類神經網路

6

悅知文化
Delight Press

2006 年 12 月 5 日，德國波昂所舉行為期十天的國際西洋棋大賽終於落幕，這場由「人腦」世界西洋棋王克拉姆尼克，以及電腦「深沉德國兵」西洋棋軟體的六場 PK 大賽，最後電腦以二勝四和再度贏了人腦。2002 年兩方首次對戰，當時人腦以 4：4 與電腦打平。

四年的光陰對電腦來說，有如「天上一天，人間一年」，新版西洋棋軟體的運算能力提升至原來的兩倍以上，可以每秒同步計算 800 萬至 1000 萬步棋路，因此在對弈過程中，電腦完全沒有任何失誤，只要人腦一鬆懈就能夠趁隙獲勝。

自從 IBM 於 1997 年推出的超級電腦「深藍」，打敗了當時的世界棋王卡斯帕洛夫之後，電腦運用它最大的武器，可以在瞬間算出百萬個可能，而且永遠不會疲倦。從此電腦取得幾乎不敗的戰績，無論是黑白棋、西洋雙陸棋、拼字遊戲、國際象棋、中國象棋等棋局中，電腦都戰無不克的大勝人腦。

但是，在眾多競賽之中，電腦唯一無法贏過人腦的就是圍棋，因為西洋棋僅三十六步，但是圍棋卻高達三百六十一步，因此，圍棋較西洋棋複雜一千倍到一百萬倍之間，目前世界上最厲害的電腦透過計算排列組合，也頂多與業餘棋手相當。電腦要贏過人腦，不能夠只從加快計算所有的排列組合下手，而必須要讓電腦懂得學會思考棋路，讓電腦從每次棋賽中累積經驗而提升判斷能力，這就是機器學習的終極目標。

讓電腦學會思考看起來像是好萊塢電影中常見的橋段與幻想，不過，資訊學家卻不這麼認為。類神經網路的歷史可以最早追溯至 1943 年，心理學家華倫 ・ 麥克庫羅（Warren McCulloch），與邏輯數學家華特 ・ 匹茲（Walter Pits）最早提出了描述神經元運作的數學模式（MP 模式）。1949 年，Hebb 提出了著名的 Hebb 學習定律，認為如果兩個神經元同時被激發時，則它們之間的連接就會獲得加強。例如，知名的巴伐洛夫著名的狗與鈴聲實驗，**當聽到鈴聲與看到食物的神經元同時被激發，此時，兩者就會建立起增強的學習關係**，Hebb 定律為類神經網路的學習機制奠定了研究方向。

1957 年，Rosenblatt 提出的感知器（Perceptron）模型，是第一個將類神經網路付諸實踐的研究成果，雖然這掀起了類神經網路研究的第一波高潮，但是，很快的 1969 年由麻省理工學院的西摩 • 派波（Seymour Papert）與馬文 • 明斯基（Marvin Minsky）就為這波熱潮澆了一桶大冷水，他們認為透過這種感知器模型，甚至連簡單的「XOR」邏輯都做不到，因此，類神經網路研究陷入了長達十年左右的黑暗期。

1982 年時，約翰 • 霍普菲爾（John Hopfield）發明了霍普菲爾網路（HNN），以及 1986 年 David E. Rumelhart 等人發明了倒傳導網路（Back-propagation），解決了先前 Papert 對於類神經網路的質疑後，使得類神經網路研究再度熱絡起來。之後，類神經網路成為機械學習與機械控制的寵兒，被廣泛的應用在汽車控制（根據主人力道與駕駛習慣訓練調整機械）、家電控制（例如，洗衣機根據偵測的衣物重量及質料計算水量）、智慧型辨識（例如，指紋辨識、聲紋辨識與臉孔辨識）等領域。

6.1 類神經網路原理

6.1.1 ｜人類神經元在做些什麼

人類的大腦大約由 10^{11} 個神經細胞（Nerve Cells）組成，它的結構包括了幾個主要的單元：

■ 神經核（Soma）：神經元的中央處理部位。

■ 軸突（Axon）：神經元中負責把神經脈衝從細胞體往外傳遞的神經纖維。

■ 樹突（Dendrites）：神經元中負責把神經脈衝傳遞至細胞體的神經纖維。

■ 突觸（Synapse）：神經元之間的連結機制。

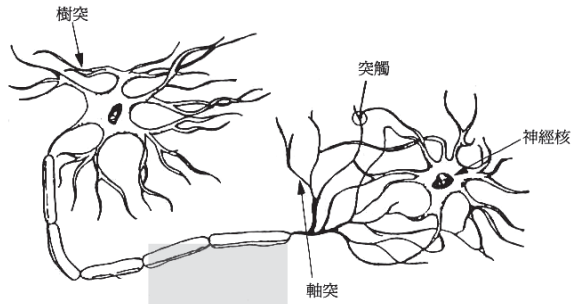

圖 6.1：人類神經元結構

樹突是接受外部輸入訊息的接受器，當**外部神經脈衝高於某個門檻值**時，神經元會被激發，而產生新的一個神經脈衝至軸突。突觸是神經與神經之間的連絡點，如果這個新的脈衝通過的是一個興奮突觸（Excitatory Synapse）時，會增加脈衝列的速率（Pulse Rate），若是通過抑制突觸（Inhibitory Synapse），則會減少脈衝列的速率。由此可以知道，影響神經元學習的主要關鍵，在於神經元如何轉換外部刺激成為新脈衝，以及突觸的強度。

