國立政治大學應用數學系

碩士學位論文

Dynamic Implement Radial Basis Function Networks

動態輻狀基底函數類神經網路建構之研究

碩士班學生：林祐宇撰

指導教授：蔡炎龍 博士

中華民國九十九年六月

誌謝

在論文撰寫期間，非常感謝指導教授蔡炎龍老師對我的耐心教導以及生活上的點點滴滴，老師除了對我在數學專業領域上的細心指導外，也帶領我在程式語言上的精進，從 LaTeX、HTML、CSS 到 Python，老師都不厭其煩的教導我，並引導我到趣遊網實習，使我獲益匪淺，收益良多。此外，我也要特地感謝陳天進老師，老師對數學的教學態度深深的影響了我，也很感謝老師給我機會讓我擔任高等微積分助教、參與大考中心批改考卷的行政助理經驗，平常生活的照顧等。

在政治大學應用數學系生活了三年，過的很開心，這裡有很多老師平常對我們都很照顧，像朋友般的相處，讓我真的有感覺是個大家庭；感謝張天財學長平常在課業上的指導，以及互相勉勵打氣的同學與學弟妹，很謝謝你們，有你們的互相扶持讓我的研究所生活更完整。另外，感謝谷俊宇學長及白志棠學長，在趣遊網實習期間耐心的教導我，給了我很多的鼓勵與專業領域上的建議。

最後感謝在口試期間的陳天進老師、蔡炎龍老師、胡馨云老師，您們的指導與建議使我的論文更加完善。

中文摘要

近年來輻狀基底函數類神經網路 (Radial Basis Function Networks , RBFN) 應用在時間序列相關問題上已有相當的成果 [13]。在這篇論文裡，我們嘗試建構一個電腦軟體工具，可以很容易造出 RBFN，應用在時間序列預測相關問題上。更進一步的說，我們的電腦工具可以輕易做出即時修正，完全符合使用者的需求。我們一開始先複習 RBFN 的基本架構， 並說明如何應用到時間序列的問題上。接著我們研究近年來相 當受到重視的 T-RBF (Temporal RBF) 架構。最後，我們解釋如何使用 Adobe Flex 去建構我們所需要的電腦軟體工具。這個工具是跨平台的程式，並且不論是雲端計算或是單機應用皆很合適。

關鍵字：輻狀基底函數、暫時性輻狀基底函數、類神經網路、時間序列

Abstract

During recent years, applying Radial Basis Function Networks (RBFN) to time series problems yields many important results. In this thesis, we try to implement a cross-platform computer tool that can easily construct a RBFN applied to time series forecasting problems. Moreover, the RBFN created by this computer tool can do real-time modification to fit specific needs. We first review the basic structures of RBFN and explain how it can be applied to time series problems. Then, we survey on so called temporal radial basis function (T-RBF) model, which draws much attention these years. Finally, we explain how we use Adobe Flex to create a computer tool as we mentioned in the beginning. The computer application is cross-platform and is suitable for both cloud computing and desktop applications.

Key word: RBF, temporal RBF, ANN, time series

目錄

第一章　緒論

RBFN 之概念最早是由 Hardy(1971)所提出,後來 Powell [10] 建立其網路架構，Moody 與 Darken(1989)則利用 RBFN 解決數學函數對應之問題，Broomhead 與 Lowe(1988)將其應用於數值逼近或內插之問題上，而後相繼有各種類似之網路結構之 RBFN 相繼被提出。隨著輻狀基底函數網路之快速發展，已有不同之訓練法則用以架構更有效率之輻狀基底函數網路，並利用於具有時間序列相關性之問題上。Mikko(1996)等人曾以輻狀基底函數網路爲基礎，結合自我迴歸(autogressive)之觀念架構網路，並將之應用於時間序列模式之預測，證實 RBFN 之準確性遠勝過 Autogressive model(AR)、Threshold Autogressive model(TAR)及倒傳遞多層感知機(Backpropagating Multiplayer Perceptron)。此外，Park 與 Sandberg(1993, 1994)證明 RBFN 有能力模擬近似任何函數，近年來更是受到廣泛的研究，如影像傳輸方法 [19]、洪水預測 [16]、河口水位預測 [17]、建築風載 [18]等。

在本篇論文裡，我們嘗試以 Adobe Flex 建構一個可以輕易造出 RBFN 的動態網頁應用程式，並且希望是跨平台的，提供給相關需求的使用者可即時改變 RBFN 架構，藉此學習 RBFN 及應用在時間序列相關問題上，以因應各項預測之需求。

最後介紹本論文之架構，在此論文的第二章，我們從時間序列的涵義開始，簡單的介紹時間序列的概念、預測的方法及類神經網路的基本概念。本論文的重點著重在前饋式類神經網路架構，討論如何以該架構結合時間的概念去處理預測的問題，所以我們以時間稽延類神經網路(TDNN)爲例，將「過去」與「現在」的資料結合類神經網路，讓大家了解具有基本時間性質的類神經網路架構，並討論其優缺點。

第三章我們開始建構輻狀基底函數類神經網路(RBFN)，它的結構主要分爲三層，(1)輸入層：將我們的學習資料(learning data)轉成輸入值對應到隱藏層，(2)隱藏層：基本概念就是由許多輻狀基底函數(radial basis function) [1]所成的集合，將輸入層的值透過非線性活化函數(activation function)轉換至輸出層；幅狀基底函數

$h : \mathbb{R} \geq 0 \rightarrow \mathbb{R}$，的形式有很多，常見的有高斯函數(Gaussian function)、二次多變數函數(multiquadratic function)等，隱藏層的大小決定於中心點的個數，也就是整個網路的大小，(3)輸出層：將隱藏層經過輻狀基底函數轉換過的值作線性組合。

中心點對 RBFN 的影響是很大的，我們也將介紹兩種中心點選取方式，隨機選取法與垂直最小平方法(OLS)，討論其選取中心點的過程並比較其優缺點。

我們希望以 RBFN 作預測時，由於 RBFN 屬於靜態類神經網路，在處理時間變動性的問題上有明顯的缺點。Day 與 Davenport [3]在模糊信號預測上根據 Mackey-Glass 差分方程式提出倒傳遞神經網路於時間應用上的方法。Lin、Ligomenides 與 Dayhoff [7]提出自適應稽延類神經網路(ATDNN) [12]，其爲一在學習過程中可改變稽延時間(time delay)的方法。在第四章裡我們將介紹 T-RBF 方法，它提昇了時間對 RBFN 的影響，它是一種動態的學習方法，可以讓我們解決 RBFN 在預測上的限制。

最後一章我們將建構動態的學習系統，以 Adobe Flex 技術開發 RBFN 網頁應用程式，讓使用者可以在網頁上面就能動態的學習 RBFN，並利用該應用程式進行預測。在這章節裡我們將以 ActionScript 3.0 建立代數工具以及 N 相關的數學工具，在應用程式上面我們將以較好的視覺效果，配合動態的資料變動與簡單的操作環境，期望讓使用者能更簡單明瞭的去學習 RBFN 網路架構，並希望藉由網頁應用程式的方式將 RBFN 推廣給更多人認識。

第二章　理論基礎

2.1　時間序列模型的目的與涵義

有人可能會問，爲什麼要推算「未來」的值呢？這其實是一個很重要的問題，這同時也是研究時間序列模型的主要目的—預測變數未來的可能變化。例如每一個股市的投資人當然都會想「預測」未來某一個股到底會漲或跌到哪裡，以利決定下一步的投資策略；政府也經常「預測」未來的的經濟成長率爲何，以考慮是否採取某些經濟政策；企業也會想「預測」未來市場對企業本身產品的需求量是多少，來決定要不要調整生產線；養雞的農民會想「預測」未來雞 的價格如何，以便決定應增產還是減產。 簡而言之，研究時間序列模型的主要目的，就是希望發掘時間序列變數現在和過去的關係，以預測此變數未來的趨勢值爲何，進而能事先做決策。

2.1.1　預測的方法

預測方法基本上可以分爲兩種基本型態：定性方法(qualitative methods)與定量方法(quantitative methods)。前者通常以專家的意見爲主，依據過去的經驗，或特殊感官功能對未來的事件做本質、特性的預測；後者則以歷史事件，化成時間序列資料趨勢圖，並判別出他的特徵，以數理方法模式化後再做量的預測。以下是一些較常使用的時間序列分析方法：

(1) 成長模式(growth)：

如古典的指數成長曲線(exponential growth curve)、邏輯曲線(logistic curve)或公配茲曲線(Gompertz curve)、多項式函數確定性(determinstic)模式。根據成長率(growth rates)定性，或由所建構之微分方程式解出一組合式參數。

(2) 指數平滑法(exponential smoothing):

將過去的資料，以特定的期數求移動平均值作爲預測值。通常其權數呈指數遞增，而較接近的資料，常給予較大的權數。

(3) 多變量迴歸方程式(multiple regression):

相關變數及其因果關係之間的探討，通常以迴歸方程式表示。由最大概似法或最小平方誤差法，估計各變數的係數。

(4) 分解法(decomposition):

對時間序列的四個特性：趨勢(T)、季節(S)、循環(C)及隨機(I)加以分析估計。

(5) 自迴歸整合移動平均模式(ARIMA):

由 Box-Jenkisn 所提出的自迴歸整合移動平均模式(AutoRegressive Integrated Moving Average , ARIMA)，主要方法爲對歷史資料分析，檢視其自相關與偏自相關等特性。

(6) 狀態空間(state space):

考慮系統中影響現在狀態的因素所構成之最小集合。而此系統未來的行爲，亦僅受到此集合現在與前一期系統輸入變因影響。

(7) 計量經濟(econometrics):

以經濟分析觀點建立計量模式。此方法需要專業經濟知識與背景。而其在蒐集資料與建構合理聯立方程式的過程中需投入較高成本及費時。

(8) 非線性模式(nonlinear model):

對不滿足線性 ARIMA 模式的一些基本假設，應用特定非線性模式來處理，可改善線性模式在配適與預測結果不佳的情況。較常用的非線性模式有：雙線性模式(bilinear modles)、門檻模式(threshold modles)、與指數 ARMA 模式等。

(9) 神經網路(neural networks):

模擬人腦神經組織，經嘗試錯誤(try and error)與修正記憶後所得之自由分模式(modle free)，來做預測。故亦可說是自由模式法或經驗學習模式法。

2.1.2 時間序列概念

時間序列模型的基本概念，可以從自回歸模型(Autoregressive model , AR Modle)開始建立。所謂的 AR 模型的意義，簡單來說就是現在的某一變數值和同一變數過去的變數值有關。我們以下用一個簡單的數學模型來說明會更容易懂。

先定義某一個變數 y 在時間點 t 時的數值用符號 y_t 來表示之，那麼該變數的現在值和過去值有關，可以用函數來表示:

$$y_t = f(y_{t-1}) \tag{2.1.1}$$

而 AR 模型就是將式 (2.1.1) 用線性函數來表示。AR 模型被廣泛的應用在很多的預測應用上 [11]，AR 模型可以 (2.1.2) 形式表示:

$$y(t) = \sum_{i=1}^{n} a_i \, y(t - \tau_i) \tag{2.1.2}$$

上式中的 $y(t - \tau_i)$ 是過去輸出值，n 爲模型的階數，τ_i 爲時間稽延。在 AR 模型建構上也有一些缺點如下:

(1) AR 模型是一個線性函數模型，在許多情況下無法追蹤有劇烈變化的預測值。

(2) 當我們要處理高階的 AR 模型時，不見得會有期望的好結果。

2.2 類神經網路

2.2.1 人工類神經網路簡介

McCulloch 和 Pitte 在 1943 年提出一個基本計算神經元(computing neuron)的模型是一種基本數學運算單元，爲往後類神經網路系統奠定了基礎。1948 年 Donal Hebb 發表了關於神經元間權重值的修改學習規則稱爲 Hebbian 學習法則，其中他認爲訊息是存在於神經元間的神經鍵上而此種學習法則成爲所有類神經網路學習規則的先趨。

類神經網路是由人類所發明出來的，主要目的是想要模仿生物神經元的行爲模式。生物神經元模型如圖 2.2.1 所示，內有細胞核(Soma)、軸突(Axon)、樹突(Dendrites)及突觸(Synapses)，當外界各種不同的刺激資訊(如：光、電、熱)轉成電流信號，經由突觸內部的電位變化，透過樹突傳遞到細胞本體，刺激細胞本體內的細胞

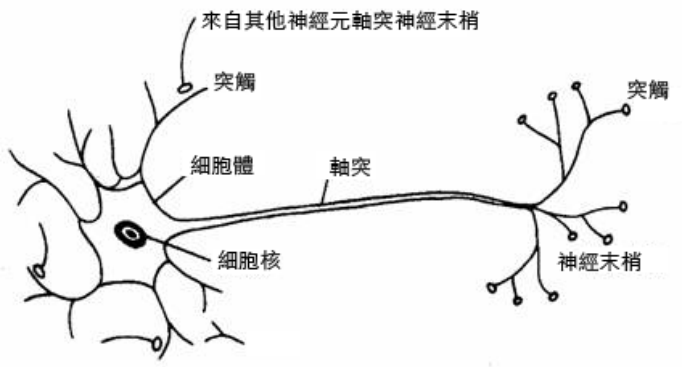


圖 2.1: 生物神經元模型

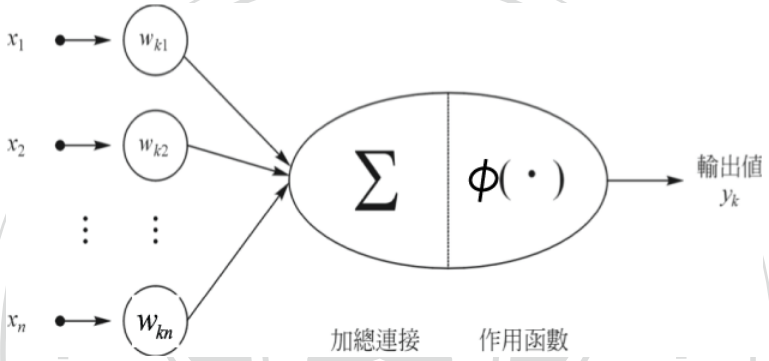


圖 2.2: 類神經元模型

核，細胞本體會自動發出電流脈波藉由軸突再傳至相連結之下一層神經元，成爲下一層神經元的輸入信號。

圖 2.2.1 爲一個人工神經元模型，表現類神經網路的基本設計概念，該圖清楚顯示一個人工神經元的輸入向量 X ，權重組 W 、活化函數 ϕ 與輸出值 Y 的基本關係架構，一個神經元 j 可以用下列二方程式來描述:

$$net_j = \sum_{i=1}^{m} w_{ji}\, x_i + b_j \tag{2.2.1}$$

$$y_j = \phi(net_j) \tag{2.2.2}$$

其中 w_{ji} 爲連結第 i 個輸入值與第 j 個人工神經元之加權值，b_j 爲偏權值，若爲正對輸入是增益，若爲負則抑制輸入值。

下面介紹幾種常見的活化函數:

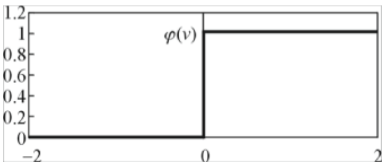


圖 2.3: 門檻值函數

(1) 門檻值函數

$$\phi(v) = \begin{cases} 1, & \text{if } v \geq 0 \\ 0, & \text{if } v < 0 \end{cases}$$

(2) 片段線性函數

$$\phi(v) = \begin{cases} 1, & \text{if } v \geq \frac{1}{2} \\ v + \frac{1}{2}, & \text{if } v > -\frac{1}{2} \\ 0, & \text{if } v \leq -\frac{1}{2} \end{cases}$$

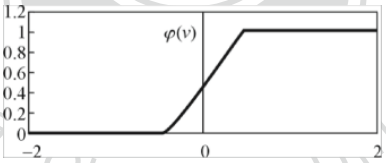


圖 2.4: 片段線性函數

(3) S形函數

$$\phi(v) = \frac{1}{1 + e^{-\alpha v}}$$

2.2.2 類神經網路分類

類神經網路可由其網路架構及學習方式來加以分類。

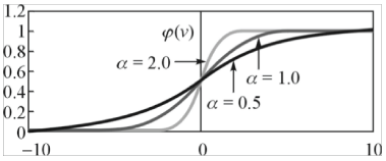


圖 2.5: S形函數

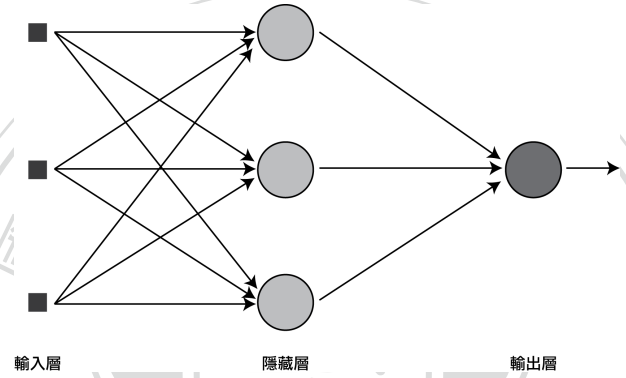


圖 2.6: 多層前饋式網路

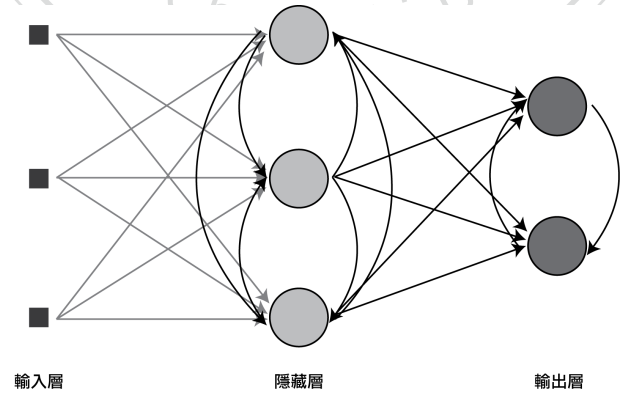


圖 2.7: 回饋式類神經網路架構

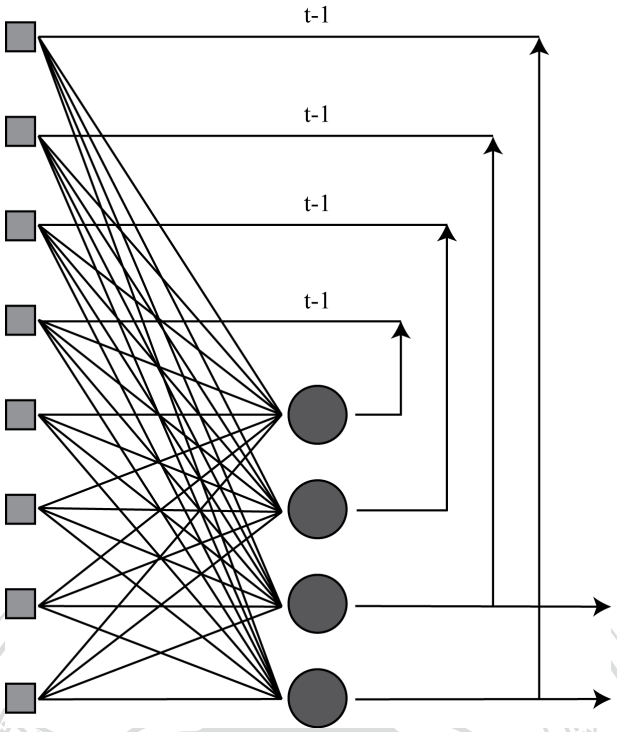


圖 2.8: 回傳的時間稽延示意圖

- 依網路架構可分爲二種:

(1) 前饋式類神經網路(Freedfroward Networks)

前饋式網路架構如圖 2.6 所示，網路中的神經元是以多層方式(multi-layer)排列，主要分爲輸入層(input layer)、隱藏層(hidden layer)及輸出層(output layer)，同一層神經元彼此不相連接，每一層神經元只接受前一層神經元的輸出值爲輸入。

(2) 回饋式類神經網路(Feedback Networks)

回饋式類神經網路與前饋式類神經網路的最大不同，在於回饋式類神經網路至少會含有一回饋迴圈，如圖 2.7 所示，一個回饋式類神經網路可能僅包含一層神經元，而在此層的神經元會各自將其輸出之訊號回傳給同一層中的其他神經元或前一層的神經元，作爲輸入資料。回饋式的架構常用於處理動態現象或動態的時間序列，這是因爲回饋式類神經網路在將其資料經由回饋回傳給前一層或同層中之其他神經元時，會自動產生一時間上的延遲，如圖 2.8 所示。而回饋式類神經網路或稱爲遞迴式網路，主要是由於網路的回饋

