集設計一個多層前饋式類神經網路，並標示輸入與輸出層的網路節點。

(b) 使用 (a) 所建立的多層前饋式類神經網路，顯示餵入一訓練資料 (sale, senior, 31…35, 46-50K) 進行倒傳遞演算法一次疊代後，網路聯結的權重值為何？請指出你設定的權重與偏移量的初始值，以及學習速率設定為多少。

8.2 支持向量機 (SVM) 擁有優異的分類正確率，但是 SVM 分類器在大型資料集上進行訓練卻是十分費時的，請討論如何克服此問題，並設計一個可擴充式的

SVM 演算法能有效率地在大型資料集上進行分類。

8.3 比較關聯分類法與鑑別性頻繁樣式基礎的分類法，為何使用頻繁樣式為基礎的分類器能在許多狀況下獲得比傳統決策樹更優異的分類正確率呢？

8.4 比較積極學習 (如決策樹、貝式分類器、類神經網路) 與偷懶學習 (如最近鄰居法、案例式推理) 的優點與缺點。

8.5 給定 k (最近鄰居的數目) 與 n (資料屬性的數目)，寫一個執行 k-最近鄰居分類的演算法。

8.6 簡明的描述使用 (a) 基因演算法 (b) 粗糙集合 (c) 模糊集合來進行分類的程序。

8.7 範例 8.3 顯示如何使用錯誤更正碼來進行多類別 (四個類別) 的分類

(a) 假設給定一個未知的資料值組，這 7 個二元分類器輸出的代碼為 0101110，它並未完全符合這四個類別的代碼，使用錯誤更正程序，此未知值組的類別標籤應該為何？

(b) 請說明為何代碼長度為 4 位元時，便無法執行錯誤更正？

8.8 證明正確率是敏感度與特異性的函數，亦即，證明公式 (8.29)。

8.9 調和平均值 (harmonic mean) 是計算平均值的一種方法，第 2 章介紹過如何計算算術平均值，這是大多數人所知道的平均值計算方法，對於正實數 $x_1, x_2, \ldots, x_n$，其調和平均值 H 為

$$H = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}} = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$

F 量測是精確率與招回率的調和平均值，請使用此項事實來推導公式 (8.32)，此外，將 F_β 改寫為 TP、FN 與 FP 的函數。

8.10 圖 8.26 的資料值組是依據分類器所回傳的機率值遞減排列而成，對每個值組，計算它的真陽值組、偽陽值組、真陰值組與偽陰值組的數目，計算它們的真陽率 (FPR) 與偽陽率 (FPR)，並繪製該資料的 ROC 曲線。

值組編號	類　別	機　率
1	*P*	0.95
2	*N*	0.85
3	*P*	0.78
4	*P*	0.66
5	*N*	0.60
6	*P*	0.55
7	*N*	0.53
8	*N*	0.52
9	*N*	0.51
10	*P*	0.40

圖 8.26　根據機率分數遞減排序的值組，其分數是由機率式分類器的回傳值。

8.11 如果個別資料值組一次可以屬於多個類別，要評估分類正確率是一項很困難的任務，在此情況下，請對你用來評估在同一資料集中比較不同分類器的準則作出評論。

8.12 假設你想要從兩個預測模型 M_1 與 M_2 中做出取捨，我們已經對每一個分類模型執行 10 輪的 10 折交叉驗證，每一輪，這兩個分類模型皆是使用相同的資料分割，M_1 得到的錯誤率為 30.5, 32.2, 20.7, 31.0, 41.0, 27.7, 26.0, 21.5, 26.0，而 M_2 所得到的錯誤率為 22.4, 14.5, 22.4, 19.6, 20.7, 20.4, 22.1, 19.4, 16.2, 35.0。請評論在顯著水準 1% 的情況下，哪一個分類模型是顯著比另一個分類模型好？