6.1.2 ｜類神經元在做些什麼

類神經網路神經元的組成是仿效人類神經元的結構，其結構如圖 6.2，其中 X1、X2、X3…就是輸入變數值，而 W1、W2、W3 則是輸入變數的權重。X1 乘上 W1 就等於外部輸入的神經脈衝，但是，在通過樹突時，神經脈衝必須大於門檻值，才能夠傳遞至神經元。所以，對於神經元來說，所有的輸入訊號可以用下式來表示：

$$I_j = \sum_i w_{ij} O_i + \Theta_j$$

其中 I 表示輸入值，O 表示前端神經元的輸出值，w 表示權重，而 Θ 表示該神經原本身的常數項。

當脈衝通過樹突進入神經元後，神經元會透過加總函數把所有的神經脈衝累加，然後透過轉換函數（Activation Function）的方式，產生新的神經脈衝（Y1），向外傳遞。

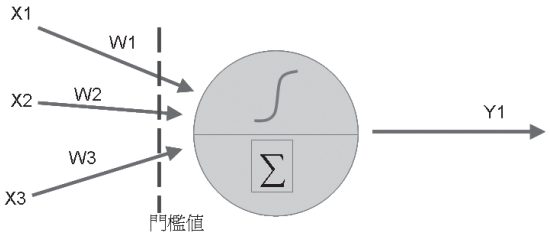

圖 6.2：神經元架構

轉換函數就是負責將神經元接收的輸入脈衝總和，轉換成輸出脈衝，但是，人類神經在處理外部刺激時，輸出訊號是有極限的，否則可能會因為輸出訊號過強而造成對神經元的傷害（舉例來說，疼痛並不會與外部刺激成正比，痛到某個程度後就沒有差別了）。因此，類神經網路在選取轉換函數時，不能夠使用傳統的線性函數，通常來說，都會選取兼具正向收斂與負向收斂的 S 型函數，目前 SQL Server 2008 所使用的轉換函數是 Sigmoid 函數（Logistic 分配，$Y=1/(1+e^{-x})$），當輸入變數趨近無限大時，則輸出值會趨近於 1，當輸入變數趨近於負無限大時，輸出變數會趨近於 0。因此，總輸入訊號強度會依照下式轉換為新的輸出訊號。

$$O_j = \frac{1}{1+e^{-I_j}}$$

6.1.3 ｜類神經網路架構與學習

將神經元彼此連結，就構成了類神經網路架構，也就是一個神經元的輸出可以變成下一個類神經網路的輸入脈衝。至於類神經網路結構的設計型態就成為演算法之間差異的關鍵。以目前 SQL Server 2008 來說，使用的是最普遍的倒傳導網路（Back-propagation Network）。

倒傳導網路是由多層的神經元結構所構成，其中最外層接收輸入變數的稱之為**輸入層**（Input Layer），而最後產生預測結果的神經元則稱之為**輸出層**（Output Layer），而介於中間的神經元稱之為**隱藏層**（Hidden Layer）。隱藏層的功能主要是增加類神經網路的複雜性，以能夠模擬複雜的非線性關係，就好像人的神經突觸連結越多，人就會越聰明。不過，很多人有錯誤的認知，以為類神經網路也是一樣，隱藏層越多，類神經網路就能夠越準確。事實上，隱藏層的確能增加模型

的複雜度，但是，過度複雜的模型反而容易造成模型過度學習（over-fitting）的問題，造成神經網路記憶了訓練組資料結構，而導致測試組預測力降低。整個神經元連結的網路結構，又稱之為「類神經網路的拓樸結構（Topology）」。

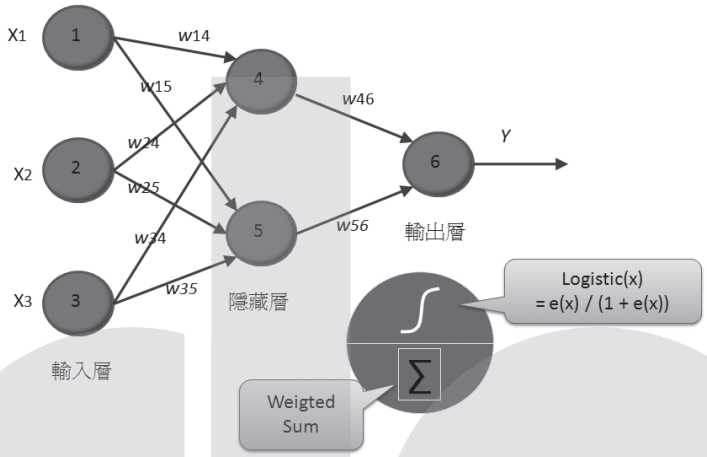

圖 6.3：類神經網路架構

微軟類神經網路神經元產生模式如下：

- 所有類別變數會先經過 MAXIMUM_STATES 參數控制與篩選，最後的選項數如果有 n 個，則最多產生 n+1 個神經元（一個保留作為遺漏值，訓練組資料有遺漏值才會產生）。
- 所有連續變數一律是最多產生兩個輸入神經元，一個為變數本體，一個是遺漏值。
- 輸出神經上限是 500 個，如果有超過則 Analysis Services 就會在採礦模型中產生新網路來代表其他輸出神經。

類神經網路可以應用在「連續變數」以及「類別變數」的預測，如果是連續變數預測，就是單一輸出層神經元的輸出訊號強度來決定連續變數值的大小。如果是類神經網路分類問題，例如，要使用類神經網路預測這筆交易是否為信用卡盜刷，則結果只有「是」或「否」兩種，此時，類神經網路輸出層就會具有兩個神經元，分別代表「是」、「否」（如果訓練組資料有遺漏值時，將會多一個「遺漏

值」神經元），當代表「是」的神經元輸出訊號強過代表「否」的神經元時，則預測結果為「是」。

倒傳導網路的學習過程，可以分成正向傳導與反向傳導兩個部分，所謂的正向傳導指的是輸入訊號從輸入層神經元進入到隱藏層，然後傳遞至輸出層，每個神經元只負責將訊號傳播至下游的神經元，以產生最終訊號（預測結果）。傳遞訊號的過程就是使用方才提到的轉換函數，以及神經元彼此連結的權重。倒傳導網路一開始權重是透過隨機的方式給予，因此一開始所產生的輸出結果會與實際結果產生極大的落差。