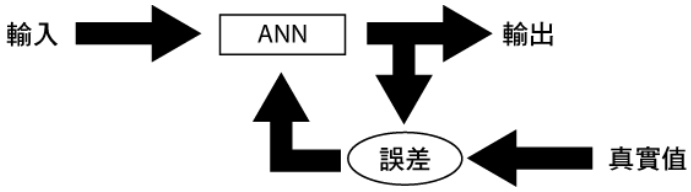


圖 2.9: 監督式學習演算法

圖 2.10: 非監督式學習演算法

項傳到前一層產生了遞迴的演算，遞迴式網路藉由遞迴項更加強網路的學習能力與表現，尤其在非線性的動態系統中，遞迴式網路通常比其他類型網路有更好的學習能力與表現。

- 依學習方式分類則可分爲下列二種:

(1) 監督式學習(Supervised learning)

如圖 2.9 此類網路的訓練樣式(pattern)有輸入、輸出值，在訓練過程中藉由輸入輸出值間的對映規則，產生一組可應用於新樣式的權重值；藉由過程一次又一次地調整網路連結的強弱，來降低網路輸出值與目標輸出值(desired output)之間的差距，直到差距小於一定的「臨界值」才告停止。如倒傳遞網路(Back-Propagation Network, BPN)、學習向量量化網路(Learning Vector Qantization, LVQ)與反傳遞網路(Counter-Propagation Network, CPN)等均爲此類網路學習方式。

(2) 非監督式學習(Unspuervied learning)

如圖 2.10 非監督式學習演算法的特色在於訓練過程只需要提供輸入資料，而不提供輸出資料，網路依輸入資料(向量)去學習及調整權重，如下一節將提到的垂直最小平方法 (Orthogonal Least-Squares, OLS)。

2.2.3 如何以神經網路作預測

神經網路爲目前發展極爲迅速之一門科學。且已知神經網路能就一確定模式(Deterministic models)作相當良好之函數近似(Functional approximation)。給予一組時間序列資料 $\{Y_0, Y_1, Y_2, ...Y_t\}$ ，其一般式可寫成:

$$Y_{t+1} = F(Y_{t-k}, Y_{t-k+1}, \ldots, Y_t) + \varepsilon_{t+1} \tag{2.2.3}$$

其中 ε_{t+1} 爲隨機亂數項，爲不可控制之未知項。而我們以神經網路來做預測時，觀點與上述之傳統方式有所不同。我們利用神經網路具備對任一函數均可做良好之近似的特性，建構一個函數 f，使 f 具有下述功能：在時間序列中之某一筆資料 Y_t 若可由前 m 筆資料經由函數 f 求得，即

$$Y_{t+1} = f(Y_{t-m}, Y_{t-m+1}, \ldots, Y_t) \tag{2.2.4}$$

在 (2.2.4) 式中不再具有 ε_{t+1} 項，此隨機項被視爲是神經網路函數 f 轉換之一部分。神經網路之學習過程即是要建立此函數 f。

以預測準確與否之觀點來看，在我們作預測前，必須事先建立一套規則讓神經網路學習。例如在 (2.2.3) 式中之規則爲：第 t 筆資料之求得必須依賴其前 m 筆之資料。一旦確定規則後，網路即可一此開始學習。

2.2.4 時間稽延類神經網路

時間稽延類神經網路(Time-Delay Neural Networks , TDNN)屬多層前饋式類神經網路架構，通常利用誤差倒傳遞演算法(Error Back Propagation , EBP)，來學習輸入及輸出的靜態映射(mapping)關係，所謂的靜態，指的是輸入與輸出的對應關係並不會隨著時間改變，屬穩定程序(stationary processes)；這樣網路架構及演算法較適用於輸入及輸出具有空間(spatial)特性或與時間無關的形態辨識(pattern recognition)問題。

然而，在我們面對的許多問題中，「時間」常是一個重要的因素，當類神經網路在處理這類的問題時，必須能夠加入「過去」的資訊，因爲光靠「現在」的訊息當做輸入項會忽略掉許多重要的相關因子，即時間前後資訊間的相關特性。位處裡這樣的問題，一般最常見的作法是使用多層前饋式類神經網路，將時間序列轉換成空間序列的形態輸

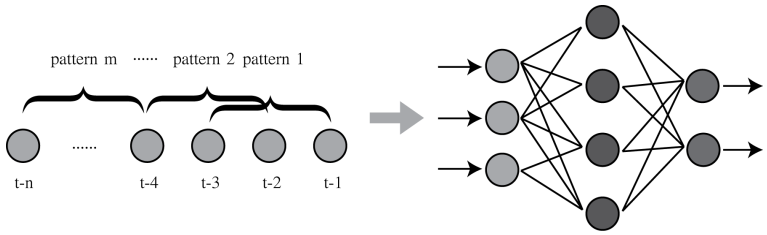


圖 2.11: 時間序列轉成空間序列的型式當成輸入項

入網路中，也就是將橫向不同時間的資料截取片段改成縱向的輸入。如圖 2.11 所示，每次輸入網路的 pattern 包含現在、前一期、前兩期的資料，這樣的輸入方式，即表示同時將與現在網路輸出有關的前幾期資料一起輸入網路。

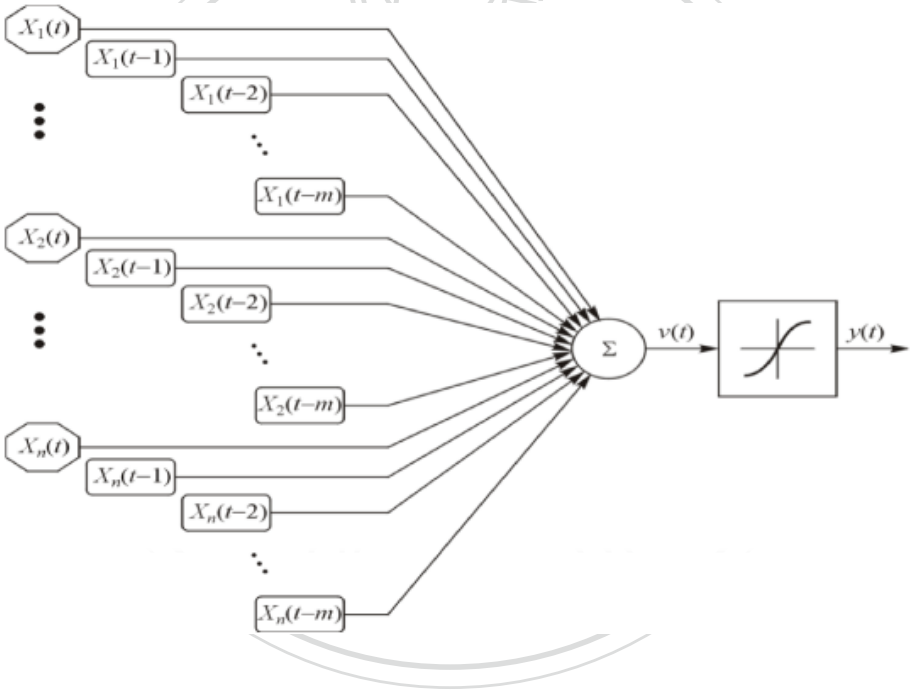


圖 2.12: TDNN 中，單一神經元接收 n 種輸入訊息，m 階時間稽延的示意圖

時間稽延類神經網路即一上述方法將輸入訊息各自稽延數個時期，並同時當做前饋示類神經網路的輸入向量，圖 2.12 顯示一個單一神經元同時接收(連結) n 種輸入訊息，且每種輸入訊息皆包含目前及 m 個時間稽延的資料，故輸入向量大小爲 $(m+1) \times n$。

典型的 TDNN 結構示輸入層將稽延的訊息合併輸入，並利用標準的倒傳遞演算法調整連結權重向量，因此訓練後的網路所建立的是一組前後相互關連的輸入型態(input

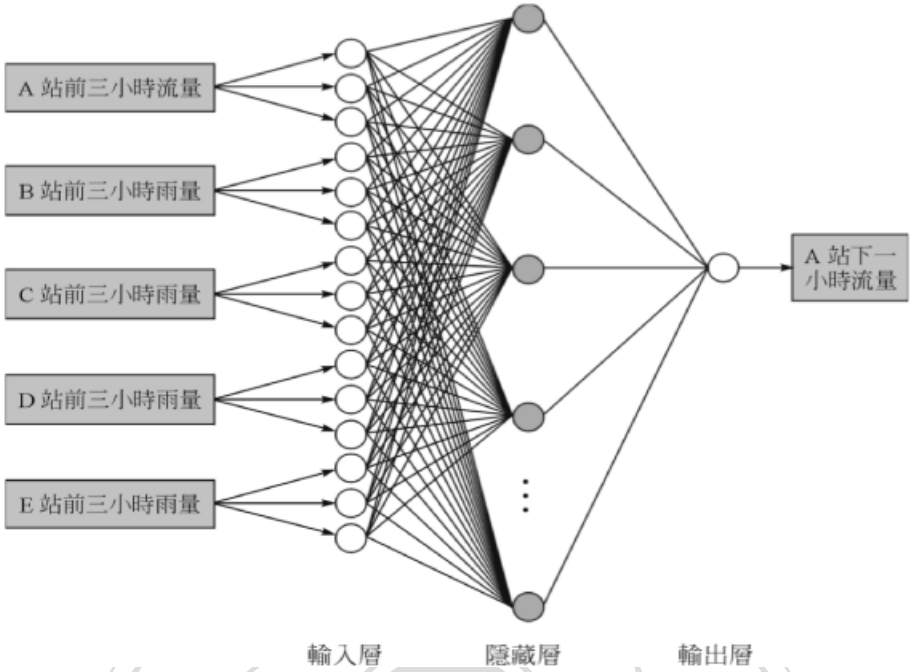


圖 2.13: TDNN 中，單一神經元接收 n 種輸入訊息，m 階時間稽延的示意圖

pattern)所映射的輸出關係。這樣以前饋示網路配合倒傳遞演算法的方式來解決具有時間相關的問題，主要的困難在於網路輸入項的時間稽延個數 m 難以決定，如果稽延個數定的太長，容易造成輸入向量過於龐大使得網路難以訓練；另外一個問題爲 TDNN 是屬於靜態類神經網路，網路結構指能處理與輸入型態 (input pattern) 相一致的類比資料(analog data)，對於非穩定程序(nonstationary processes)，此一方式就難以處理。

靜態類神經網路往往會因爲輸入維度太大，而增加網路的複雜度，使得期計算時間耗時，以下我們以圖 2.13 爲例來做說明:

假設集水區內有 A～E 五個水文站，A 爲流量站，B～E 爲雨量站，每個站這一小時、前一小時，前二小時的水文訊息，都可能影響 A 站下一小時的流量大小；若以靜態類神經網路爲架構來預測 A 站下一小時的流量，則必須輸入 A～E 這一小時、前一小時、前二小時等三筆輸入資料，其輸入維度爲 $5 \times 3 = 15$ 個，因此輸入層至隱藏層的連結權重向量大小爲 $15 \times K$ (K: 隱藏層個數)；此一作法大大提高網路的訓練時間與複雜度。當模式架構完成後，因固定了「輸入爲三小時內的資料 ── 輸出爲下一小時的資料」的網路輸入-輸出型態 ，故只適用於此集水區的特定狀況，且網路所記憶的僅爲

集水區這三小時內的資訊，對於超過三小時的訊息則完全沒有記憶。

TDNN 的特點在於理論簡單易明瞭、應用也較爲普及，但需要大量的訓練資料、容易陷入局部最小值、訓練時間長、網路收斂慢等，造成實際應用上的困擾；尤其在對時間序列方面的問題進行預測時，爲了達到模擬效果，網路的輸入往往使用區塊式的輸入方式，將相連時間的資料串成一個輸入向量，雖然包含了前幾時刻得資訊，但這樣並無法有效建立區塊內的時序行爲；簡單的來說，TDNN 對於動態系統或相關時間序列的處理能力較爲薄弱，當網路架構完成進行預測時，則由「樣本比對」方式將資料代入網路中，因此在不同時間點預測相同輸入資料時，TDNN 會產相同的輸出值，因而無法有效抓住時間序列的特性。

第三章　輻狀基底函數類神經網路

3.1　前言

簡單的說輻狀基底函數類神經網路(Radial Basis Function Networks , RBFN)是由輻狀基底函數(Radial Basis Function)做爲活化函數(activation function) [5]所構成的類神經網路，又稱半徑式類神經網路，具備良好的映射能力。 其架構具有輸入層、一層隱藏層以及輸出層，屬於基本前饋式類神經網路(Feedforward Artificial Neural Network) [14]。

它的結構類似於兩層的認知機(perceptron)，輸入層和隱藏層間的每個神經元完全相互連結。輻射基底函數神經網路和傳統認知器基本上的不同，在於隱藏層神經元的轉移函數(transfer function)。傳統的認知器中，隱藏層神經元的輸出只與它的輸入有關。但是，輻射基底函數神經網路的隱藏層神經元輸出不僅和它的輸入有關，並且又與稱爲中心點(center)的新增向量參數有關。其主要特性有三: (1)僅有三層結構。(2)輸出層全由線性處理單元組成。(3)隱藏層轉換函數是由輸入資料與神經元中心之距離決定 [15]。

定義 3.1.1 **輻**狀基底函數是由至中心點的距離所構成的函數。換句話說，如果 $\phi(X) = h(||X - C||)$ ，則 $h : \mathbb{R}^+ \cup \{0\} \longrightarrow \mathbb{R}$ 是一個幅狀基底函數，如此幅狀基底函數 h 將實數上的向量通過距離表示爲一個純量函數。ϕ 爲隱藏層的活化函數。

3.2　建構 RBF 網路

如果我們考慮 p 個在 $\mathbb{R}^n$ 的點所成的集合與其相對應在 $\mathbb{R}$ 上的期望輸出值表示如下:

$$D = \{(X_i, y_i) \in \mathbb{R}^n \times \mathbb{R},\ 1 \leq i \leq p \mid f(X_i) = y_i\}$$

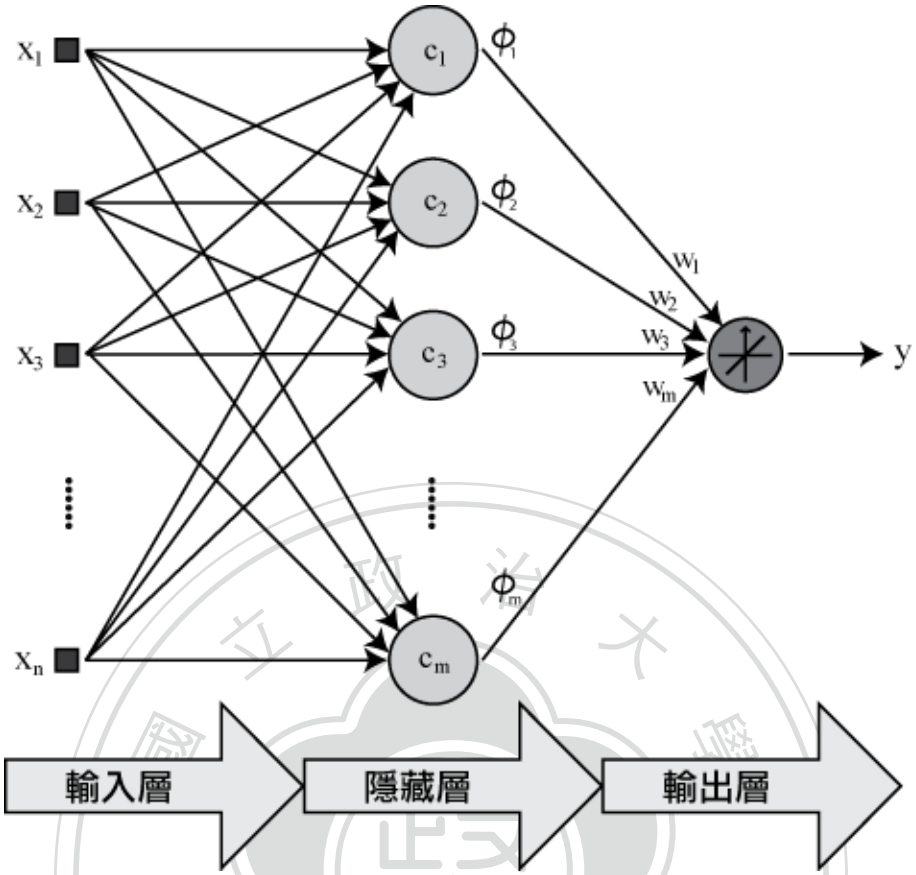


圖 3.1: 輻狀基底函數類神經網路架構圖

上面的集合 D 資料明顯描述輸出值爲一維的函數 f，若輸出值爲高維度時，我們可以分別考慮拆開輸出值的每一項，所以在這篇論文裡我們只考慮一維的輸出值。

RBF 類神經網路主要概念是建構許多輻狀基底函數 $h(||X - C||)$，以函數逼近法找出輸入與輸出間的映射關係。其架構主要包含輸入層、隱藏層及輸出層，每一層扮演不同的角色；輸入層是輸入資料與輻狀基底函數網路連結的介面層，其工作是讀取訓練資料 X 並由資料點隨機選取中心點 C；隱藏層將輸入資料通過非線性函數進行非線性映射到隱藏空間，建立各中心點相對應的輻狀基底函數 $h(||X - C||)$；輸出層則是將隱藏層的輸出進行線性組合而求得輸出值。

以建立 p 個訓練範例資料 $X = \{X_1, X_2, \cdots, X_p\}$，$X_i \in \mathbb{R}^n$、$m$ 個神經元(即中心點個數)的隱藏層之 RBFN 爲例，其中神經元個數即是中心點的數目，其架構如圖 3.1 所式:

在輸入層中，將訓練範例資料輸入網路後選取中心點以作爲隱藏層的神經元中心

點，並將輸入向量傳至隱藏層中的每個輻狀基底函數，其中中心點的選取方法有多種形式，於下節會介紹兩種；在隱藏層中計算各輸入向量與神經元中心點的距離，經由非線性函數的轉換，獲得各神經元的輸出值如 (3.2.1) 式

$$\phi_i(X) = h(||X - C_i||) \quad i = 1, 2, ..., m \tag{3.2.1}$$

上式中 ϕ_i 爲隱藏層中第 i 個中心點的輻狀基底函數，C_i 表隱藏層中第 i 個神經元中心點，共 m 個神經元中心點； $||X - C_i||$ 表示欲求點 X 與中心點 C_i 的距離(通常使用歐式距離)；並於隱藏層中經過加權傳至輸出層後線性組合求得輸出值，其中 w_i 爲隱藏層第 i 個神經元至輸出值的權重值，ϕ_i爲隱藏層第 i個神經元輸出值，y 爲輸出層中的輸出值， 如 (3.2.2) 式

$$y(X) = \sum_{i=1}^{m} w_i \cdot \phi_i(X) = \sum_{i=1}^{m} w_i \cdot h(||X - C_i||) \tag{3.2.2}$$

其中 h 通常是非線性的，也就是我們剛剛提到在隱藏層中的非線性轉移函數，下面將介紹幾種常見的形式，如圖 3.2 所示:

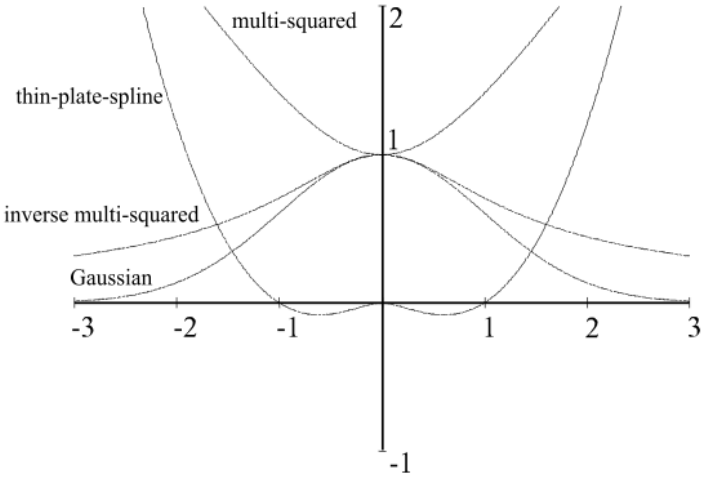


圖 3.2: 幾種輻狀基底函數形式

1. 線性函數(linear function)

$$h(||X - C||) = ||X - C|| \tag{3.2.3}$$

2. 三次函數(cubic function)

$$h(||X-C||) = ||X-C||^3 \tag{3.2.4}$$

3. Duchon 多變數曲線(Duchon multivariate splines)

$$h(||X-C||) = \begin{cases} ||X-C||^{2m-n} ln\,||X-C|| & , 2m > n \text{ 且 } n \text{ 是偶數} \\ ||X-C||^{2m-n} & , \text{其他情形} \end{cases} \tag{3.2.5}$$