而反向傳導則是當輸出值（預測結果）與實際結果有落差時，則將可以計算出單筆案例的誤差訊號，如下式：

$$Error_j = O_j(1 - O_j) + (T_j - O_j)$$

其中 O 表示輸出訊號，而 T 表示實際的真值。

所謂的倒傳導，指的是要將此誤差項從輸出層反饋至隱藏層，此時，輸出層誤差項會根據神經連結，將誤差依照權重分配至隱藏層神經元。

$$Error_j = O_j(1 - O_j) \sum_k Error_k w_{jk})$$

因此，我們可以把類神經網路的學習過程當作是透過修正權重的方式，使得輸出值能夠接近實際值。當我們把一組訓練資料輸入類神經網路時，我們稱之為一個學習循環，此時，類神經網路會計算**整批樣本的輸出值與實際值的差異**，我們通常使用均方和誤差（Means Squared Error；MSE）來表示一個學習循環的誤差程度。此時，演算法可以根據誤差訊號的大小，同步修正各神經連結的權重，假設修正量與輸入訊號強度以及錯誤訊號成正比，此時，修正量以及修正後權重可以透過下式表示：

$$\triangle w_{ij} = I^* Error_j O_i$$

$$w_{ij} = w_{ij} + \triangle w_{ij}$$

其中第一式內的 l 表示學習速率（Learning Rate），通常是介於 0~1 之間，當數值越大，每次權數的修正量就會越大。因此，類神經網路的學習重心在於如何自動的、有效率的調整權重大小以及學習速率，目前最常使用的方式是最陡坡降法（The Gradient Steepest Descent Method），就是將誤差以可微分函數表示，透過微分斜率的梯度來產生權重的修正量，誤差為正向時（輸出值大於實際值），此時，降低神經權重，反之則增加神經權重，如此不斷反覆學習，讓誤差降低，這個過程就是我們所謂的「學習」。

我們以圖 6.3 的相位為例，來說明權重的計算模式：

Step01：首先，利用隨機的方式，產生各神經元間的權重，以及隱藏層與輸出層神經元的常數項。

W_{14}	W_{15}	W_{24}	W_{25}	W_{34}	W_{35}	W_{46}	W_{56}	Θ_4	Θ_5	Θ_6
0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1

Step02：根據輸入訊號，計算各隱藏層神經元的輸出訊號，假設輸入案例 (X1,X2,X3,Y)=(1,0,1,1)。

神經元 4：

總輸入訊號：$I_j=\sum_i w_{ij}O_i+\Theta_j=1*0.2+0*0.4+1*-0.5-0.4=-0.7$

轉換輸出訊號：$O_j=\frac{1}{1+e^{-I_j}}=\frac{1}{1+e^{0.7}}=0.332$

神經元 5：

總輸入訊號：$I_j=\sum_i w_{ij}O_i+\Theta_j=1*-0.3+0*0.1+1*0.2+0.2=0.1$

轉換輸出訊號：$O_j=\frac{1}{1+e^{-I_j}}=\frac{1}{1+e^{-0.1}}=0.525$

Step03：根據隱藏層輸出訊號，計算輸入層神經元的輸出訊號。

神經元 6：

總輸入訊號：$I_j=\sum_i w_{ij}O_i+\Theta_j=0.332*-0.3+0.525*-0.2+0.1=-0.105$

轉換輸出訊號：$O_j=\frac{1}{1+e^{-I_j}}=\frac{1}{1+e^{0.105}}=0.474$

Step04：此時，輸出值 0.474 與實際真值 1 不一致，我們可以計算神經元 6 的誤差項。

神經元 6 誤差項：

$Error_j=O_j(1-O_j)(T_j-O_j)=0.474(1-0.474)(1-0.474)=0.1311$

Step05：將此誤差項反饋至隱藏層，可以計算出隱藏層神經元誤差值。

神經元 4 誤差項：

$Error_j=O_j(1-O_j)\sum_k Error_k w_{jk})=0.322(1-0.322)(0.1311)(-0.3)=-0.0087$

神經元 5 誤差項：

$Error_j=O_j(1-O_j)\sum_k Error_k w_{jk})=0.525(1-0.525)(0.1311)(-0.2)=-0.0065$

Step06：最後根據神經元誤差項，更新各神經元的權重以及常數項，假設學習速率為 0.9。

$W_{14}=0.2+0.9(-0.0087)(1)=0.192$

$W_{15}=-0.3+0.9(-0.0065)(1)=-0.306$

$W_{24}=0.4+0.9(-0.0087)(0)=0.4$

$W_{25}=0.1+0.9(-0.0065)(0)=0.1$

$W_{34}=-0.5+0.9(-0.0087)(1)=-0.508$

$W_{35}=0.2+0.9(-0.0065)(1)=0.194$

$W_{46}=-0.3+0.9(0.1311)(0.332)=-0.261$

$W_{56}=-0.2+0.9(0.1311)(0.525)=-0.138$

$\theta_4=-0.4+0.9(-0.0087)=-0.408$

$\theta_5=0.2+0.9(-0.0065)=0.194$

$\theta_6=0.1+0.9(0.1311)=0.218$

如此，就達成一個學習循環的類神經網路權重修正，接下來持續此步驟，即可以使得輸出值越來越接近實際值，而達到建立模型的目的。

根據類神經網路理論，誤差要變成原來的 1/2 時，則需要 2^2=4 倍的時間，也就是說，當學習時間無限長時，誤差應該會無限趨近於零。但是事實上，如果當模型訓練越久，類神經網路越有可能會記憶訓練組資料裡的資料內容，**這樣雖然訓練組資料能夠誤差趨近於零，卻會造成新資料輸入時預測能力的降低**，這就是我們在《第 2 章: 資料採礦流程 CRIPS-DM》提到的過度學習問題，就好像填鴨式教學，給一個學生同一份考卷做無窮次，理論上，總有一天他會考一百分（只不過是背答案而已），卻不保證他以後每次考試都能一百分，甚至可能因為背答案習慣了而考低分。

為了避免過度學習的問題，因此，在類神經網路裡需要一組新的樣本來驗證權重修正的正確性，我們稱這組資料為鑑效組（Validation Set，在微軟的術語裡是 Holdout Set），**這組資料不會用來訓練類神經網路，只是用它來驗證神經權重修正的正確性**。在大多數的商用演算法中，會自動透過隨機抽樣的方式，從訓練資料中抽取出鑑效組。在訓練的過程中，訓練組資料理論上誤差會無限趨近於零，但是，鑑效組誤差一開始也會隨著降低，當過度學習的現象發生時，此時，鑑效組的誤差會從低點反轉開始增加，那麼鑑效組誤差最低點時的權重組合，就是最佳的訓練結果。目前 SQL Server 2008 中可以透過 HOLDOUT_PERCENTAGE 參數來指定鑑效組資料集之抽樣比率。

除了過度學習之外，使用最陡坡降法也會造成網路無法正確收斂至正確的結果，主要原因在於透過梯度的方向修正權重來達成誤差值的最小化。不過，由於轉換函數以及誤差函數本身是非線性函數，因此，如果一開始隨機權重是落在不正確的地方時，演算法會一直趨近至區域誤差最低的部位，我們稱之為**「區域最佳解」**，而陷在這裡一直出不來。但是，「區域最佳解」不一定等於**「絕對最佳解」**，因此，在進行類神經網路時，可以透過修改一開始產生權重的亂數模式，或是變動學習速率，來多次嘗試以避免區域最佳解的問題。

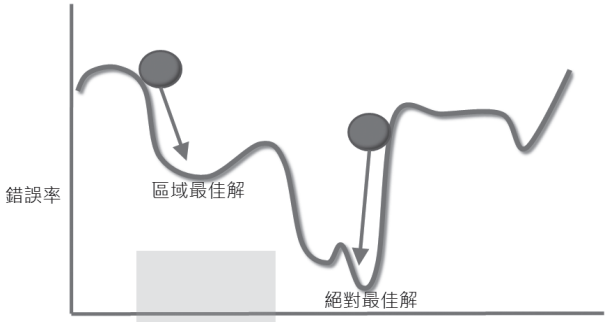

圖 6.4：區域最佳解

6.1.4 ｜建構類神經網路注意事項

類神經網路模型與群集演算法一樣，**其實都只能處理連續數值**，如果輸入變數為類別變數時，也會先把它換為虛擬變數（也就每個選項轉為 1 或 0 的編碼），因此，如果類別選項過多，會造成神經元數量過多，而導致類神經網路模型收斂困難。因此，在輸入類別變數之前要先注意類別選項數量，最好能做適度縮減，或是利用連續性指派的方法進行轉換。

至於極端值也可能造成誤差項過大，使得神經元收斂不易，建模之前也必須使用本書《第 3 章：資料預處理與變數篩選》所介紹的技巧先行調整。

此外，在大多數的資料採礦工具中，類神經網路是無法處理遺漏值的，當資料發生遺漏時，訓練的過程中會將這筆具遺漏資料的紀錄完全剔除。但是，在 SQL Server 2008 中特別強調處理遺漏值的能力，在處理變數時會多產生一個衍生神經元用來代表遺漏狀態。因此，即使輸入變數具有遺漏值，同樣可以用做為訓練資料。相同的，在使用模型預測時，就算輸入資料具有遺漏值，也同樣可以產生預測結果。

類神經網路是透過修正權重的模式來產生學習成果，**演算法本身並不具有變數篩選的功能**，也就是所有的變數都會計算出屬於自己的權重，因此，如果放入無效變數或是不穩定趨勢的變數時，很容易造成模型學習效果不佳，或是收斂過慢的狀況。因此，建議建立類神經網路之前要先篩選變數，或是可以先使用決策樹演算法建立模型，然後根據決策樹篩選出來的變數作為輸入類神經網路之用。

此外，在篩選變數時必須要注意事前檢定變數之間的相關性，否則很容易造成所謂共線性（Co-linearity）問題，也就是兩個輸入變數呈現高度正相關時，會使得函數得到無限多組解，導致類神經網路無法收斂，相關技巧在《第 3 章：資料預處理與變數篩選》中都有介紹。

所以，總合以上特性描述，可以知道類神經網路雖然有良好的預測能力，但是，在輸入的變數上的限制是比較多的，為了達到良好的預測效果，因此，資料預處理的步驟會在建模的過程中扮演關鍵的角色，在建模時請注意以下過程是否確實做到：

- 類別變數盡量轉換為連續變數。
- 類別變數盡量進行選項合併或是縮減。
- 連續變數需要修正極端值。
- 盡量避免遺漏值。
- 需要先進行變數篩選。
- 需要先處理變數間的共線性（相關係數）。

6.2 設計類神經網路模型

6.2.1 ｜建立類神經網路模型

類神經網路演算法同時可以解決「分類」以及「預測」兩種問題，也就是同時可以用來預測**類別變數**以及**連續變數**，不過，這並不會影響到它設定的流程，原則上，操作的步驟與決策樹演算法「完全相同」，僅需要將設定演算法畫面設定為類神經網路演算法即可。