4. 薄平面曲線函數(thin-plate-spline function)

$$h(||X-C||) = ||X-C||^2 ln\,||X-C|| \tag{3.2.6}$$

5. 高斯函數(Gaussian function)

$$h(||X-C||) = exp(-\frac{||X-C||^2}{\sigma^2}) \tag{3.2.7}$$

6. 二次多變數函數(multiquadratic function)

$$h(||X-C||) = \sqrt{||X-C||^2 + \sigma^2} \tag{3.2.8}$$

7. 二次多變數倒函數(inverse multiquadratic function)

$$h(||X-C||) = \frac{1}{\sqrt{||X-C||^2 + \sigma^2}} \tag{3.2.9}$$

範例 3.2.1 假設以原點爲中心點，輸入的資料 $X \in \mathbb{R}^2$，輸出值是一個常數且幅狀基底函數 $h(||X-C||) = ||X-C||^3$，則:

$$y(x_1, x_2) = w(x_1 + x_2)^{3/2}$$

範例 3.2.2 假設有兩個中心點，分別是 $c_1 = -1$、$c_2 = 2$，輸入的資料 $x_1, x_2 \in \mathbb{R}$，轉移函數選擇高斯函數且其 $\sigma = 1$，則:

$$y(x_1, x_2) = w_1 e^{(x_1+1)^2} + w_2 e^{(x_2-2)^2}$$

RBFN 在學習過程中一般採用兩階段是的混和學習法，即前階段訓練採用非監督式學習，如下一節將提到的垂直最小平方法(Orthogonal Least-Squares, OLS)，後階段訓練則爲監督式學習方式。

3.3 RBF 中心點之選取

中心點的選取對於 RBFN 的計算有直接的影響,其原因有(1)中心點的數目，會影響計算的效率，降低中心點個數，可有效的降低網路的複雜度。(2)輻狀基底函數是以各資料點對中心點取歐式距離的函數，因此決定中心點的個數，同時也決定了整個網路的大小。(3)中心點的位置間接影響到網路訓練的收斂速度。然而，現今並沒有特定的方法能決定中心點的個數使 RBFN 可既有效率而且準確的進行模擬計算。因此，乃以複合式學習法於隱藏層前階段選取中心點，後階段進行權重向量的修正。

一般而言，RBFN 學習策略會因中心點選取方法不同而有所改變。 而大體上，學習策略均以求得誤差的最小平方和爲目標。

中心點選取法大致分爲隨機性、聚類法(clustering)以及監督式選取等三種；其中聚類法是將輸入資料依照相似的程度予以分類，對於輸入資料較單純的數值回歸運算，在此不納入應用。故本節主要是說明中心點的隨機選取法，以及屬於監督式選取的垂直最小平方法。

3.3.1 隨機選取法

此爲 RBF 中心點選取最簡單的方法，由訓練範例資料點中隨機選取 m_1 個作爲隱藏層的中心點，並求出中心點與中心點間的最大距離 d_{max}，爲避免所有的輻狀基底函數過度陡峭或過度平緩，建議標準偏差值 σ 爲 (3.3.1) 式，其爲一經驗公式(Ham, 2001)。

$$\sigma = \frac{d_{max}}{\sqrt{m_1}} \tag{3.3.1}$$

因此，若輻狀基底函數如高斯函數 (3.2.7) 時

$$h(||X - C_i||) = exp(-\frac{m_1}{d_{max}^2}||X - C_i||^2) \quad i = 1, 2, ..., m \tag{3.3.2}$$

此中心點選取法的優點爲快速、容易，但其缺點則爲須具備大量的訓練資料，經由隨機選取的中心點才具有代表性，而且當輸入空間範圍很大時，中心點的個數要夠多結果才較爲可靠。

3.3.2 垂直最小平方法

垂直最小平方法(Orthogonal Least Squares, OLS)是由 Chen (1991)、Ham

(2001)、Kecman (2001)等人提出。其選取中心點選取的主要概念是將所有訓練範例的輸入點皆視爲潛在的中心點，而每一次的選取標準則是以網路輸出誤差的下降率(error reduction ratio)最大的輸入點爲新的中心點；換句話說，一次最多加入一個能使輸出誤差下降最多的資料點作爲新的中心點，直到輸出層誤差值達到可接受的程度爲止。而利用 OLS 演算法來選取中心點是根據 Gram-Schmidt 垂直理論而發展出來的 [2]。

定義 3.3.1 假設 $A \in \mathbb{R}^{m\times n}$，若 $X \in \mathbb{R}^{n\times 1}$ 使得 $||AX-b||$ 爲最小，則稱 x 爲 $||AX-b||$ 的最小平方解*(least square solution)*

定理 3.3.2 假設 $A \in \mathbb{R}^{m\times n}$，$b \in \mathbb{R}^{m\times 1}$ ，則 $X \in \mathbb{R}^{n\times 1}$ 使得 $||AX-b||$ 爲最小 $\Leftrightarrow A^TAX = A^Tb$

假設 $\{(X_i, d_i)\}_{i=1}^{p}$ 當做學習資料，其中 X_i 是輸入值， d_i 是輸入值相對應的資料值，由 (3.2.2) 式我們可知 $\{y(X_i)\}_{i=1}^{p}$ 爲經過 RBFN 輸出的結果；我們當然期望 $y(X_i)$ 的值與 d_i 是相等的。

$$y(X_i) = d_i \quad for \;\; i = 1, 2, ...p \tag{3.3.3}$$

然而 (3.3.3) 式中等式左邊與右邊當然是不相等的，下面我們將討論如何找出最佳的權數(weights) $\{w_i\}_{i=1}^{m}$ 使得 $||y(X)-d||$ 最小。

在 p 是有限的情況下，我們將 (3.2.2) 式與 (3.3.3) 式重新寫成 (3.3.4) 式

$$\sum_{i=1}^{m} w_i \begin{bmatrix} \phi_i(X_1) \\ \phi_i(X_2) \\ \vdots \\ \phi_i(X_p) \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_p \end{bmatrix} \tag{3.3.4}$$

假設

$$A = \begin{bmatrix} h(||X_1 - C_1||) & h(||X_1 - C_2||) & \cdots & h(||X_1 - C_m||) \\ h(||X_2 - c_1||) & h(||X_2 - C_2||) & \cdots & h(||X_2 - C_m||) \\ \vdots & \vdots & \vdots & \vdots \\ h(||X_p - C_1||) & h(||X_p - C_2||) & \cdots & h(||X_p - C_m||) \end{bmatrix} \tag{3.3.5}$$

不失一般性的我們可以假設 A 爲行向量獨立(因爲中心點個數往往小於資料輸入量，也就是說 $m \leq p$)，如此我們可以確保 A^TA 可逆；由定理 3.3.2 我們可求得最小平方解

$$W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix} = (A^TA)^{-1}A^T \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_p \end{bmatrix} \tag{3.3.6}$$

以及網路輸出值 Y

$$Y = \begin{bmatrix} y(X_1) \\ y(X_2) \\ \vdots \\ y(X_m) \end{bmatrix} = A(A^TA)^{-1}A^T \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_p \end{bmatrix} \tag{3.3.7}$$

定理 3.3.3 (Gram-Schmidt 正交化過程)

假設 V 爲分布於 F 的內積空間，$S = \{v_1, v_2, ...v_n\} \subseteq V$ 爲線性獨立集，令 $u_1 = v_1$ 且

$$u_k = v_k - \sum_{i=1}^{k-1} \frac{< v_k, u_i >}{||u_i||^2} u_i \quad , 2 \leq k \leq n$$

,則 $\{u_1, u_2, ...u_n\}$ 爲不含零向量的正交集且 $span\{u_1, u_2, ...u_n\} = span\{v_1, v_2, ...v_n\}$

接下來我們將定義誤差的下降率(error reduction ratio)，首先我們將 (3.3.4) 加入誤差項 $\{e_i\}_{i=1}^p$ 讓等式成立:

$$\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_p \end{bmatrix} = \sum_{i=1}^{m} w_i \begin{bmatrix} \phi_i(X_1) \\ \phi_i(X_2) \\ \vdots \\ \phi_i(X_p) \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_p \end{bmatrix} \tag{3.3.8}$$

並且將 (3.3.5) 式的 A (經過輻狀基底函數轉化得到的矩陣)分解成 QR 如 (3.3.9) 式

$$A = QR \quad , \text{其中 } R \in \mathbb{R}^{m \times m} = \begin{bmatrix} 1 & r_{12} & \cdots & r_{1m} \\ 0 & 1 & \cdots & r_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{3.3.9}$$

上式中 $Q \in \mathbb{R}^{p\times m}$ ，爲 A 矩陣經過 Gram-Schmidt 正交化過程得到的結果，並滿足 $Q = [q_1, q_2, ..., q_m]$，其中 $\{q_i\}_{i=1}^{m}$ 爲正交基底向量且 $Q^TQ = diag(h_1, h_2, ...h_m)$ 爲正對角矩陣(positive diagonal matrix)：

$$h_i = {q_i}^T q_i = \sum_{k=1}^{p} {q_{ki}}^T q_{ki} \quad 1 \le i \le m \tag{3.3.10}$$

結合 (3.3.7) 式以及 (3.3.9) 式，我們可以將 (3.3.8) 式改寫成 (3.3.11) 式

$$D = AW + E = QRW + E \quad ,\text{其中} D = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_p \end{bmatrix} \quad E = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_p \end{bmatrix} \tag{3.3.11}$$

令 $G = RW$，則 $D = QG + E$ 既目標輸出值 D 表示爲垂直基底向量 $\{q_i\}_{i=1}^{m}$ 的線性組合，由最小平方法可得最佳解 $G \in \mathbb{R}^{m\times 1}$ 爲 (3.3.12) 式和 (3.3.13) 式

$$G = (Q^TQ)^{-1}Q^TD \tag{3.3.12}$$

$$G = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_m \end{bmatrix} \quad , g_i = \frac{{q_i}^T D}{{q_i}^T q_i} \quad , i = 1, 2, ...m \tag{3.3.13}$$

最後由於 Q 是垂直基底向量矩陣，所以目標輸出值 D 的平方和可表示爲 (3.3.14) 式

$$D^TD = G^TQ^TQG + E^TE = \sum_{k=1}^{m} {g_k}^2 h_k + E^TE \tag{3.3.14}$$

定義 3.3.4 誤差下降率*(err)*$= \dfrac{D^TD - E^TE}{D^TD} = \dfrac{\sum_{k=1}^{m} {g_k}^2 h_k}{D^TD} = \dfrac{\text{網路輸出值}}{\text{目標輸出值}}$

由定義可以看出誤差下降率(err)爲網路輸出值與目標輸出值的比值，此比值越高代表模式推估能力越好，而對應第 t 個中心點時，誤差下降率爲：

$$[\text{err}]_t = \frac{{g_t}^2 h_t}{D^TD} = \frac{(q_t^TD)^2}{q_t^Tq_tD^TD} \tag{3.3.15}$$

在計算所有輸入項的誤差下降率後，取其造成最大之誤差下降率即 $[\text{err}]_{max}$ 的輸入項爲隱藏層第一個神經元的中心點，在扣除該輸入項資料後，依上述步驟重新計算，挑選剩餘的輸入項中造成誤差下降率最大值的輸入項爲下一個神經元的中心點，並將所有的最大誤差下降率累計起來 $\sum[\text{err}]$，如此依序增加隱藏層之神經元數目，直到整個網路架構滿足所設定的容忍誤差(tolerance)爲止。假設此時選取的中心點個數爲 m 個:

$$1 - \sum_{i=1}^{m} [\text{err}]_i < \text{容忍誤差} \tag{3.3.16}$$

$$\sum_{i=1}^{m} [\text{err}]_i > \text{設定的正確率}$$

3.3.3 OLS 中心點選取法計算步驟

(1) 令 $q_1^{(i)} = \phi_i \quad , i = 1, 2 \ldots p$ 代入 (3.3.15) 式，並找出 $[\text{err}]_{k_1} = max\{\, [\text{err}]_i \quad , 1 \leq i \leq p\}$，$k_1$ 爲第 1 次挑選到的資料點，則 $q_1 = \phi_{k_1}$，$c_1 = c_{k_1}$

(2) 將剩下的 ϕ_i 與 q_1 分別作正交化得到 $q_2^{(i)}$ 再次代入 (3.3.15) 式，並找出 $[\text{err}]_{k_2} = max\{\, [\text{err}]_i \quad , 1 \leq i \leq p \quad , i \neq k_1\}$，$k_2$ 爲第 2 次挑選到的資料點，可得到 $q_2 = \phi_{k_2}$，$c_2 = c_{k_2}$；同理我們可以整理出

$$a_{nj}^{(i)} = \frac{q_j^T \phi_i}{q_j^t q_j} \quad , j = 1, 2 \ldots n-1 \quad , \forall n \geq 2$$

$$q_n^{(i)} = \phi_i - \sum_{j=1}^{n-1} a_{nj}^{(i)} \, q_j \quad , 1 \leq i \leq p \quad , i \neq k_1, \ldots i \neq k_{n-1}$$

$$[\text{err}]_{k_n} = max\{\, [\text{err}]_i \quad , 1 \leq i \leq p \quad , i \neq k_1, \ldots i \neq k_{n-1}\}$$

直到滿足 (3.3.16) 式爲止。

(3) 最後計算連結權重(synaptic weight) w_i 得到輸出值:

$$G = RW$$

$$y(X) = \sum_{i=1}^{m} w_i \cdot \phi_i(X)$$

3.4 動態輻狀基底函數類神經網路

靜態類神經網路(Static Neural Network)例如 RBF、TDNN 等，它們都有共同特徵就是沒有遞迴項。靜態類神經網路對資料結構的長期(long-term)存在關係可以有良好解析效果，但對於動態系統的暫時性(temporal)或短暫性(short-term)的動態變化就難以依實際情況加以調整掌握，其網路輸出結果完全由外界輸入項，與連結權重決定，即輸入項確認後，導入以訓練好的網路只會有一種輸出值，由於缺少遞迴項，無法將具有時便性的資訊納入網路中考量，故這類網路在先天上屬於穩定靜態的。

所謂的動態類神經網路指的是神經元間的回饋動作，雖然倒傳遞類神經是以監督式學習方式進行訓練，也有一個回傳的動作，是將誤差回傳來調整權重，以減少網路輸出值與目標值之間的差距，但這樣的動作僅在訓練階段調整權重，待訓練完成後權重值就不在變動，故此一誤差回傳動作屬於網路外進行，並不是發生在網路內部神經元的架構內，因此只能算是一種靜態的類神經網路。

簡單的來看，我們可以在靜態類神經網路架構中加入遞迴項，使其成爲動態的類神經網路，具有動態記憶或稱爲短暫記憶(short-term memory)。一般來說，網路的記憶分爲兩種，一種爲長期記憶，另一爲短期記憶;長期記憶或稱爲穩定的記憶，是經由監督示學習將訓練網路資料建入網路，儲存於網路連結權重(synaptic weight)上；當網路所需要處理的問題具有時間性尺度(temporal dimension)時，則需要暫時的記憶來儲存距離現在不久前的過去資料。動態神經網路除了維持靜態神經網路計有強健推估能力特性外，更提供對於生物神經網路在有關於短期記憶功能上，有更佳的處理方式 [6]。

第四章　T-RBF (Temporal RBF) 方法

回顧一開始提到的時間稽延類神經網路(TDNN)，它很成功的使用時間稽延的方法結合在前饋式類神經網路當中，然而其缺點也是很明顯的，就是在於一開始就固定了稽延的時間在整個訓練過程當中，如何決定稽延時間的個數以及每次稽延最適當的長度常常會造成很大的困擾。

標準的 RBF 可以幫助我們完成複雜的非線性辨識問題，但限制於固定的靜態問題而不是隨著時間變動的問題，靜態類神經網路在處理時間性問題上的缺點就是在隱藏層與輸出層之間無法對暫時性模式(temporal pattern)的輸入有回應的動作，爲了有所回應，我們必須在網路的每一層給予時間稽延。

T-RBF(Temporal RBF)，就像是自適應稽延類神經網路(Adaptive Time Delay Neural Network , ATDNN) [7] [8]、局部暫時性記憶(Local Short Term Memory , LSTM)等，已經被提出來可以解決這類問題的限制。在整個訓練中適應稽延時間(time delay)及權數，這是一種動態的學習方法，這可以讓我們的網路在應用程序上可以動態的隨著時間變動 [4]。

4.1　T-RBF 網路架構

在建構 T-RBF 網路架構之前，首先我們來定義稽延區塊(delay block):

定義 4.1.1 時間框架*(Time frame)* T_{ji} 是由稽延時間所成的集合 $(\tau_{ji1}, \tau_{ji2}.....\tau_{jin})$，其使用在前一層的節點 i 到下一層的節點 j 的連結之間；其中 n 爲稽延的長度。

定義 4.1.2 稽延區塊*(delay block)*，如圖 *4.1* 所示，由前一層的節點 i 到下一層的節點 j 的連結中我們放置一組稽延區塊所構成，圖中的 N_k 表示第 k 層。

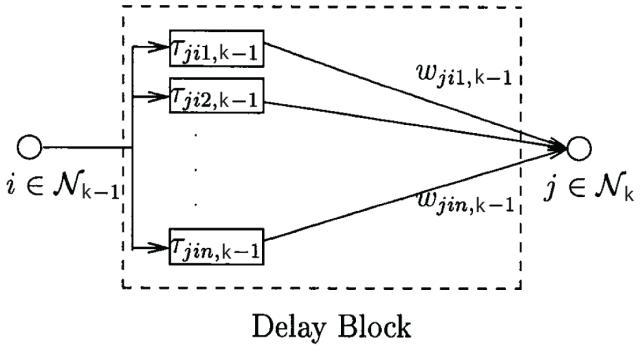


圖 4.1: 介於 N_{k-1} 層的第 i 個節點與 N_k 層第 j 個節點的 Delay Block

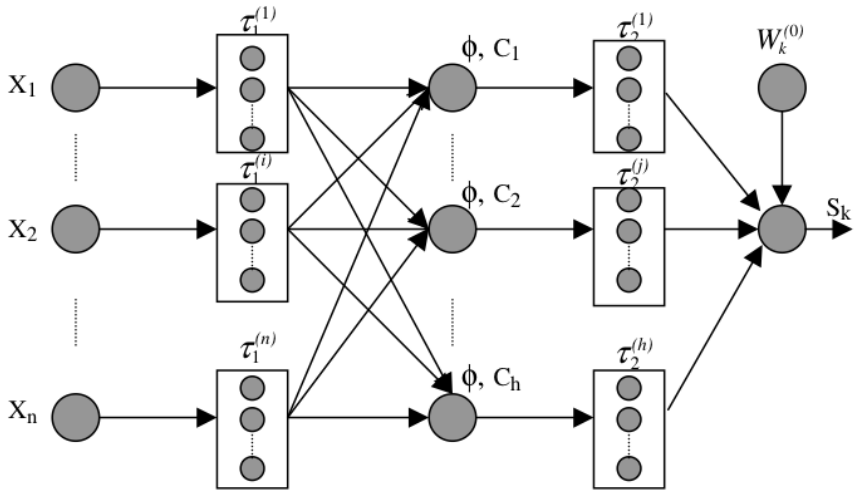


圖 4.2: T-RBF 模型

如圖 4.2 所示，我們建構一個 n 個輸入項的輸入層，整個輸入層對應一個稽延區塊(delay block) τ_1，h 個節點的隱藏層，整個隱藏層對應一個稽延區塊 τ_2，最後是一個輸出層。而圖 4.3 中，$\tau_1^{(i)}$ 表示第 i 個輸入值所對應的稽延區塊，且 $dim(\tau_1^{(i)}) = L$，$\tau_2^{(j)}$ 爲第 j 個中心點活化函數所對應的稽延區塊，且 $dim(\tau_2^{(j)}) = m$。

接下來我們將定義輸出函數:

$$S_k = \varphi\left(W_k^0 + \sum_{j=1}^{h}\sum_{p=1}^{dim(\tau_2^j)} Y_{jp}\, W_{jpk}^{(2)}\right) \tag{4.1.1}$$