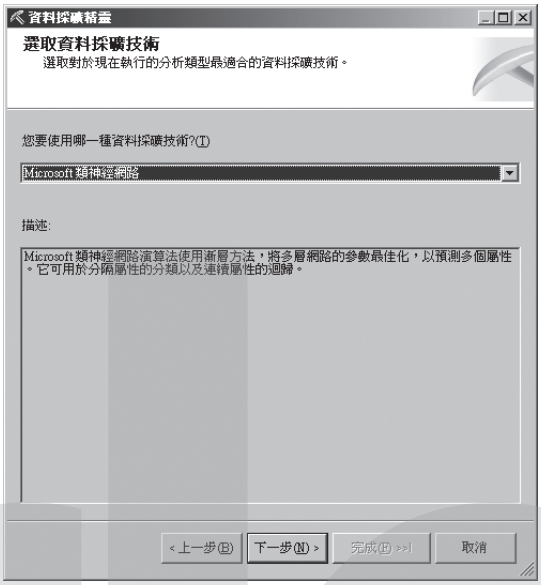

圖 6.5：設定演算法

6.2.2 ｜類神經網路分類模型

以下我們以電信公司交叉銷售為案例，建立類神經網路分類模型。範例資料庫的「BROADBAND」資料表是記錄一家電信公司，針對手機用戶推廣無線寬頻網卡的結果，在此希望利用類神經網路來找出有申裝潛力的客戶名單，各欄位的說明如下：

欄位名稱	資料型別	意義
CUST_ID	int	客戶代碼
GENDER	char(1)	性別
AGE	int	年齡
TENURE	int	申辦門號月份數
CHANNEL	char(1)	申辦通路
AUTOPAY	char(1)	自動轉帳扣繳
ARPB_3M	float	最近三個月平均電話費

欄位名稱	資料型別	意義
CALL_PARTY_CNT	float	撥出號碼數
DAY_MOU	float	白天通話分鐘數
AFTERNOON_MOU	float	下午通話分鐘數
NIGHT_MOU	float	夜間通話分鐘數
AVG_CALL_LENGTH	float	平均通話長度
BROADBAND	char(1)	回應無線寬頻註記

由於類神經網路模型效果會與輸入變數的處理密切相關，建議讀者先進行以下處理（範例光碟中各章範例的專案檔已經有處理好的結果），可以得到較佳的結果：

- 將白天通話分鐘數、下午通話分鐘數，以及夜間通話分鐘數相加產生「總通話分鐘數（minute of usage；MOU）」衍生變數。
- 絕對數量可以透過轉換為比例的方式變為相對變數，例如，可以將白天通話分鐘數、下午通話分鐘數，以及夜間通話分鐘數除以「總通話分鐘數（minute of usage；MOU）」以產生白天通話分鐘數比率…等比率變數。相同白天通話分鐘數的人可能會因為總通話分鐘數的不同而有不同的比率，所以，可以幫我們挖掘出不同的客戶面相。請注意，由於有可能有客戶是零使用量，所以要於具名計算利用 Case...when... 處理除以零的問題。

```
DAY_MOU _RATIO:
CASE
WHEN (DAY_MOU+AFTERNOON_MOU+NIGHT_MOU)=0
THEN 1
ELSE
DAY_MOU/(DAY_MOU+AFTERNOON_MOU+NIGHT_MOU)
END
```

■連續變數之間需要先利用相關係數的方式排除高度相關變數，例如，在此資料中下午通話分鐘數（AFTERNOON_MOU）與總通話分鐘數（MOU）高度相關（相關係數 =0.8786，各位可以根據《第 3 章：資料預處理與變數篩選》介紹的公式自行計算），在此只需要保留 MOU 即可，因為總通話分鐘數比下午通話分鐘數包含更多的訊息。

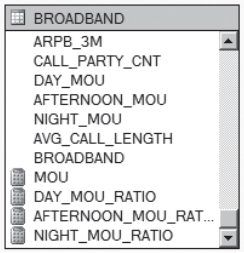

圖 6.6：產生具名計算

資料預處理完成後請根據以下步驟建立分類模型：

Step01：新增資料採礦結構，選取「從現有的關聯式資料庫或資料倉儲」來定義模型。

Step02：選用「類神經網路」演算法。

Step03：選取資料來源檢視，並勾選「BROADBAND」資料表為案例資料表。

Step04：選取輸入變數。

Step05：檢視資料型別與內容型別是否有誤。

Step06：設定測試集比率。

Step07：採礦結構命名

6.2.3 ｜類神經網路視覺化工具

類神經網路一直最讓人詬病之處，在於類神經網路是由複雜的權重與轉換函數所構成，因此，通常被視作是黑箱規則（就算是把那些權重與轉換函數列出來，恐怕也不是人類能看懂的）。但是，在資料採礦的過程中，除了獲得一個準確的模型之外，更重要的是希望能夠從規則內容中找出有意義的內容，因此，一個便利的視覺化工具是必要的。

不過，從類神經網路的原理來看，就算呈現出來也是一堆複雜的權重與函數，而且我們無法從權重的正負號，來判斷辨識之間是呈現正相關或負相關。因此，在 SQL Server 2008 類神經網路檢視器中，**並非以呈現神經元之間的權重，而主要是呈現資料內部的機率分布架構**。