其中 φ 爲輸出的活化函數，它可能是線性的或者 S 形函數(sigmoid form)，W_k^0 爲偏移

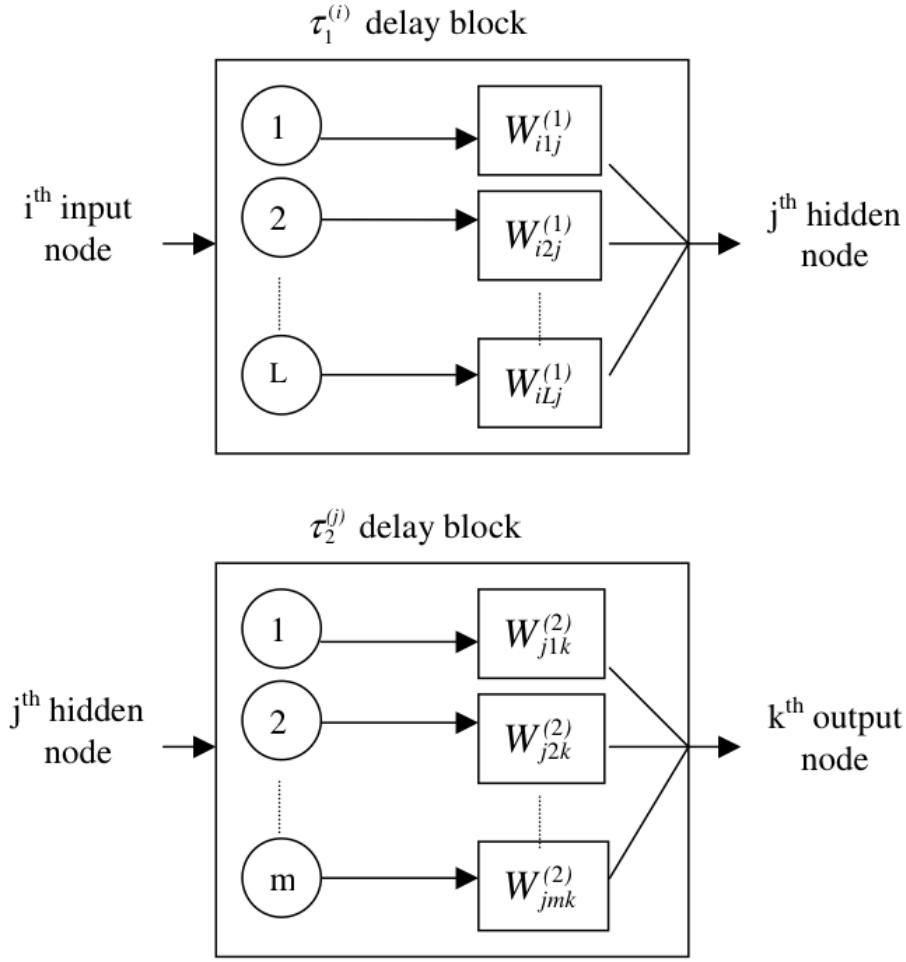


圖 4.3: $\tau_1^{(i)}$ 與 $\tau_2^{(j)}$ 稽延區塊示意圖

量(bias)，$W_{jpk}^{(2)}$ 爲隱藏層和輸出層之間的權數值。其中 Y_{jp} 爲隱藏層中的活化函數:

$$Y_{jp} = \sum_{i=1}^{n} \sum_{q=1}^{dim(\tau_1^i)} W_{iqj}^{(1)} \, \phi(||X_{iq} - C_{jp}||) \tag{4.1.2}$$

在 (4.1.2) 式中 ϕ 爲輻狀基底函數，$W_{iqj}^{(1)}$ 表示輸入層與隱藏層的權數，C_{jp} 爲在第 j 個中心點的第 p 個時間分量，X_{iq} 表示第 i 個輸入值的第 q 個時間分量。

4.1.1 網路展開平行演算法(Network Unfolding Algorithm)

在上述網路架構裡，我們發現一個困難的地方就是在於如何計算隱藏層中每一個稽延區塊的活化函數；除非每個稽延區塊的稽延長度只有 1 (only one delay)。對於如此

問題我們將引進網路展開平行演算法(Network Unfolding Algorithm , NUA) [9]，該演算法能夠簡化圖 4.2，降低了複雜性，如此我們採用此演算法。它主要爲下列兩個階段:

定義 4.1.3 p_{ji}是一個信號值從節點 i 到節點 j 並在時間 t 時通過時間框架*(time frame)* T_{ji} 所成的集合。也就是說，$p_{ji}(t, T_{ji}) = [s_i(t - \tau_{ji1}), ..., s_i(t, \tau_{jin})]$，其中 $s_i(t)$ 是時間 t 從節點 i 發出的信號，n 爲 T_{ji} 的長度。

- 階段一:

  第一個階段有下列四個步驟，在此並沒有以演算法表示，只是敘述演算法概念:

  Step1 展開輸入項:

  拆開稽延區塊裡的每一稽延時間並複製該輸入項與之對應，如圖 4.4 中的 Step1 表示。

  Step2 重新調整輸入的時間遲滯量:

  移除介在輸入層與隱藏層中間的稽延時間，並將輸入項以信號值向量表示，即 $[p_{j1}(t, T_{j1}), ..., p_{jn_I}(t, T_{jn_I})]$，其中 n_I 爲介於輸入層與隱藏層中間的稽延長度，如圖 4.4 中的 Step2 表示。

  Step3 展開隱藏層:

  複製前兩步驟所產生的輸入項，如同 Step1，將隱藏層的第 j 個節點所有的稽延時間拆開並個對應到一組輸入項，如圖 4.4 中的 Step3 表示。

  Step4 重新調整輸入的時間遲滯量:

  如同 Step2 ，將介於隱藏層與輸出層中間的稽延時間以信號值向量的方式併入輸入項中，如圖 4.4 中的 Step4 表示。

  圖 4.5 爲網路平行展開演算法(NUA)的第二種模式，是以先展開介於隱藏層與輸出層的稽延時間開始。

- 階段二： 我們將利用第三章所提到的垂直正交最小平方法(Orthogonal Least Squares, OLS)來選取中心點，完成整個網路的大小。

經過階段一與階段二，我們期望目標爲 (1) 隱藏層與輸出層之間只有權數(weights)，沒有稽延時間。(2) 輸入層與隱藏層之間只有稽延時間，沒有權數。

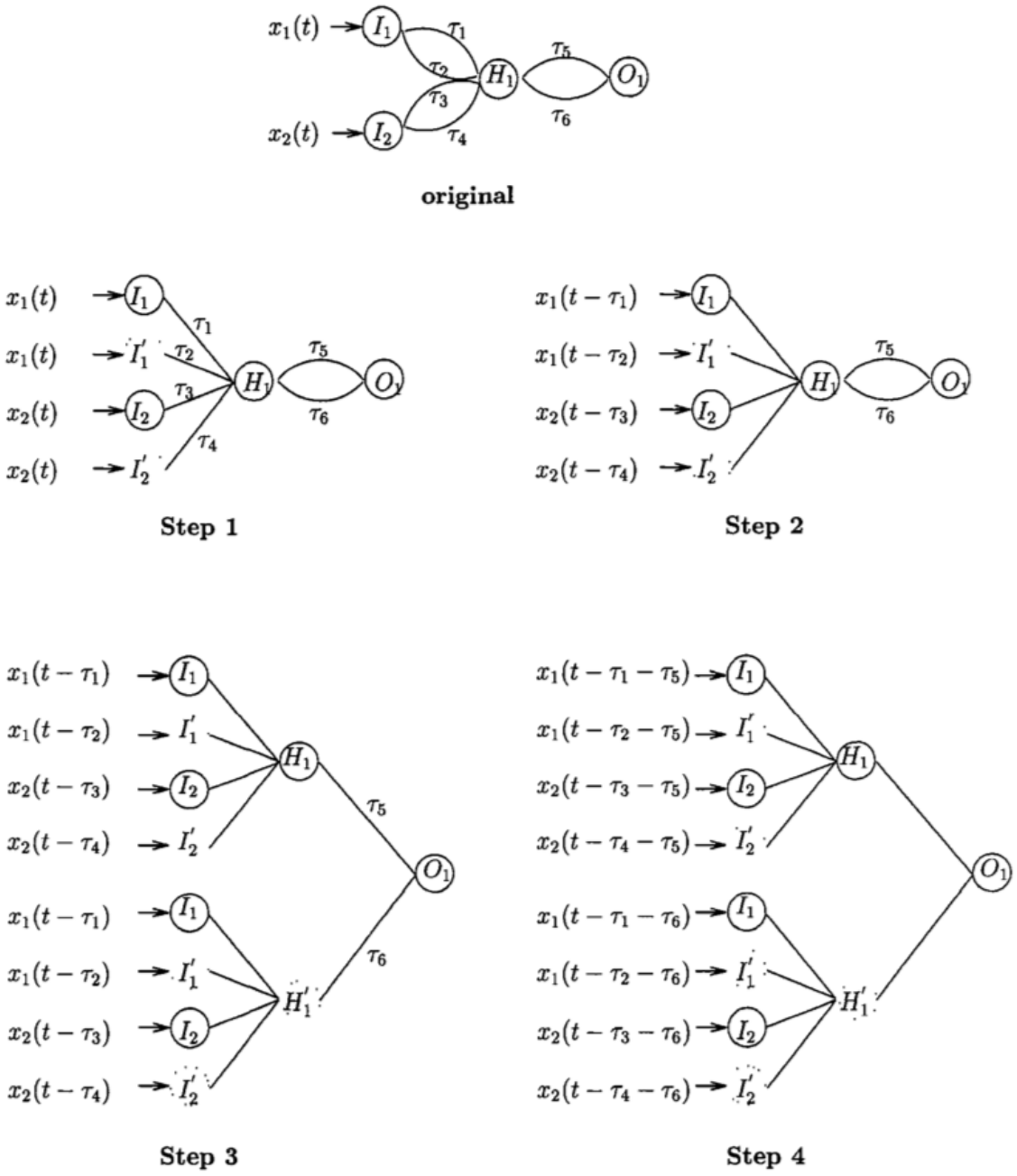


圖 4.4: NUA 展開示意圖

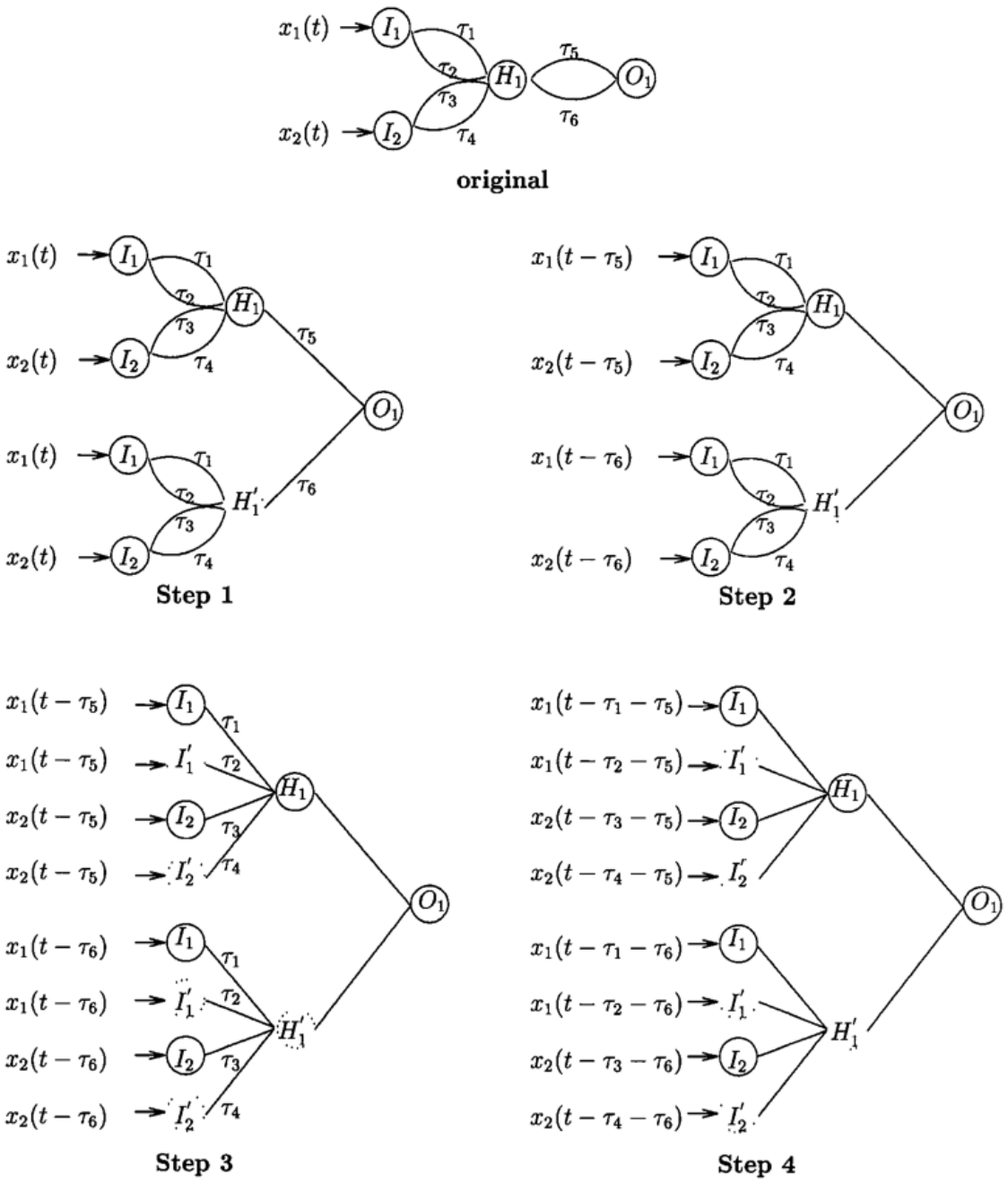


圖 4.5: NUA 展開的第二種模式：先展開第二層稽延時間

第五章　RBF 圖形介面化工具

5.1　Adobe Flex 介紹

隨著 WWW 的興起，應用程式不再侷限於一台電腦上執行，逐漸演化成一個個的網頁，使用者端只需安裝瀏覽器跟一些插件，輸入網址，一切完成。但網頁應用程式究竟與一般傳統應用程式有很大的差異，加上各家瀏覽器對於標準的支援不同，開發出跨平台跨瀏覽器的應用成爲一件吃力不討好的工作。Adobe Flex 正是解決這些問題的方案之一，適用於建立和維護可部署在各種主要瀏覽器、桌上型電腦和作業系統上，具豐富使用者介面的網路應用程式。Flex 是一組工具的集合，除了應用程式的開發環境外，豐富美觀的元件提供了各種建構應用程式所必需的元素。

Flex(在 Flex 3.0 之後更名爲 Flash Builder 4) 是 Adobe 公司所發展用來開發「RIA(Rich Internet Application)」的工具，「RIA」簡單的說就是在 Internet 上面執行的高親和性操作介面的應用程式。然而應用程式的發展大致可以簡單區分爲4個階段:

(1) DOS 應用程式: 文字操作介面。

(2) Window 應用程式:

- 優點:
  (a) 圖形化操作介面，使用者容易操作。
  (b) 程式容易開發，開發工具有: VB、Delphi、PowerBuilder...。
- 缺點:
  (a) 版本控制不易，新版本通常必須更新到使用者的電腦上。
  (b) 跨平台困難，同一份程式無法在不同類型的作業系統上面執行。

(c) 一般應用在區域網路上。

(3) Web 應用程式:

優點:

(a) 版本控制容易，使用者的電腦上只需要安裝瀏覽器。

(b) 應用在 Internet 上。

缺點:

(a) 雖然是圖形操作介面，但是受限於瀏覽器的功能，操作介面的親和性普遍不高。

(b) 缺乏好用的開發工具。

(4) RIA: 結合 Window 和 Web 應用程式的優點。
這樣的發展趨勢在類似 Flex 這樣的開發工具出現之後，應該會變的更快速，可以預見在不久的將來，Internet 上的應用程式都會有更好用的操作介面。

長久以來，大家對應用程式的要求不外乎以下這些特性:

(1) 操作介面有一致性並且容易使用。

(2) 跨平台。

(3) 系統穩定性好。

(4) 程式可以容易並且快速的開發、維護、擴充。

(5) 容易安裝和改版。

(6) 程式人員的教育訓練成本低。

(7) 發展大型系統。

(8) 在 Internet上 執行。

基本上 Flex 提供了滿足這些需求的條件，主要原因是它具備下面這些特性:

(1) 豐富的視覺化元件:

RIA 的特性之一就是操作介面的親和性，這也是 Web 應用程式最大的問題，這些元件是構成操作介面的基本元素。Flex 應用程式和 Web 應用程式不同的地方是: Web 應用程式將原始碼傳送到 Client 的瀏覽器執行，所以操作介面受限於瀏覽器的功能，Flex 應用程式則是將編譯過的程式碼送到 Client 端的 Flash 執行，操作介面可以有更大的彈性。

(2) 物件導向(Object Oriented , OO)特性:

利用物件的封裝和繼承，可以一步一步建構發展系統所需要的元件，便於快速開發穩定的大型系統。 在軟體開發模式中，有一個 80/20 法則，就是用 20% 的時間去開發系統 80% 的程式，再用 80% 的時間，去完成剩下的 20%，這是因爲我們很容易找出大多數系統共同的部分，將這些共同部分包裝成元件，並且套用到 80% 的程式，而不能套用的部分通常是包含特殊的規則或是操作流程，這部分會需要比較多的時間來討論、修改和撰寫。要實現這個 80/20 法則，依賴的就是這個 OO 的特性。

(3) 除錯和編譯。

(4) 跨平台: Flex 使用 Flash 做爲 Client 端的 Virtual Machine 來跟 Flex 應用程式溝通，每個平台有自己版本的 Flash，藉此達到跨平台的目的。

5.2 使用 Adobe Flex 建構 RBF 應用程式

我們將以 Flex 在 Internet 上開發 RBFN 應用程式，由於 ActionScript 3.0 本身並沒有強大的數學函式庫，只有基本的數學運算，所以我們必須先建立一些數學工具以供使用。

(1) 矩陣定義、常用矩陣屬性及工具，程式詳見附錄 A。

(2) 利用 Householder 矩陣來求 QR 分解及相關的應用(包含最小平方解)，程式詳見附錄 B。

(3) RBFN 相關工具，程式詳見附錄 C。

(4) RBFN 隨機選取法，程式詳見附錄 D。

(5) RBFN 垂直最小平方法，程式詳見附錄 E。

5.3 RBFN 圖形介面工具使用說明

5.3.1 設定區畫面

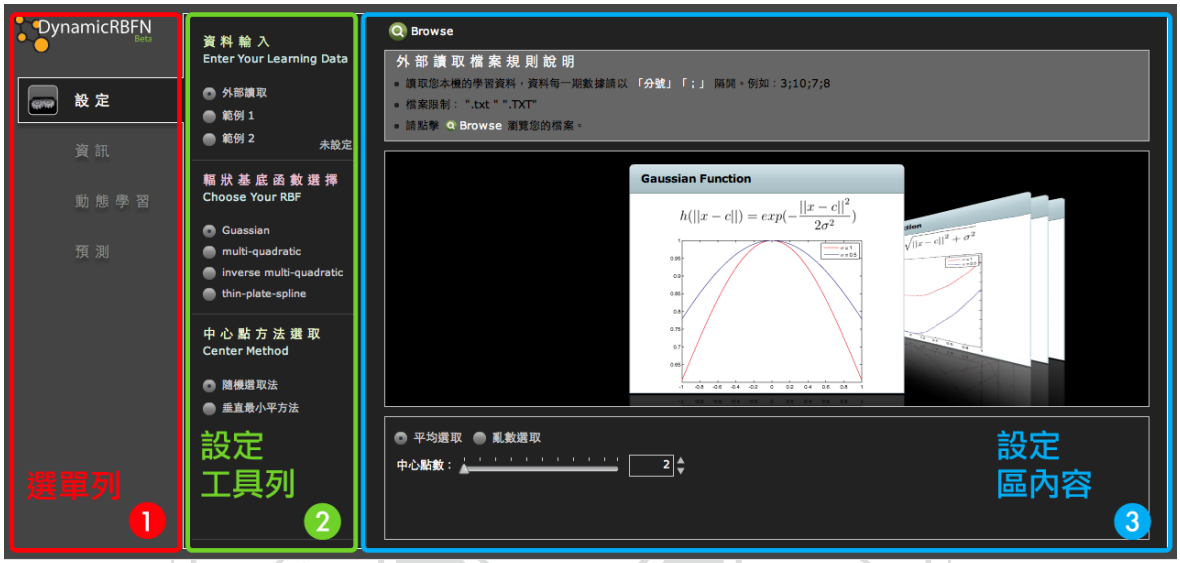


圖 5.1: 設定區畫面示意圖

(1) 選單列 (如圖 5.1 — 1 號區域): 包含設定，資訊，動態學習及預測等功能鍵。

- 設定: 設定您的學習資料，輻狀基底函數及中心點選取方法。
- 資訊: 依照您在設定區所設定後產生的結果資訊(輸出值、眞實值及誤差值)。
- 動態學習: 依照使用者在設定區所設定後產生的結果資訊，可經由動態工具列，變動設定(如變動輻狀基底函數、中心點個數等)，藉此產生動態圖形變化，目前此功能僅提供中心點選取法爲機選取法使用。
- 預測: 在設定區只提供以「當期」預測「下一期」的功能，如需改變如「當期」、「前一期」、「前二期」預測「後二期」可依使用者需由在預測功能區進行變動及預測未來。

(2) 設定工具列 (如圖 5.1 — 2 號區域): 學習資料設定(可選擇外部讀取 (如圖 5.2) 或者範例), 輻狀基底函數設定, 中心點選取法設定。

(3) 設定區內容 (如圖 5.1 — 3 號區域): 在 2 號區域設定的詳細內容及介紹。

圖 5.2: 外部讀取示意圖

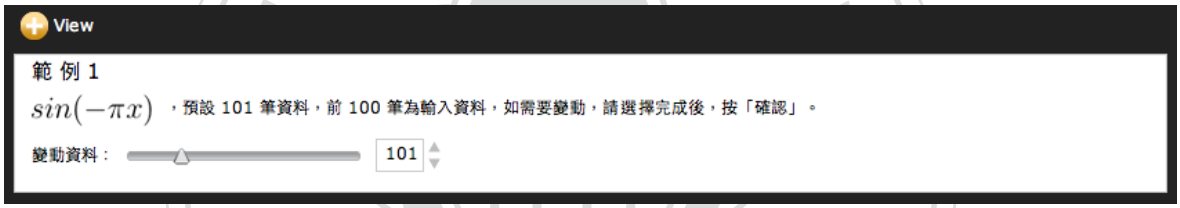


圖 5.3: 範例 $sin(-\pi\, x)$ 示意圖

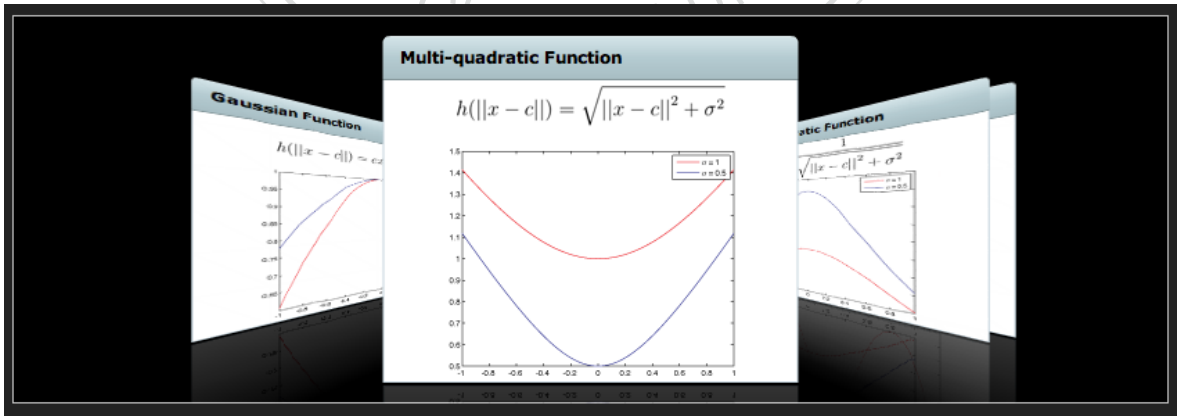


圖 5.4: 輻狀基底函數設定示意圖

圖 5.5: 隨機選取法內容設定示意圖

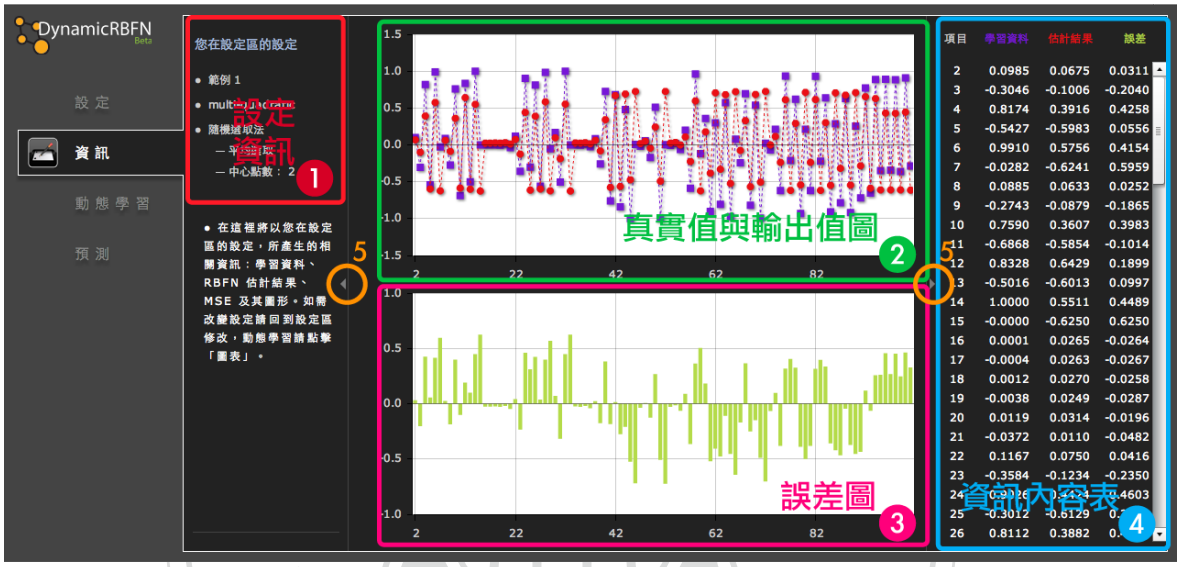


圖 5.6: 資訊區畫面示意圖

5.3.2 資訊區畫面

(1) 選設定資訊列 (如圖 5.6 — 1 號區域)：您在設定區設定的內容。

(2) 眞實值與輸出值圖 (如圖 5.6 — 2 號區域)：眞實值與 RBFN 輸出值對照圖，按著滑鼠左鍵拖拉 (如圖 5.7)，可局部區域放大 (如圖 5.8)。

(3) 誤差圖 (如圖 5.6 — 3 號區域)：眞實值與 RBFN 輸出值誤差圖，按著滑鼠左鍵拖拉，可局部區域放大。

(4) 資訊內容表 (如圖 5.6 — 4 號區域)：項目、眞實值、RBFN 輸出值及誤差。

(5) 放大按鍵 (如圖 5.6 — 5 號按鍵)：點擊可放大眞實值與輸出值圖以及誤差圖 (如圖 5.9)。

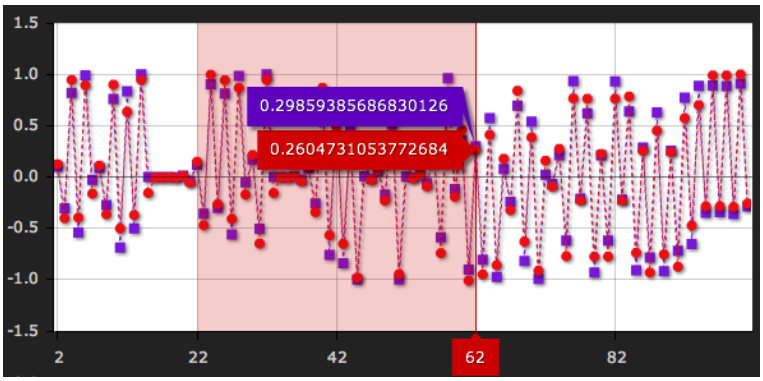


圖 5.7: 滑鼠左鍵拖拉局部放大操作示意圖

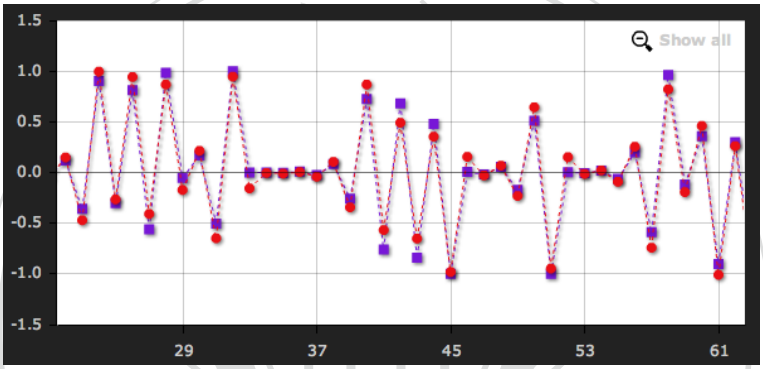


圖 5.8: 局部區域放大示意圖

5.3.3 動態學習區畫面

(1) 動態工具列 (如圖 5.10 — 1 號區域):

a. 輻狀基底函數: 有四種選擇, 分爲別: 高斯函數、二次多變數函數、二次多變數倒函數及薄平面曲線函數。

b. 隨機選取法選取方式: 平均選取與亂數選取。

c. 中心點個數設定。

d. 範圍控制: 圖形局部範圍控制。

(2) 動態圖形區 (如圖 5.10 — 2 號區域): 區域內顯示學習資料(眞實值與輸出值)圖, 有線形圖、點圖以及誤差圖, 依使用者在動態工具列的設定, 動態變動圖形。

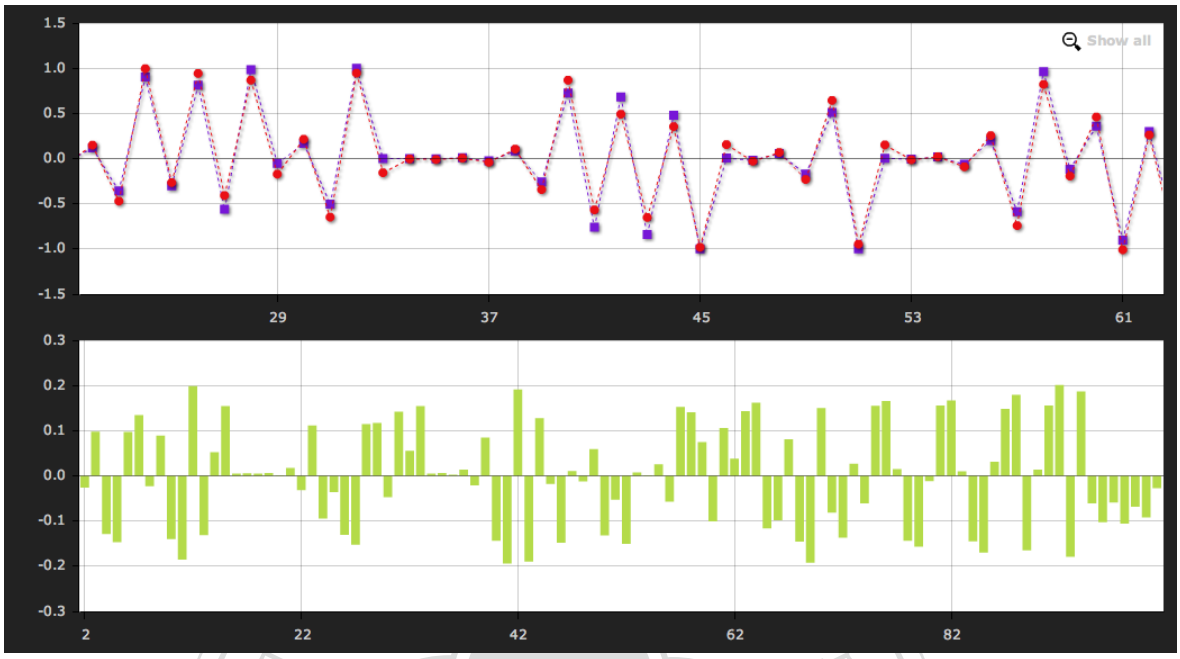


圖 5.9: 點擊放大按鍵示意圖

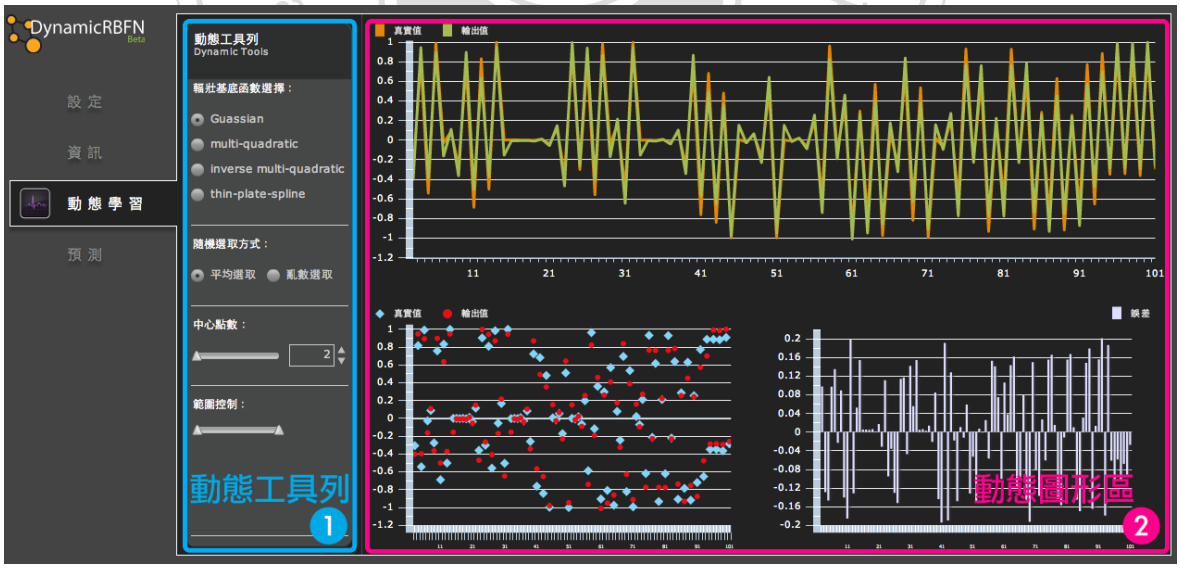


圖 5.10: 動態學習區畫面示意圖

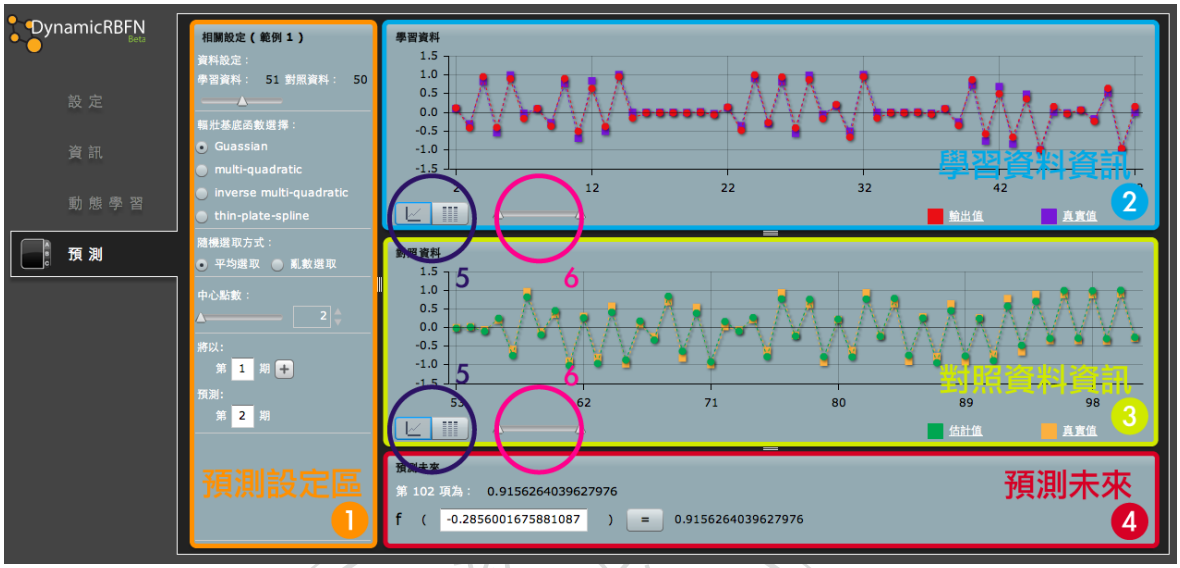


圖 5.11: 預測區畫面示意圖

5.3.4 預測區畫面

(1) 預測設定區 (如圖 5.11 — 1 號區域):

a. 學習資料與對照資料設定：依使用者需求調整資料比例。

b. 輻狀基底函數：有四種選擇，分爲別：高斯函數、二次多變數函數、二次多變數倒函數及薄平面曲線函數。

c. 隨機選取法選取方式：平均選取與亂數選取；若設定爲垂直最小平方法則爲容忍誤差設定，並且沒有 d 項。

d. 中心點個數設定。

e. 預測方式設定：預設以「當期」預測「下一期」，可依使用者需求進行變更。

(2) 學習資料資訊 (如圖 5.11 — 2 號區域)：含學習資料圖(眞實值與輸出值)及學習資料表(眞實值與輸出值)。

(3) 對照資料資訊 (如圖 5.11 — 3 號區域)：含對照資料圖(眞實值與估計值)及對照資料表(眞實值與估計值)。

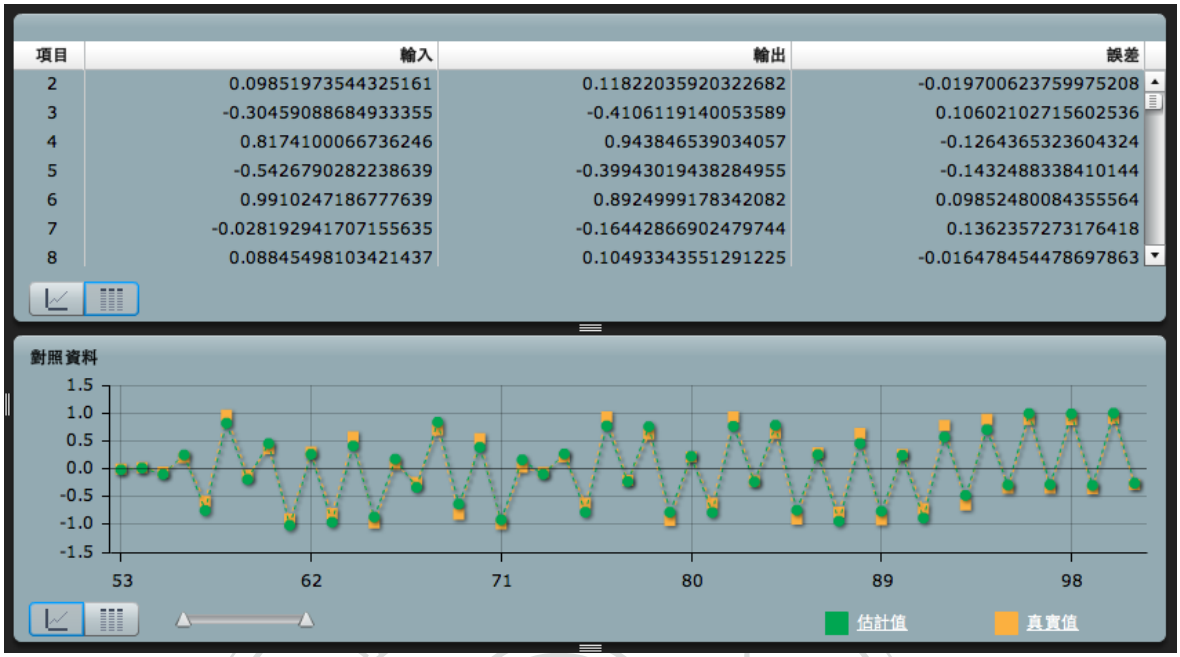


圖 5.12: 切換學習資料圖與學習資料表示意圖

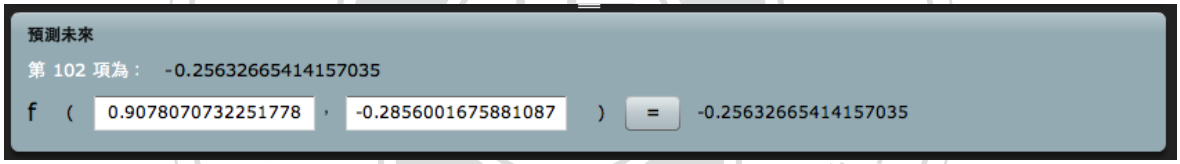


圖 5.13: 預測未來示意圖

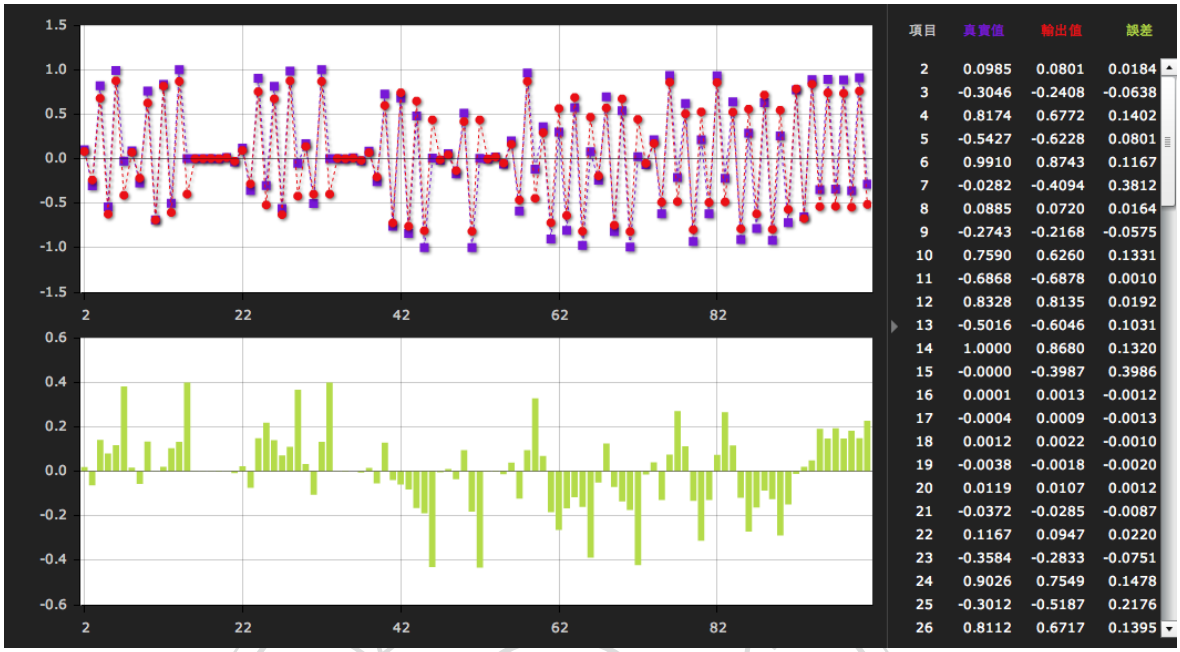
(4) 預測未來 (如圖 5.11 — 4 號區域): 系統直接預測未來第一項，也可自行輸入資料預測未來，如圖 5.13。