在 SQL Server 2008 類神經網路檢視器中分成三個區塊，分別是「輸入」、「輸出」與「變數」，見圖 6.7。

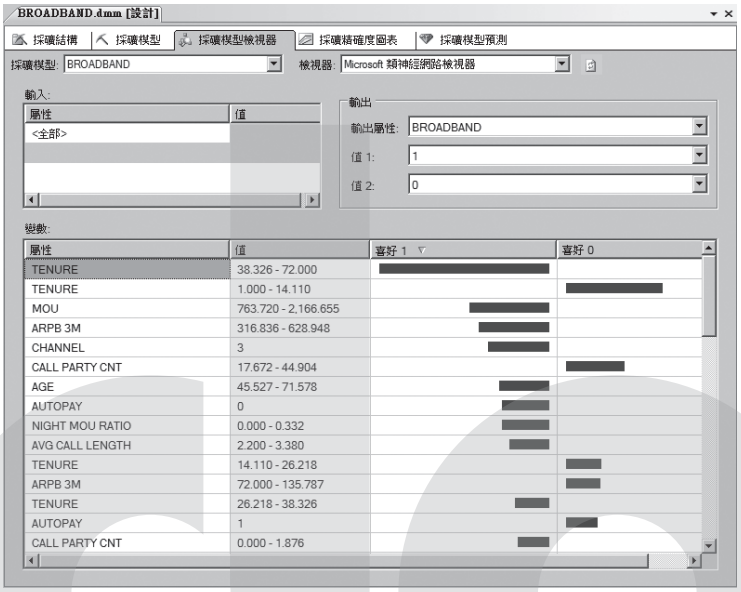

圖 6.7：類神經網路檢視器

輸入表示我們要輸入的「過濾條件」，類似 T-SQL 中的 WHERE 子句，可以在此輸入條件來限縮母體的範圍。分析者可以透過輸入區域的「屬性」選擇所要的變數，以及利用「值」來切換該變數的值範圍，類別變數會呈現所有選項，而連續變數則系統會自動切割成級距。如果我們選取的「屬性」為全部時，代表不針對輸入樣本資料做任何條件限制，即分析標的為母體（正確的說法應該是訓練組資料，在這裡的母體並不含測試組與鑑效組資料），見圖 6.8。

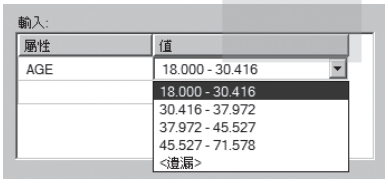

圖 6.8：類神經網路檢視器輸入區域

輸出區域中所列出的輸出屬性，則是指我們當初建立預測模型時的「可預測變數」，如果是類別變數，可以利用下拉式選單將此可預測變數內的值進行兩兩比對；如果是連續變數，則是將此連續變數離散化後進行比較。

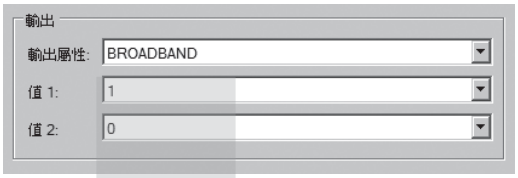

圖 6.9：類神經網路檢視器輸出區域

類神經網路檢視器的功能，就是把輸入區域所設定的屬性作為過濾條件，再將過濾出的子集合，檢視其中值 1 與值 2 間比例的差異。至於「變數」區域，則是列舉在此子集合內所有的變數選項組合，依照顯著性由高至低排列。

以圖 6.10 為例，當我們輸入條件為年齡介於 18.0~30.4 之間時，檢視器就會協助我們把符合這些條件的樣本篩選出來，然後計算這群樣本中的各個變數選項中可預測變數（是否會申辦無線寬頻）的選項分布機率。

例如，我們可以發現這群樣本中，如果申辦門號月份數（Tenure）狀態為 38.3~72.0 時，則輸出屬性值 1（BROADBAND=1，會申辦）的機率會高於輸出屬性值 2（BROADBAND=0，不會申辦），也因此，長條圖會呈現在「喜好值 1」的那一側，所以，我們就可以知道在這個輸入條件的樣本中，申辦門號月份數為 38.3~72.0 的人申辦機率是高的，反之，在這個輸入條件樣本中，如果申辦門號月份數為 1.0~14.1，則趨勢會逆轉，申辦機率會較低。

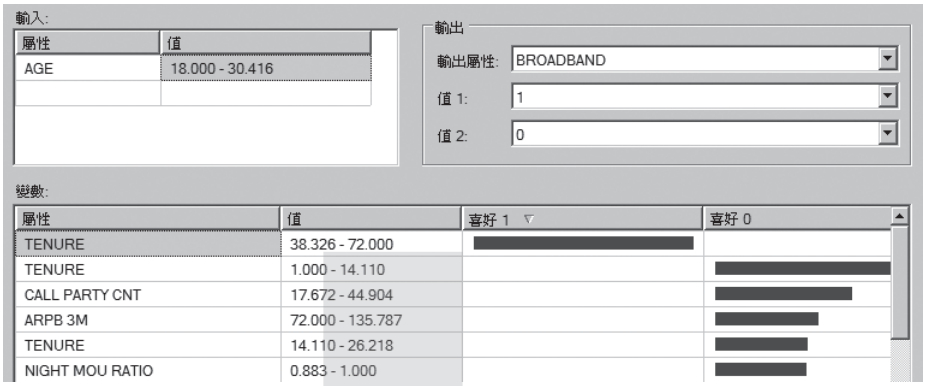

圖 6.10：設定類神經網路檢視器

不過，高低的結論必須根據比較來取得，因此，比較的基準不是絕對的比率，而是相對的增益。只要將滑鼠移至長條圖上時，會出現如圖 6.11 的分布視窗，其中表示在此樣本中，發生值 1（會申辦）的機率為 45.38%，而發生值 2（不會申辦）的機率為 53.48%。