(5) 切換鍵 (如圖 5.11 — 5 號按鍵): 切換學習資料圖(對照資料圖)與學習資料表(對照資料表)，如圖 5.12。

(6) 範圍控制 (如圖 5.11 — 6 號拖拉控制): 圖形局部範圍控制。

5.4 操作實例

- 我們以建構 $sin(-\pi x)$ 爲例:



項目	真實值	輸出值	誤差
2	0.0985	0.0801	0.0184
3	-0.3046	-0.2408	-0.0638
4	0.8174	0.6772	0.1402
5	-0.5427	-0.6228	0.0801
6	0.9910	0.8743	0.1167
7	-0.0282	-0.4094	0.3812
8	0.0885	0.0720	0.0164
9	-0.2743	-0.2168	-0.0575
10	0.7590	0.6260	0.1331
11	-0.6868	-0.6878	0.0010
12	0.8328	0.8135	0.0192
13	-0.5016	-0.6046	0.1031
14	1.0000	0.8680	0.1320
15	-0.0000	-0.3987	0.3986
16	0.0001	0.0013	-0.0012
17	-0.0004	0.0009	-0.0013
18	0.0012	0.0022	-0.0010
19	-0.0038	-0.0018	-0.0020
20	0.0119	0.0107	0.0012
21	-0.0372	-0.0285	-0.0087
22	0.1167	0.0947	0.0220
23	-0.3584	-0.2833	-0.0751
24	0.9026	0.7549	0.1478
25	-0.3012	-0.5187	0.2176
26	0.8112	0.6717	0.1395

圖 5.14: 中心點個數: 3，實例操作圖

(1) 設定 101 筆資料，前 100 筆爲網路輸入值，後 100 筆爲眞實值，以「當期」預測「下一期」的方式建構 RBFN。

(2) 輻狀基底函數：高斯函數(Gaussian function)。

(3) 中心點選取方式：隨機選取法平均選取。

(4) 中心點個數：3 個中心點，如圖 5.14。

- 我們同樣以建構 $sin(-\pi x)$ 爲例，但改變中心點個數作爲比較：

(1) 設定 101 筆資料，前 100 筆爲網路輸入值，後 100 筆爲眞實值，以「當期」預測「下一期」的方式建構 RBFN。

(2) 輻狀基底函數：高斯函數(Gaussian function)。

(3) 中心點選取方式：隨機選取法平均選取。

(4) 中心點個數：10 個中心點，如圖 5.15。

比較圖 5.14 與 圖 5.15 明顯看得出來中心點數目影響整個網路的準確度。

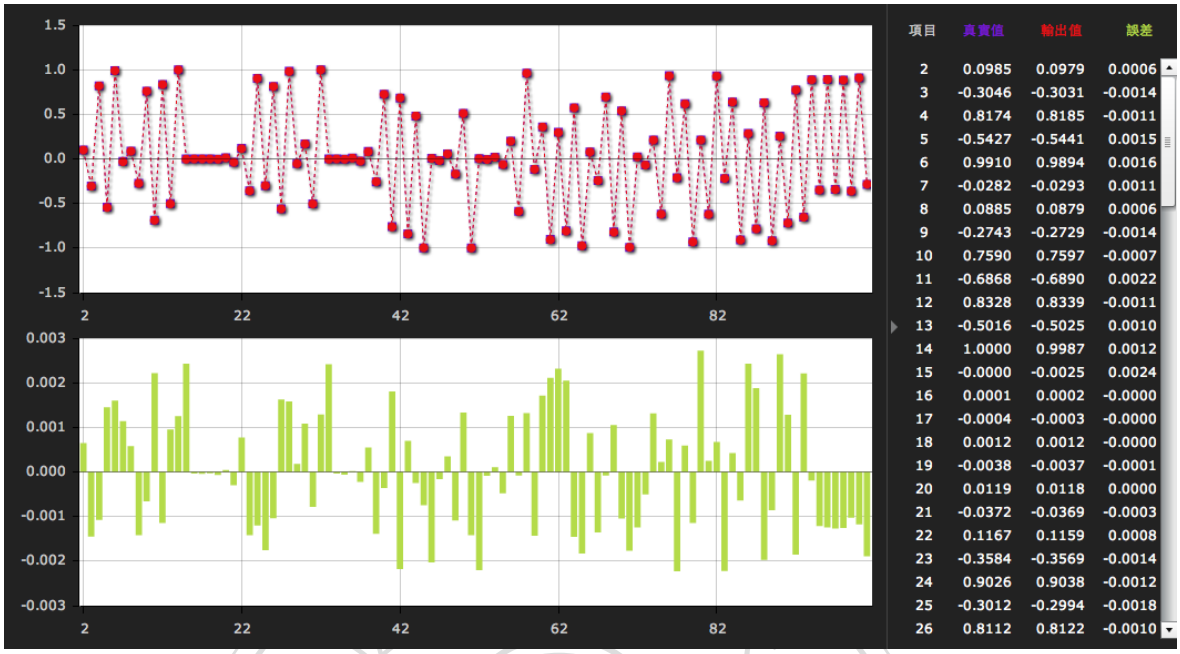


圖 5.15: 中心點個數: 10，實例操作圖

- 我們以建構 $4x(1-x)$ 爲例:

  (1) 設定 101 筆資料，前 100 筆爲網路輸入值，後 100 筆爲眞實值，以「當期」預測「下一期」的方式建構 RBFN。

  (2) 輻狀基底函數: 高斯函數(Gaussian function)。

  (3) 中心點選取方式: 垂直最小平方法。

  (4) 容忍誤差: 0.5，如圖 5.16。

- 我們同樣以建構 $4x(1-x)$ 爲例，但改變容忍誤差作爲比較:

  (1) 設定 101 筆資料，前 100 筆爲網路輸入值，後 100 筆爲眞實值，以「當期」預測「下一期」的方式建構 RBFN。

  (2) 輻狀基底函數: 選擇高斯函數(Gaussian function)。

  (3) 中心點選取方式: 垂直最小平方法。

  (4) 容忍誤差: 0.1，如圖 5.17。

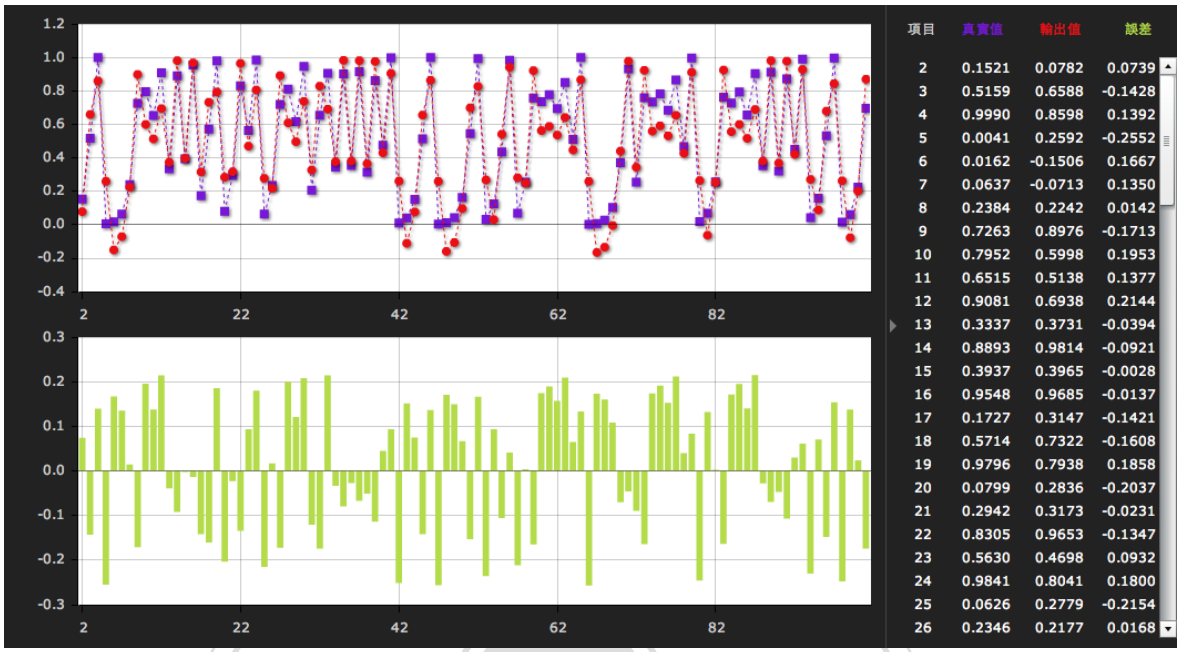


圖 5.16: 容忍誤差：0.5，實例操作圖

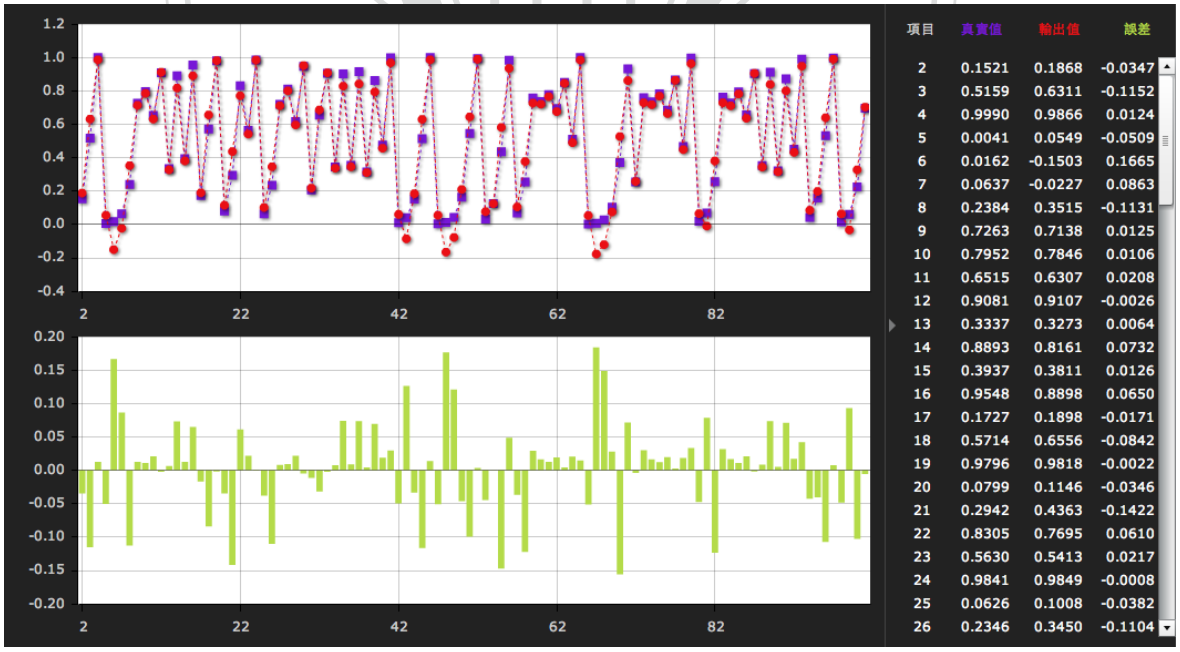


圖 5.17: 容忍誤差：0.1，實例操作圖

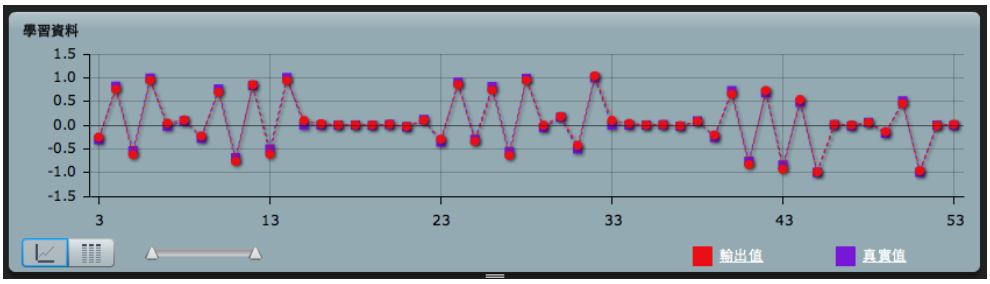


圖 5.18: 學習資料圖

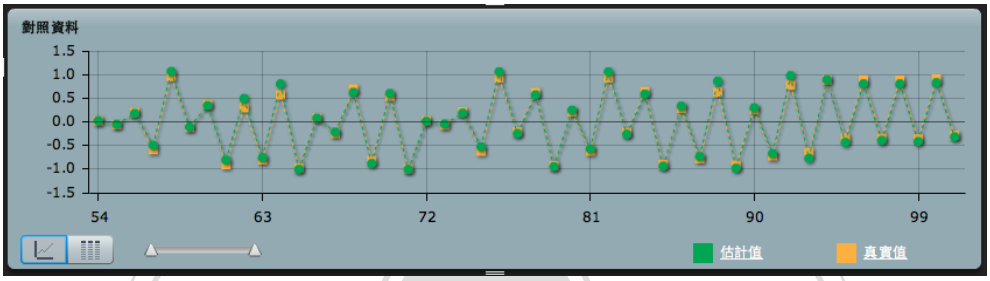


圖 5.19: 對照資料圖

5.5 預測實例

- 我們以預測 $sin(-\pi x)$ 爲例:

  (1) 設定 51 筆學習資料 (如圖 5.18)，50 筆對照資料 (如圖 5.19)。

  (2) 輻狀基底函數: 高斯函數(Gaussian function)。

  (3) 中心點選取方式: 隨機選取法平均選取。

  (4) 中心點個數: 8 個中心點。

  (5) 以「第一期」預測「第三期」的方式進行預測，如圖 5.20。

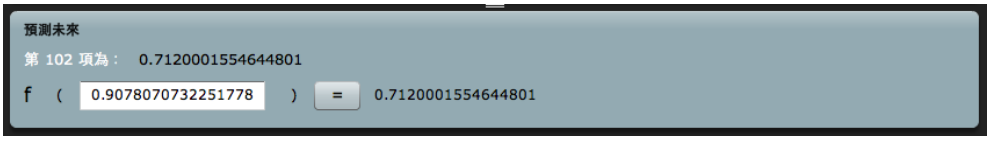


圖 5.20: 預測結果圖

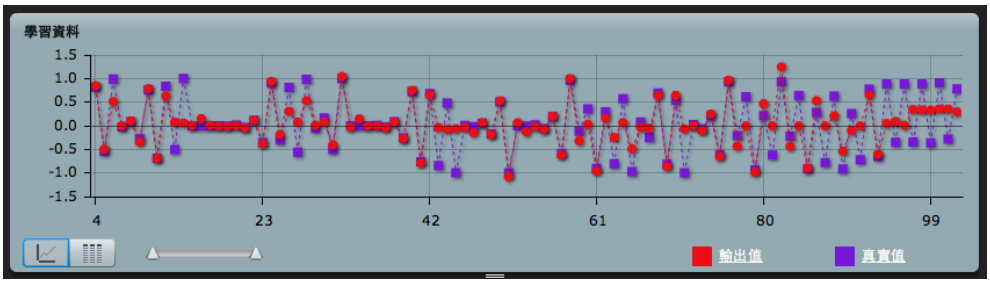


圖 5.21: 學習資料圖

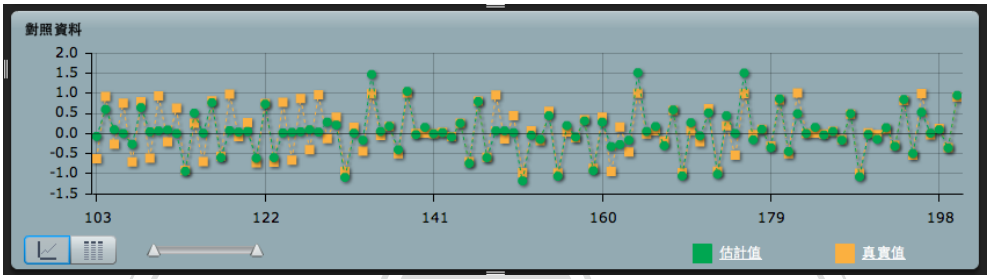


圖 5.22: 對照資料圖

- 我們以預測 $sin(-\pi x)$ 爲例:

  (1) 設定 100 筆學習資料 (如圖 5.21)，100 筆對照資料 (如圖 5.22)。

  (2) 輻狀基底函數: 高斯函數(Gaussian function)。

  (3) 中心點選取方式: 隨機選取法平均選取。

  (4) 中心點個數: 15 個中心點。

  (5) 以「第一期」、「第二期」預測「第四期」的方式進行預測，如圖 5.23。

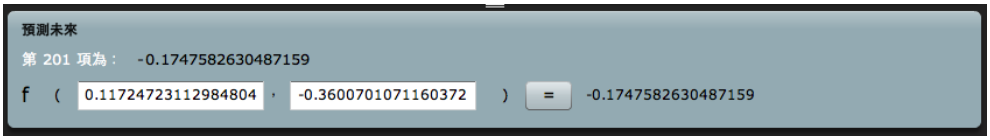


圖 5.23: 預測結果圖

- 我們以預測 $4x(1-x)$ 爲例:

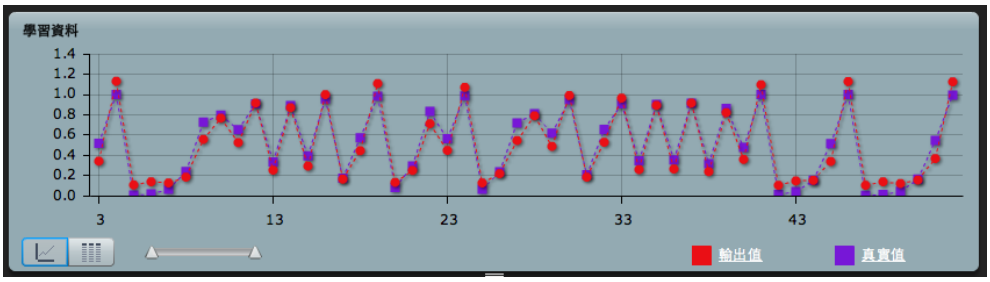


圖 5.24: 學習資料圖

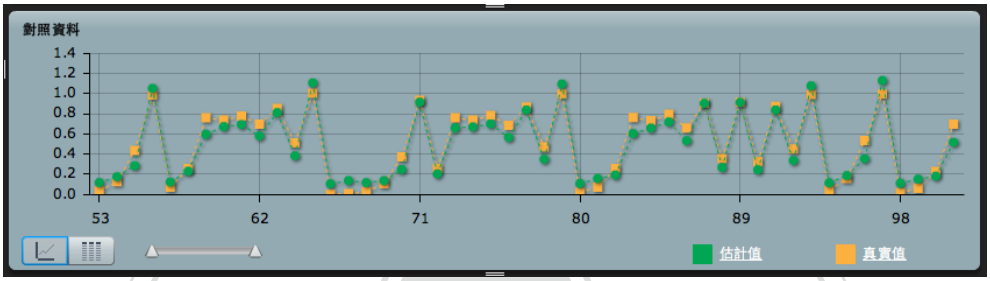


圖 5.25: 對照資料圖

(1) 設定 51 筆學習資料 (如圖 5.24)，50 筆對照資料 (如圖 5.25)。

(2) 輻狀基底函數：高斯函數(Gaussian function)。

(3) 中心點選取方式：垂直最小平方法。

(4) 容忍誤差：0.1。

(5) 以「第一期」、「第二期」預測「第三期」的方式進行預測，如圖 5.26。

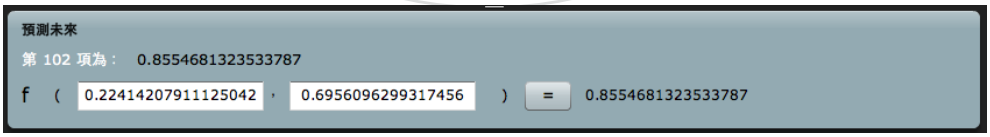


圖 5.26: 預測結果圖

● 我們以預測 $4x(1-x)$ 爲例:

(1) 設定 51 筆學習資料 (如圖 5.27)，50 筆對照資料 (如圖 5.28)。

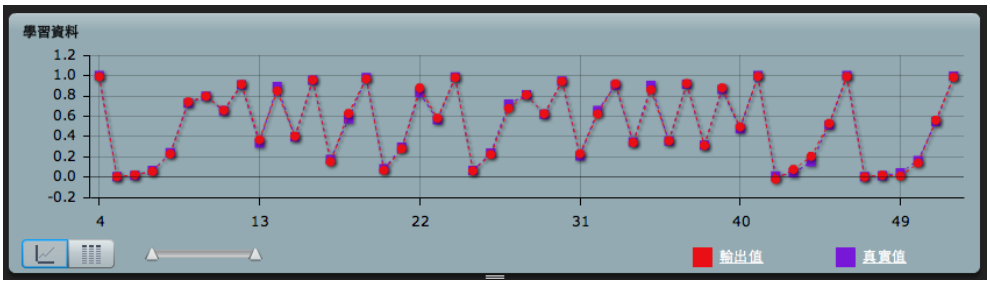


圖 5.27: 學習資料圖

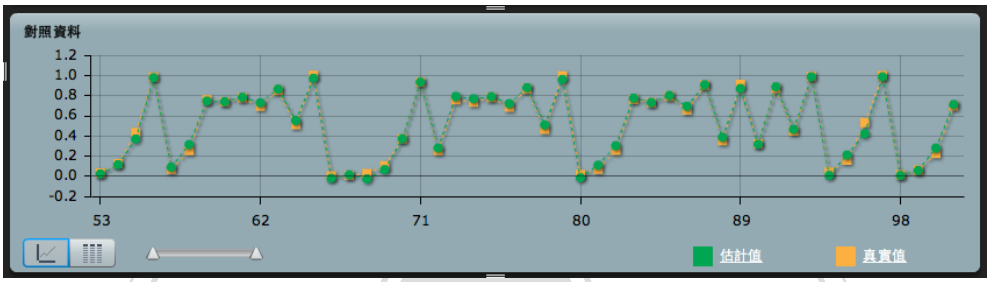


圖 5.28: 對照資料圖

(2) 輻狀基底函數：二次多變數函數(multiquadratic function)。

(3) 中心點選取方式：垂直最小平方法。

(4) 容忍誤差：0.0001。

(5) 以「第一期」、「第二期」、「第三期」預測「第四期」的方式進行預測，如圖 5.29。

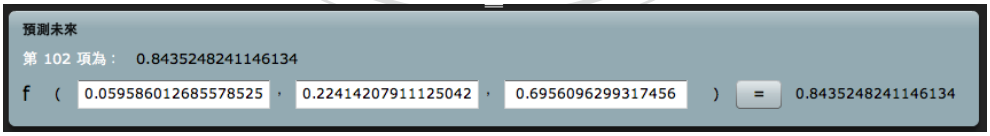


圖 5.29: 預測結果圖

5.6 結論

藉由 RBFN 互動式介面應用程式，我們可以很輕易的了解 RBFN 的網路架構，也

很容易的比較出不同的輻狀基底函數對預測結果的差異，進而選擇適合學習資料的的幅狀基底函數；也可以藉由動態改變中心點個數時，比較中心點對整個網路的影響：(1)網路運算速度，(2)預測值的精準度。當資料如果很龐大，OLS 中心點選取法比隨機選取中心點對預測結果的確有比較準確，如此對預測未來的目標值能有較佳參考。

附錄

A. 矩陣定義、常用矩陣屬性及工具

```
package com.clevr.matrixalgebra {



        /*
         *      Matrix class ported from Jama.
         *
         *      @langversion ActionScript 3.0
         *      @playerversion Flash 9.0
         *
         *      @author Matt Kane
         *      @since  04.06.2007
         */
        public class RealMatrix {

                /** Array for internal storage of elements.
                 */
                private var A:Array;

                /** Row and column dimensions.
                 */
                private var m:int, n:int;

                /* ------------------------
                   Constructors
                 * ------------------------ */

                /** Construct an m-by-n matrix of zeros.
                   We always show a vector as 1 column matrix which
                       means all the elements be a Array.
                   @param m    Number of rows.
                   @param n    Number of colums.
                */

                public function RealMatrix (mm:*, nn:int = 1, s:Number
                    = NaN) {
                        if (mm is Array) {
                                m = mm.length;
                                var tempB:Array;
                                if (mm[0] is Array){
                                        n = mm[0].length;
                                        for (i = 0; i < m; i++) {
                                                if (mm[i].length != n)
                                                    {
                                                        throw new Error
                                                            ("All rows
                                                            must have
                                                            the same
                                                            length.");
                                                }
                                        }
                                }else{
                                        n=1;
                                        for ( i=0; i<m; i++ ){
                                                tempB = new Array(1);
                                                tempB[0] = mm[i];
                                                mm[i] = tempB;
                                        }
                                }
                                this.A = mm;
                                return;
                        }
                        m = mm;
                        n = nn;
```

```
            A = new Array(m);
            var i:int;
            var j:int;
            for (i = 0; i < m; i++) {
                    A[i] = new Array(n);
            }
            if (!isNaN(s)) {
                    for (i = 0; i < m; i++) {
                            for (j = 0; j < n; j++) {
                                    A[i][j] = s;
                            }
                    }
            }
    }

    /** Construct a matrix from a copy of a 2-D array.
       @param arr    Two-dimensional array of doubles.
     */

    public static function constructWithCopy(arr:Array):
        RealMatrix {
            var mm:int = arr.length;
            var nn:int = arr[0].length;
            var X:RealMatrix = new RealMatrix(mm,nn);
            var C:Array = X.getArray();
            var j:int;
            for (var i:int = 0; i < mm; i++) {
                    if (arr[i].length != nn) {
                            throw new Error ("All rows must
                                have the same length.");
                    }
                    for (j = 0; j < nn; j++) {
                            C[i][j] = arr[i][j];
                            /*trace("C[" + i + "][" + j +
                                "]: " + C[i][j]);*/
                    }
            }
            return X;
    }

    /** Access the internal two-dimensional array.
       @return     Pointer to the two-dimensional array of
            matrix elements.
     */

    public function getArray ():Array {
            return A;
    }

    /** Copy the internal two-dimensional array.
       @return     Two-dimensional array copy of matrix
            elements.
     */

    public function getArrayCopy ():Array {
            var C:Array = new Array(m);
            var j:int;
            for (var i:int = 0; i < m; i++) {
                    C[i] = new Array(n);
                    for (j = 0; j < n; j++) {
                            C[i][j] = A[i][j];
                    }
            }
            return C;
    }

    /** Make a one-dimensional column packed copy of the
        internal array.
       @return     Matrix elements packed in a one-
            dimensional array by columns.
     */

    public function getColumnPackedCopy ():Array {
            var vals:Array = new Array(m*n);
            var j:int;
            for (var i:int = 0; i < m; i++) {
```

```
                for (j = 0; j < n; j++) {
                        vals[i+j*m] = A[i][j];
                }
        }
        return vals;
}

/** Make a n-by-1 vector be the kth row vector.
   @return     the kth row of this matrix.
 */

public function getRowVector(k:int):RealMatrix{
        return new RealMatrix(A[k]);
}

/** Make a one-dimensional row packed copy of the
    internal array.
   @return     Matrix elements packed in a one-
       dimensional array by rows.
 */

public function getRowPackedCopy ():Array {
        var vals:Array = new Array(m*n);
        var j:int;
        for (var i:int = 0; i < m; i++) {
                for (j = 0; j < n; j++) {
                        vals[i*n+j] = A[i][j];
                }
        }
        return vals;
}

/** Get row dimension.
   @return     m, the number of rows.
 */

public function getRowDimension ():int {
        return m;
}

/** Get column dimension.
   @return     n, the number of columns.
 */

public function getColumnDimension ():int {
        return n;
}

/** Get a single element.
   @param i    Row index.
   @param j    Column index.
   @return     A(i,j)
   @exception  ArrayIndexOutOfBoundsException
 */

public function get (i:int, j:int):Number {
        return A[i][j];
}

/** Get a submatrix.
   @param i0   Initial row index
   @param i1   Final row index
   @param j0   Initial column index
   @param j1   Final column index
   @return     A(i0:i1,j0:j1)
 */

public function getMatrix (i0:int, i1:int, j0:int, j1:
    int):RealMatrix {
        var X:RealMatrix = new RealMatrix(i1-i0+1,j1-j0
            +1);
        var B:Array = X.getArray();
        for (var i:int = i0; i <= i1; i++) {
                for (var j:int = j0; j <= j1; j++) {
                        B[i-i0][j-j0] = A[i][j];
                }
        }
        return X;
```

```
}


/** Set a single element.
   @param i    Row index.
   @param j    Column index.
   @param s    A(i,j).
 */

public function set (i:int, j:int, s:Number):void {
        A[i][j] = s;
}


/** Matrix transpose.
   @return    A
 */

public function transpose ():RealMatrix {
        var X:RealMatrix = new RealMatrix(n,m);
        var C:Array = X.getArray();
        var j:int;
        for (var i:int = 0; i < m; i++) {
                for (j = 0; j < n; j++) {
                        C[j][i] = A[i][j];
                }
        }
        return X;
}
/** get maximun of k column.
 *  initial k is 0.
   @return    maximum value.
 */

public function getColumnMaximun(k:int=0):Number {
        var max:Number
        if (n==1){
                if (k!=0){
                        throw new Error("only 1 column.
                            ");
                        return max;
                }
                max     =Number(A[0]);
                for (var i:int=1;i<m;i++){
                        max = Math.max(max,A[i]);
                }
        }else{
                max     =Number(A[0][k]);
                for (i=1;i<m;i++){
                        max = Math.max(max,A[i][k]);
                }
        }
        return max;
}
/** get minmun of k column.
 *  initial k is 0.
   @return    minmun value.
 */

public function getColumnMinimun(k:int=0):Number {
        var min:Number
        if (n==1){
                if (k!=0){
                        throw new Error("only 1 column.
                            ");
                        return min;
                }
                min     =Number(A[0]);
                for (var i:int=1;i<m;i++){
                        min = Math.min(min,A[i]);
                }
        }else{
                min     =Number(A[0][k]);
                for (i=1;i<m;i++){
                        min = Math.min(min,A[i][k]);
                }
        }
        return min;
```

```
}

/** One norm
   @return    maximum column sum.
 */

public function norm1 ():Number {
        var f:Number = 0;
        var i:int;
        var s:Number;
        for (var j:int = 0; j < n; j++) {
                s = 0;
                for (i = 0; i < m; i++) {
                        s += Math.abs(A[i][j]);
                }
                f = Math.max(f,s);
        }
        return f;
}

/** Two norm
   @return    maximum singular value.
 */

public function norm2 ():Number {
        return (new SingularValueDecomposition(this).
            norm2());
}

/** Infinity norm
   @return    maximum row sum.
 */

public function normInf ():Number {
        var f:Number = 0;
        var j:int;
        var s:Number;
        for (var i:int = 0; i < m; i++) {
                s = 0;
                for (j = 0; j < n; j++) {
                        s += Math.abs(A[i][j]);
                }
                f = Math.max(f,s);
        }
        return f;
}

/** Frobenius norm
   @return    sqrt of sum of squares of all elements.
 */

public function normF ():Number {
        var f:Number = 0;
        var j:int;
        for (var i:int = 0; i < m; i++) {
                for (j = 0; j < n; j++) {
                        f = Maths.hypot(f,A[i][j]);
                }
        }
        return f;
}

/**  Unary minus
   @return    -A
 */

public function uminus ():RealMatrix {
        var X:RealMatrix = new RealMatrix(m,n);
        var C:Array = X.getArray();
        var j:int;
        for (var i:int = 0; i < m; i++) {
                for (j = 0; j < n; j++) {
                        C[i][j] = -A[i][j];
                }
        }
        return X;
}
```

```
/** C = A + B
   @param B     another matrix
   @return      A + B
 */

public function plus (B:RealMatrix):RealMatrix {
        checkMatrixDimensions(B);
        var X:RealMatrix = new RealMatrix(m,n);
        var C:Array = X.getArray();
        var j:int;
        for (var i:int = 0; i < m; i++) {
                for (j = 0; j < n; j++) {
                        C[i][j] = A[i][j] + B.A[i][j];
                }
        }
        return X;
}

/** A = A + B
   @param B     another matrix
   @return      A + B
 */

public function plusEquals (B:RealMatrix):RealMatrix {
        checkMatrixDimensions(B);
        var j:int;
        for (var i:int = 0; i < m; i++) {
                for (j = 0; j < n; j++) {
                        A[i][j] = A[i][j] + B.A[i][j];
                }
        }
        return this;
}

/** C = A - B
   @param B     another matrix
   @return      A - B
 */

public function minus (B:RealMatrix):RealMatrix {
        checkMatrixDimensions(B);
        var X:RealMatrix = new RealMatrix(m,n);
        var C:Array = X.getArray();
        var j:int;
        for (var i:int = 0; i < m; i++) {
                for (j = 0; j < n; j++) {
                        C[i][j] = A[i][j] - B.A[i][j];
                }
        }
        return X;
}

/** A = A - B
   @param B     another matrix
   @return      A - B
 */

public function minusEquals (B:RealMatrix):RealMatrix {
        checkMatrixDimensions(B);
        var j:int;
        for (var i:int = 0; i < m; i++) {
                for (j = 0; j < n; j++) {
                        A[i][j] = A[i][j] - B.A[i][j];
                }
        }
        return this;
}

/** Element-by-element multiplication, C = A.*B
   @param B     another matrix
   @return      A.*B
 */

public function arrayTimes (B:RealMatrix):RealMatrix {
        checkMatrixDimensions(B);
        var X:RealMatrix = new RealMatrix(m,n);
        var C:Array = X.getArray();
        var j:int;
```

```
            for (var i:int = 0; i < m; i++) {
                    for (j = 0; j < n; j++) {
                            C[i][j] = A[i][j] * B.A[i][j];
                    }
            }
            return X;
    }

    /** Element-by-element multiplication in place, A = A.*
        B
       @param B     another matrix
       @return      A.*B
     */

    public function arrayTimesEquals (B:RealMatrix):
        RealMatrix {
            checkMatrixDimensions(B);
            var j:int;
            for (var i:int = 0; i < m; i++) {
                    for (j = 0; j < n; j++) {
                            A[i][j] = A[i][j] * B.A[i][j];
                    }
            }
            return this;
    }

    /** Element-by-element right division, C = A./B
       @param B     another matrix
       @return      A./B
     */

    public function arrayRightDivide (B:RealMatrix):
        RealMatrix {
            checkMatrixDimensions(B);
            var X:RealMatrix = new RealMatrix(m,n);
            var j:int;
            var C:Array = X.getArray();
            for (var i:int = 0; i < m; i++) {
                    for (j = 0; j < n; j++) {
                            C[i][j] = A[i][j] / B.A[i][j];
                    }
            }
            return X;
    }

    /** Element-by-element right division in place, A = A./
        B
       @param B     another matrix
       @return      A./B
     */

    public function arrayRightDivideEquals (B:RealMatrix):
        RealMatrix {
            checkMatrixDimensions(B);
            var j:int;
            for (var i:int = 0; i < m; i++) {
                    for (j = 0; j < n; j++) {
                            A[i][j] = A[i][j] / B.A[i][j];
                    }
            }
            return this;
    }

    /** Element-by-element left division, C = A.\B
       @param B     another matrix
       @return      A.\B
     */

    public function arrayLeftDivide (B:RealMatrix):
        RealMatrix {
            checkMatrixDimensions(B);
            var X:RealMatrix = new RealMatrix(m,n);
            var C:Array = X.getArray();
            var j:int;
            for (var i:int = 0; i < m; i++) {
                    for (j = 0; j < n; j++) {
                            C[i][j] = B.A[i][j] / A[i][j];
                    }
```

```
        }
        return X;
}

/** Element-by-element left division in place, A = A.\B
   @param B     another matrix
   @return      A.\B
 */

public function arrayLeftDivideEquals (B:RealMatrix):
    RealMatrix {
        checkMatrixDimensions(B);
        var j:int;
        for (var i:int = 0; i < m; i++) {
                for (j = 0; j < n; j++) {
                        A[i][j] = B.A[i][j] / A[i][j];
                }
        }
        return this;
}

/** Multiply a matrix by a scalar, C = s*A
   @param s     scalar
   @return      s*A
 */

public function times (s:*):RealMatrix {
        if (s is RealMatrix) {
                return timesMatrix(s);
        }
        var X:RealMatrix = new RealMatrix(m,n);
        var C:Array = X.getArray();
        var j:int;
        for (var i:int = 0; i < m; i++) {
                for (j = 0; j < n; j++) {
                        C[i][j] = s*A[i][j];
                }
        }
        return X;
}

/** Multiply a matrix by a scalar in place, A = s*A
   @param s     scalar
   @return      replace A by s*A
 */

public function timesEquals (s:Number):RealMatrix {
        var j:int;
        for (var i:int = 0; i < m; i++) {
                for (j = 0; j < n; j++) {
                        A[i][j] = s*A[i][j];
                }
        }
        return this;
}

/** Linear algebraic innear Product, A . B
   @param B     another matrix
   @return      Matrix innear product, A . B
 */

public function innearProduct (B:RealMatrix):Number{
        if ( n!=1 && m!=1 ){
                throw new Error("Please use matrix
                    product, this is use to do inner
                    product.");
        }
        if ( B.n!=1 && B.m!=1 ){
                throw new Error("Please use matrix
                    product, this is use to do inner
                    product.");
        }
        return this.transpose().timesMatrix(B).getArray
            ()[0][0];
}

/** Linear algebraic matrix multiplication, A * B
   @param B     another matrix
```

```
   @return     Matrix product, A * B
 */

public function timesMatrix (B:RealMatrix):RealMatrix {
        if (B.m != n) {
                throw new Error("Matrix inner
                    dimensions must agree.");
        }
        var X:RealMatrix = new RealMatrix(m,B.n);
        var C:Array = X.getArray();
        var Bcolj:Array = new Array(n);
        var Arowi:Array;
        var s:Number;
        var k:int;
        var i:int;
        for (var j:int = 0; j < B.n; j++) {
                for (k = 0; k < n; k++) {
                        Bcolj[k] = B.A[k][j];
                }
                for (i = 0; i < m; i++) {
                        Arowi = A[i];
                        s = 0;
                        for (k = 0; k < n; k++) {
                                s += Arowi[k]*Bcolj[k];
                        }
                        C[i][j] = s;
                }
        }
        return X;
}

/** LU Decomposition
   @return     LUDecomposition
   @see LUDecomposition
 */

/*public function lu ():LUDecomposition {
   return new LUDecomposition(this);
 }*/

/** QR Decomposition
   @return     QRDecomposition
   @see QRDecomposition
 */

public function qr ():QRDecomposition {
        return new QRDecomposition(this);
}

/** Cholesky Decomposition
   @return     CholeskyDecomposition
   @see CholeskyDecomposition
 */

/*public CholeskyDecomposition chol () {
   return new CholeskyDecomposition(this);
 }*/

/** Singular Value Decomposition
   @return     SingularValueDecomposition
   @see SingularValueDecomposition
 */

public function svd ():SingularValueDecomposition {
        return new SingularValueDecomposition(this);
}

/** Eigenvalue Decomposition
   @return     EigenvalueDecomposition
   @see EigenvalueDecomposition
 */

/*public EigenvalueDecomposition eig () {
   return new EigenvalueDecomposition(this);
 }*/

/** Solve A*X = B
   @param B    right hand side
```

```
   @return      solution if A is square, least squares
       solution otherwise
 */

public function solve (B:RealMatrix):RealMatrix {
        /*return (m == n ? (new LUDecomposition(this)).
            solve(B) :*/
        return (new QRDecomposition(this)).solve(B);
}

/** Solve X*A = B, which is also A'*X' = B'
   @param B     right hand side
   @return      solution if A is square, least squares
       solution otherwise.
 */

public function solveTranspose (B:RealMatrix):
    RealMatrix {
        return transpose().solve(B.transpose());
}

/** Matrix inverse or pseudoinverse
   @return      inverse(A) if A is square, pseudoinverse
       otherwise.
 */

public function inverse ():RealMatrix {
        return solve(identity(m,m));
}

/** Matrix determinant
   @return      determinant
 */

/*public function det ():Number {
   return new LUDecomposition(this).det();
 }*/

/** Matrix rank
   @return      effective numerical rank, obtained from
       SVD.
 */

public function rank ():int {
        return new SingularValueDecomposition(this).
            rank();
}

/** Matrix condition (2 norm)
   @return      ratio of largest to smallest singular
       value.
 */

public function cond ():Number {
        return new SingularValueDecomposition(this).
            cond();
}

/** Matrix trace.
   @return      sum of the diagonal elements.
 */

public function trace ():Number {
        var t:Number = 0;
        for (var i:int = 0; i < Math.min(m,n); i++) {
                t += A[i][i];
        }
        return t;
}

/** Generate matrix with random elements
   @param m     Number of rows.
   @param n     Number of colums.
   @return      An m-by-n matrix with uniformly
       distributed random elements.
 */
```

```
                public static function random (m:int, n:int):RealMatrix
                    {
                        var A:RealMatrix = new RealMatrix(m,n);
                        var X:Array = A.getArray();
                        var j:int;
                        for (var i:int = 0; i < m; i++) {
                                for (j = 0; j < n; j++) {
                                        X[i][j] = Math.random();
                                }
                        }
                        return A;
                }

                /** Generate identity matrix
                   @param m    Number of rows.
                   @param n    Number of colums.
                   @return     An m-by-n matrix with ones on the
                       diagonal and zeros elsewhere.
                 */

                public static function identity (m:int, n:int):
                    RealMatrix {
                        var A:RealMatrix = new RealMatrix(m,n);
                        var X:Array = A.getArray();
                        var j:int;
                        for (var i:int = 0; i < m; i++) {
                                for (j = 0; j < n; j++) {
                                        X[i][j] = (i == j ? 1.0 : 0.0);
                                }
                        }
                        return A;
                }


                /** Check if size(A) == size(B) **/

                private function checkMatrixDimensions (B:RealMatrix):
                    void {
                        if (B.m != m || B.n != n) {
                                throw new Error("Matrix dimensions must
                                    agree.");
                        }
                    }

        }
}
```