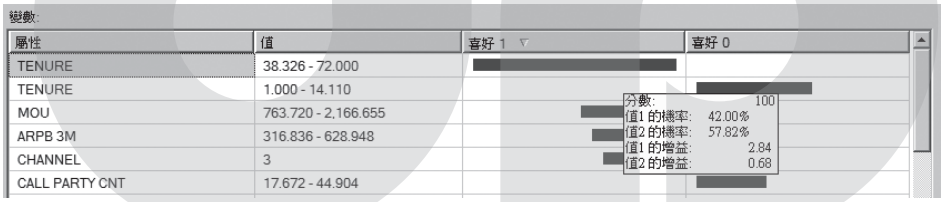

圖 6.11：變數視窗

乍看之下，這種人應該「不會申辦」機率較高，其實不然，因為申辦寬頻本身是稀有事件，因此，需要與「母體（排除測試組與鑑校組）」的值 1 與值 2 比率進行比較，由於「母體（排除測試組與鑑校組）」的值 1 分布為 14.79%、值 2 分布為 85.21%。分布視窗會產生「增益（Lift）」的指標，也就是將發生機率除以母體的原始發生機率，此時該條件下的增益為：

$$\text{Lift}_{\text{值1}} = \frac{42.00\%}{14.79\%} = 2.840$$

$$\text{Lift}_{\text{值2}} = \frac{57.82\%}{85.21\%} = 0.678$$

增益值大於 1 表示發生的機率高於母體，因此，該條件下發生值 1 的機率會高於母體 2.84 倍，該條件會被視作為傾向發生值 1。至於「分數」則是根據所有變數的增益的絕對值的大小，將最大增益正規化為 100（因此，圖 6.11 最顯著的變數分數為 100），最小增益值正規化為 0。

要注意的是，透過類神經網路檢視器只是協助我們了解變數之間交互作用的敘述性統計，而非權重高低，我們只能使用此檢視器瞭解變數重要性，但是，仍然無法一窺類神經網路結構全貌。不過，它對我們來說還是有積極的意義，例如，我們可將變數視窗拉到最下方，看到哪些是對於此樣本重要性最低的變數（如果根本沒出現在介面上的代表重要性更低的），如圖 6.12 所示：

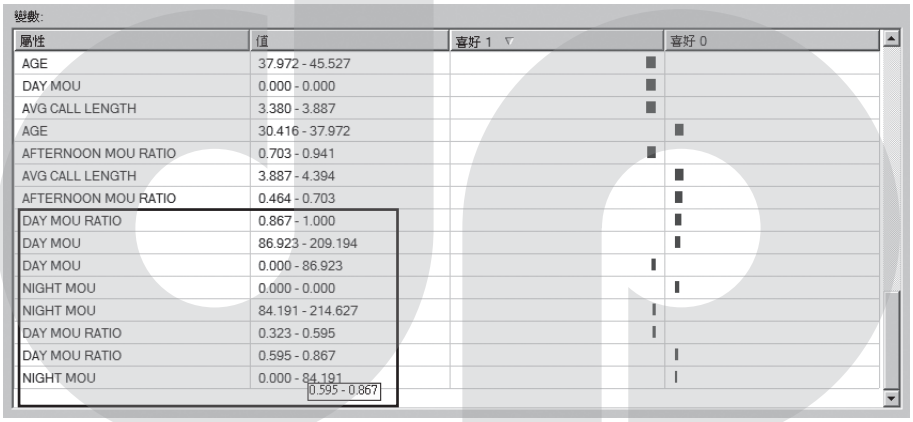

圖 6.12：重要性最低的變數

此時建立新模型時，就可以將這些低重要性的變數拿掉，見圖 6.13，也就是將這些低重要性變數的使用註記改為「忽略」，即可簡化模型神經網路架構，以達到更佳的預測效果（**有相同預測結果的模型是越簡單越好！**）

結構	BROADBAND	neural2
	Microsoft_Neural_Network	Microsoft_Neural_Network
AFTERNOON MOU RATIO	Input	Input
AGE	Input	Input
ARPB 3M	Input	Input
AUTOPAY	Input	Input
AVG CALL LENGTH	Input	Input
BROADBAND	PredictOnly	PredictOnly
CALL PARTY CNT	Input	Input
CHANNEL	Input	Input
CUST ID	Key	Key
DAY MOU RATIO	Input	忽略
DAY MOU	Input	忽略
GENDER	Input	Input
MOU	Input	Input
NIGHT MOU RATIO	Input	Input
NIGHT MOU	Input	忽略
TENURE	Input	Input

圖 6.13：拿掉低重要性變數

6.2.4 ｜類神經網路預測模型

類神經網路演算法除了類別變數分類之外，同時也可以進行連續變數的預測問題。以下我們將利用美國統計局所收集的波士頓地區房價資料集（來源自 Harrison, D. and Rubinfeld, D.L. `Hedonic prices and the demand for clean air', J. Environ. Economics & Management, vol.5, 81-102, 1978），利用類神經網路技術來建立波士頓地區房價預測的模型。此資料集位於範例資料庫的「HOUSE_PRICE」資料表，各欄位說明如下：

欄位名稱	資料型別	意義
HOUSE_ID	Int	房屋編號
CRIME_RATE	Float	所在城市犯罪率
BUSINESS_RATIO	Float	非零售業商業區占都市面積比
RIVER_SIDE	char(1)	是否靠河岸
NOX	Float	空氣氮氧化合汙染物濃度
ROOM_NUM	Int	房間數
HOUSE_AGE	Float	屋齡
WIIGHTED_DISTANCE	Float	到波士頓五大辦公中心加權平均距離

欄位名稱	資料型別	意義
HIGHWAY_INDEX	Float	到高速公路遠近指標
TAX_RATE	Float	每萬元地價稅
TEACHER_RATE	Float	該城市師生比（每多少學生分配一個老師）
BLACK_INDEX	Float	居民黑人比率指標（1000(Bk - 0.63)^2）
LOW_STATUS_RATIO	Float	低階層人口比率
HOUSE_PRICE	Float	平均房屋單價

建立模型的過程與分類模型完全相同，不過，請注意的是，筆者並不太建議各位建立類神經網路預測模型時使用「建議」變數的功能，因為微軟的建議變數功能其實比較適合類別變數，對於類神經網路擅長的連續變數的推薦能力並沒有太好（因為是先將連續變數離散化為類別變數再檢定）。所以，筆者比較建議可以先全放入變數（但需要先排除高相關係數變數），然後根據類神經網路檢視器排除低重要性變數。

圖 6.14：建議變數

如圖 6.14 中，低階層人口比率（LOW_STATUS_RATIO）並未在推薦名單中，但是在類神經網路檢視器中，見圖 6.15，比較房價最高（值 2）與房價最低（值 1）的兩群，最顯著的差異就是低階層人口比率（LOW_STATUS_RATIO）變數，低階層人口比率高則偏向低房價，反之則高房價；次要重要變數包括到波士頓五大辦公中心加權平均距離（WIIGHTED_DISTANCE），以及該城市師生比（TEACHER_RATE）等，這些變數在推薦變數中排名都並不高，所以，筆者還是建議各位回歸類神經網路檢視器，以得到較合理的變數組合。

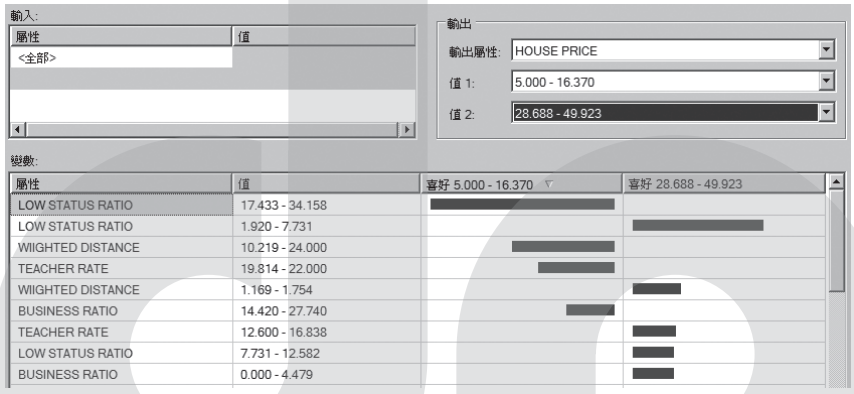

圖 6.15：房價模型類神經網路檢視器

6.3 類神經網路模型調整

6.3.1 ｜類神經網路演算法參數

SQL Server 2008 類神經網路演算法提供以下調整參數：

參數名稱	參數說明
HIDDEN_NODE_RATIO	此參數可用來指定隱藏層神經元數量。演算法計算隱藏層內的節點數目的公式為：HIDDEN_NODE_RATIO * SQRT({ 輸入層神經元的數目 } * { 輸出層神經元的數目 })。預設值為 4。
HOLDOUT_PERCENTAGE	使用此參數可以用來指定要抽取多少百分比的樣本來作為鑑效組，預設值為 30。

參數名稱	參數說明
HOLDOUT_SEED	透過此參數可以用來產生隨機抽樣抽取鑑效組的亂數種子，修改此參數會造成鑑效組成員變動。如果沒有指定此參數，演算法會依據模型名稱產生亂數種子，以確保重新處理模型時保持相同的鑑效組資料成員。預設值為 0。
MAXIMUM_INPUT_ ATTRIBUTES	最大輸入變數數量，如果輸入變數超過此值，則演算法會自動啟動變數篩選功能，如果將此參數設為 0，則關閉限制門檻。預設值為 255。
MAXIMUM_OUTPUT_ ATTRIBUTES	最大輸出（預測）變數數量，如果輸入變數超過此值，則演算法會自動啟動變數篩選功能，如果將此參數設為 0，則關閉限制門檻。預設值為 255。
MAXIMUM_STATES	由於類別變數選項過多時，會造成神經元數量膨脹，因此，使用本參數可以設定選項數的上限，如果類別變數的選項數目大於此參數限制，演算法會使用出現頻率最高的選項，並將其餘的選項視為遺漏。預設值為 100。
SAMPLE_SIZE	演算法會根據 SAMPLE_SIZE 或是總案例數 * (1 - HOLDOUT_ PERCENTAGE/100) 兩者中取其低者來作為訓練組資料案例數。預設值為 10000。

6.3.2 ｜類神經網路最佳拓樸結構

在類神經網路拓樸結構中，有一類拓樸結構稱之為金字塔結構（Pyramids），是在諸多文獻中記錄具備較佳的收斂效果與預測能力。所謂的金字塔結構就是指**輸入層神經元多於隱藏層神經元；隱藏層神經元多於輸出神經元的網路架構**。

在微軟類神經網路中是利用 HIDDEN_NODE_RATIO 參數來調整隱藏層神經元數量（目前僅支援一層隱藏層），隱藏層內的節點數目的公式為：

HIDDEN_NODE_RATIO * SQRT({輸入層神經元的數目} * {輸出層神經元的數目})

也就是說，是將 HIDDEN_NODE_RATIO 參數乘以輸入層神經元與輸出層神經元數量的幾何平均數，透過這樣的方式比較容易達到金字塔結構。

以剛才的房價預測模型為例，輸出變數是房價（沒有遺漏值），因此，輸出層神經元數量為 1。至於輸入變數除了 RIVER_SIDE 是類別變數之外，其他 11 個變數都是連續變數（只有 TEACHER_RATE 有遺漏值），輸入層神經元數量為：

■ RIVER_SIDE 共兩個（一個為 1，一個為 0）。

■ TEACHER_RATE 共兩個（一個為變數本體，一個為遺漏值）。

■ 其他 10 個無遺漏值的連續變數都各有 1 個。

所以，輸出層神經元共有 14 個，因此，幾何平均數為 SQRT(1*14)=3.74，預設 HIDDEN_NODE_RATIO 參數值為 4。而隱藏層神經元數量為 3.74*4=14.96（無條件捨去為 14 個），如果使用預設值該神經網路並非是金字塔結構，若將 HIDDEN_NODE_RATIO 參數值改為 2，那麼隱藏層神經元就會變成 3.74*3=11.22（無條件捨去為 11 個），比較接近金字塔結構，此時，模型收斂的狀況以及預測效果會比較理想。請注意，如果隱藏層神經元數過少也可能會因為神經網路過於簡單，而造成模型成效不佳。