B. 利用 Householder 矩陣來求 QR 分解及相關的應用

```
package com.clevr.matrixalgebra {

        /** QR Decomposition.

           For an m-by-n matrix A with m >= n, the QR decomposition is
               an m-by-n
           orthogonal matrix Q and an n-by-n upper triangular matrix R
               so that
           A = Q*R.
           The QR decompostion always exists, even if the matrix does
               not have
           full rank, so the constructor will never fail.  The primary
               use of the
           QR decomposition is in the least squares solution of
               nonsquare systems
           of simultaneous linear equations.  This will fail if
               isFullRank()
```

```
 */
public class QRDecomposition {

        /* ------------------------
           Class variables
         * ------------------------ */

        /** Array for internal storage of decomposition.
         */
        private var QR:Array;

        /** Row and column dimensions.
         */
        private var m:int, n:int;

        /** Array for internal storage of diagonal of R.
         */
        private var Rdiag:Array;

        /* ------------------------
           Constructor
         * ------------------------ */

        /** QR Decomposition, computed by Householder
            reflections.
            @param A    Rectangular matrix
         */

        public function QRDecomposition (A:RealMatrix) {
                // Initialize.
                QR = A.getArrayCopy();
                m = A.getRowDimension();
                n = A.getColumnDimension();
                Rdiag = new Array(n);

                // Main loop.
                var i:int;
                var j:int;
                var nrm:Number;
                var s:Number;
                for (var k:int = 0; k < n; k++) {
                        // Compute 2-norm of k-th column
                            without under/overflow.
                        nrm = 0;
                        for (i = k; i < m; i++) {
                                nrm = Maths.hypot(nrm,QR[i][k])
                                    ;
                        }

                        if (nrm != 0.0) {
                                // Form k-th Householder vector
                                    .
                                if (QR[k][k] < 0) {
                                        nrm = -nrm;
                                }
                                for (i = k; i < m; i++) {
                                        QR[i][k] /= nrm;
                                }
                                QR[k][k] += 1.0;

                                // Apply transformation to
                                    remaining columns.
                                for (j = k+1; j < n; j++) {
                                        s = 0.0;
                                        for (i = k; i < m; i++)
                                             {
                                                s += QR[i][k]*
                                                    QR[i][j];
                                        }
                                        s = -s/QR[k][k];
                                        for (i = k; i < m; i++)
                                             {
                                                QR[i][j] += s*
                                                    QR[i][k];
                                        }
                                }
                        }
                }
```

```
                Rdiag[k] = -nrm;
            }
        }

        /* ------------------------
           Public Methods
         * ------------------------ */

        /** Is the matrix full rank?
           @return     true if R, and hence A, has full rank.
         */

        public function isFullRank ():Boolean {
                for (var j:int = 0; j < n; j++) {
                        if (Rdiag[j] == 0)
                                return false;
                }
                return true;
        }

        /** Return the Householder vectors
           @return     Lower trapezoidal matrix whose columns
                define the reflections
         */

        public function getH ():RealMatrix {
                var X:RealMatrix = new RealMatrix(m,n);
                var H:Array = X.getArray();
                var j:int;
                for (var i:int = 0; i < m; i++) {
                        for (j = 0; j < n; j++) {
                                if (i >= j) {
                                        H[i][j] = QR[i][j];
                                } else {
                                        H[i][j] = 0.0;
                                }
                        }
                }
                return X;
        }

        /** Return the upper triangular factor
           @return     R
         */

        public function getR ():RealMatrix {
                var X:RealMatrix = new RealMatrix(n,n);
                var R:Array = X.getArray();
                var j:int
                for (var i:int = 0; i < n; i++) {
                        for (j = 0; j < n; j++) {
                                if (i < j) {
                                        R[i][j] = QR[i][j];
                                } else if (i == j) {
                                        R[i][j] = Rdiag[i];
                                } else {
                                        R[i][j] = 0.0;
                                }
                        }
                }
                return X;
        }

        /** Generate and return the (economy-sized) orthogonal
            factor
           @return     Q
         */

        public function getQ ():RealMatrix {
                var X:RealMatrix = new RealMatrix(m,n);
                var Q:Array = X.getArray();
                var i:int;
                var k:int;
                var j:int;
                var s:Number;
                for (k = n-1; k >= 0; k--) {
                        for (i = 0; i < m; i++) {
                                Q[i][k] = 0.0;
```

```
                        }
                        Q[k][k] = 1.0;
                        for (j = k; j < n; j++) {
                                if (QR[k][k] != 0) {
                                        s = 0.0;
                                        for (i = k; i < m; i++)
                                             {
                                                s += QR[i][k]*Q
                                                    [i][j];
                                        }
                                        s = -s/QR[k][k];
                                        for (i = k; i < m; i++)
                                             {
                                                Q[i][j] += s*QR
                                                    [i][k];
                                        }
                                }
                        }
                }
                return X;
        }

        /** Least squares solution of A*X = B
           @param B    A Matrix with as many rows as A and any
               number of columns.
           @return     X that minimizes the two norm of Q*R*X-B
               .
         */

        public function solve (B:RealMatrix):RealMatrix {
                if (B.getRowDimension() != m) {
                        throw new Error("Matrix row dimensions
                            must agree.");
                }
                if (!this.isFullRank()) {
                        throw new Error("Matrix is rank
                            deficient.");
                }

                // Copy right hand side
                var nx:int = B.getColumnDimension();
                var X:Array = B.getArrayCopy();

                var k:int;
                var j:int;
                var i:int;
                var s:Number;

                // Compute Y = transpose(Q)*B
                for (k = 0; k < n; k++) {
                        for (j = 0; j < nx; j++) {
                                s = 0;
                                for (i = k; i < m; i++) {
                                        s += QR[i][k]*X[i][j];
                                }
                                s = -s/QR[k][k];
                                for (i = k; i < m; i++) {
                                        X[i][j] += s*QR[i][k];
                                }
                        }
                }
                // Solve R*X = Y;
                for (k = n-1; k >= 0; k--) {
                        for (j = 0; j < nx; j++) {
                                X[k][j] /= Rdiag[k];
                        }
                        for (i = 0; i < k; i++) {
                                for (j = 0; j < nx; j++) {
                                        X[i][j] -= X[k][j]*QR[i
                                            ][k];
                                }
                        }
                }
                return (new RealMatrix(X,n,nx).getMatrix(0,n
                    -1,0,nx-1));
        }
}
```

```
}
```

C. RBFN 相關工具

```
package com.RBFPackage
{
        import com.clevr.matrixalgebra.Maths;
        import com.clevr.matrixalgebra.RealMatrix;


        public class RBF
        {
                public function RBF()
                {
                }
                /**
                 * the norm of a - b
                 */
                private static function norm(a:*,b:*):Number{
                        if ( a == b ){
                                return 0.0;
                        }
                        var s:Number;
                        if ( a is RealMatrix && b is RealMatrix ){
                                var c:RealMatrix = (a as RealMatrix);
                                var d:RealMatrix = (b as RealMatrix);
                                var e:RealMatrix = c.minus(d);
                                s = e.norm2();
                        }else{
                                s = Math.abs( Number(a)-Number(b) );
                        }
                        return s;
                }
                /**
                 * Guassian function
                 */
                public static function Gaussian(a:*,b:*,sigma:Number):
                    Number{
                        var s:Number = norm(a,b) / sigma;
                        return Math.exp( -s*s );
                }
                /**
                 * Multi-quadratic function
                 */
                public static function MultiQuadratic(a:*,b:*,sigma:
                    Number):Number{
                        var c:Number = norm(a,b);
                        return Maths.hypot(c,sigma);
                }

                /**
                 * Inverse multi-quadratic function
                 */
                public static function InverseMultiQuadratic(a:*,b:*,
                    sigma:Number):Number{
                        var c:Number = norm(a,b);
                        return 1/Maths.hypot(c,sigma);
                }

                /**
                 * Thin-plate-spline function
                 */
                public static function ThinPlateSpline(a:*,b:*):Number{
                        var c:Number = norm(a,b);
                        return c*c*( Math.log(c) );
                }
```

```
public static function Error(learningData:RealMatrix,
    output:RealMatrix):RealMatrix{
        /*  error */
        var n:int = learningData.getRowDimension();
        var tempR:RealMatrix = new RealMatrix(n);
        var A:Array = tempR.getArray();
        var s:Number;
        for (var i:int=0;i<n;i++){
                s = learningData.getArray()[i][0]-
                    output.getArray()[i][0];
                A[i][0] = s;
        }
        return tempR;
}

private static function getOptimalSigma(centers:
    RealMatrix):Number{
        var n:int = centers.getRowDimension();
        var l:int = centers.getColumnDimension();
        var max:Number=0.0;
        var min:Number=0.0;
        if ( l==1 ){
                max = centers.getColumnMaximun();
                min = centers.getColumnMinimun();
        }else{
                var k:Number;
                var tempA:RealMatrix;
                var tempB:RealMatrix
                for (var i:int=0;i<n;i++){
                        tempA = centers.getRowVector(i)
                            ;
                        for (var j:int=0;j<n;j++){
                                tempB = centers.
                                    getRowVector(j);
                                k = tempA.minus(tempB).
                                    norm2();
                                max = Math.max(max,k);
                                min = Math.min(min,k);
                        }
                }
        }
        return Math.abs(max - min)/Math.sqrt(n);
}
/**
* forecast
* **/
public static function forecast(XInput:RealMatrix,
    XCenter:RealMatrix,funName:String,XWeights:
    RealMatrix):RealMatrix{
        var m:int = XInput.getRowDimension();
        var n:int = XCenter.getRowDimension();
        var k:int = XCenter.getColumnDimension();
        var A:Array = XInput.getArrayCopy();
        var C:Array = XCenter.getArrayCopy();
        var phi:RealMatrix = new RealMatrix(m,n);
        var H:Array = phi.getArray();
        var i:int;
        var j:int;
        var sigma:Number = getOptimalSigma(XCenter);
        var tempC:RealMatrix;
        var tempA:RealMatrix;
        if (funName=="Guassian"){
                for ( i=0; i<m; i++ ){
                        tempA = new RealMatrix(A[i]);
                        for ( j=0; j<n; j++ ){
                                if (k!=1){
                                        tempC = new
                                            RealMatrix(
                                            C[j]);
                                        H[i][j]=
                                            Gaussian(
                                            tempA,tempC
                                            ,sigma );
                                }else{
                                        H[i][j]=
                                            Gaussian(
                                            Number(A[i
                                            ]),Number(C
```

```
                                                [j]),sigma
                                                );
                                        }
                                }
                        }
                }else if (funName == "multi-quadratic"){
                        for ( i=0; i<m; i++ ){
                                tempA = new RealMatrix(A[i]);
                                for ( j=0; j<n; j++ ){
                                        if (k!=1){
                                                tempC = new
                                                    RealMatrix(
                                                    C[j]);
                                                H[i][j]=
                                                    MultiQuadratic
                                                    ( tempA,
                                                    tempC,sigma
                                                     );
                                        }else{
                                                H[i][j]=
                                                    MultiQuadratic
                                                    ( Number(A[
                                                    i]),Number(
                                                    C[j]),sigma
                                                     );
                                        }
                                }
                        }
                }else if (funName == "inverse multi-quadratic")
                    {
                        for ( i=0; i<m; i++ ){
                                tempA = new RealMatrix(A[i]);
                                for ( j=0; j<n; j++ ){
                                        if (k!=1){
                                                tempC = new
                                                    RealMatrix(
                                                    C[j]);
                                                H[i][j]=
                                                    InverseMultiQuadratic
                                                    ( tempA,
                                                    tempC,sigma
                                                     );
                                        }else{
                                                H[i][j]=
                                                    InverseMultiQuadratic
                                                    ( Number(A[
                                                    i]),Number(
                                                    C[j]),sigma
                                                    );
                                        }
                                }
                        }
                }else if (funName == "thin-plate-spline"){
                        for ( i=0; i<m; i++ ){
                                tempA = new RealMatrix(A[i]);
                                for ( j=0; j<n; j++ ){
                                        if (k!=1){
                                                tempC = new
                                                    RealMatrix(
                                                    C[j]);
                                                H[i][j]=
                                                    ThinPlateSpline
                                                    ( tempA,
                                                    tempC );
                                        }else{
                                                H[i][j]=
                                                    ThinPlateSpline
                                                    ( Number(A[
                                                    i]),Number(
                                                    C[j]) );
                                        }
                                }
                        }
                }

                return phi.timesMatrix(XWeights);
        }
```

```
        }
}
```

D. RBFN 隨機選取法

```
package com.RBFPackage
{
        import com.clevr.matrixalgebra.RealMatrix;



        public class RBFN_Random
        {

                private var phi:RealMatrix;

                private var input:RealMatrix;

                private var centers:RealMatrix;

                private var sigma:Number;

                private var numCenters:int;

                private var weights:RealMatrix;

                public function RBFN_Random(learningData:RealMatrix,
                    type:int=0,centerNum:int=1,funName:String="Guassian
                    ")
                {
                        input = learningData;
                        numCenters = centerNum;
                        centers = centerMethod(type);
                        creatRBFMatrix(funName);

                }

                private function centerMethod(type:int):RealMatrix{
                        var tempCenter:RealMatrix;
                        var c:Array;
                        var n:int = input.getColumnDimension();
                        var max:Number;
                        var min:Number;
                        var r:Number;
                        if ( n == 1){
                                tempCenter = new RealMatrix(numCenters)
                                    ;
                                c = tempCenter.getArray();
                                max = input.getColumnMaximun();
                                min = input.getColumnMinimun();
                                if (type==0){
                                        r = (max-min)/(numCenters+1);
                                        for ( var i:int=0; i<numCenters
                                            ; i++ ){
                                                c[i][0] = min+r*(i+1);
                                        }
                                }else if (type==1){
                                        for ( i=0; i<numCenters; i++ ){
                                                r = max-min;
                                                c[i][0] = min+r*Math.
                                                    random();
                                        }
                                }
                        }else{
                                tempCenter = new RealMatrix(numCenters,
                                    n);
```

```
                    c = tempCenter.getArray();
                    for (i=0;i<n;i++){
                            max = input.getColumnMaximun(i)
                                ;
                            min = input.getColumnMinimun(i)
                                ;
                            if (type==0){
                                    r = (max-min)/(
                                        numCenters+1);
                                    for ( var j:int=0; j<
                                        numCenters; j++ ){
                                            c[j][i] = min+r
                                                *(j+1);

                                    }
                            }else if (type==1){
                                    r = max-min;
                                    for ( j=0; j<numCenters
                                        ; j++ ){
                                            c[j][i] = min+r
                                                *Math.
                                                random();
                                    }
                            }
                    }
            }
            return tempCenter;
    }

    /**
     * get centers
     * **/
    public function getCenters():RealMatrix{
            return centers;
    }

    private function getOptimalSigma(centers:RealMatrix):
        Number{
            var n:int = centers.getRowDimension();
            var l:int = centers.getColumnDimension();
            var max:Number=0.0;
            var min:Number=0.0;
            if ( l==1 ){
                    max = centers.getColumnMaximun();
                    min = centers.getColumnMinimun();
            }else{
                    var k:Number;
                    var tempA:RealMatrix;
                    var tempB:RealMatrix
                    for (var i:int=0;i<n;i++){
                            tempA = centers.getRowVector(i)
                                ;
                            for (var j:int=0;j<n;j++){
                                    tempB = centers.
                                        getRowVector(j);
                                    k = tempA.minus(tempB).
                                        norm2();
                                    max = Math.max(max,k);
                                    min = Math.min(min,k);
                            }
                    }
            }
            return Math.abs(max - min)/Math.sqrt(n);
    }

    private function creatRBFMatrix(funName:String="
        Guassian"):void{

            var m:int = input.getRowDimension();
            var n:int = centers.getRowDimension();
            var k:int = centers.getColumnDimension();
            var A:Array = input.getArrayCopy();
            var C:Array = centers.getArrayCopy();
            phi = new RealMatrix(m,n);
            var H:Array = phi.getArray();
            var i:int;
            var j:int;
```

```
sigma = getOptimalSigma(centers);
var tempC:RealMatrix;
var tempA:RealMatrix;
if (funName=="Guassian"){
        for ( i=0; i<m; i++ ){
                tempA = new RealMatrix(A[i]);
                for ( j=0; j<n; j++ ){
                        if (k!=1){
                                tempC = new
                                    RealMatrix(
                                    C[j]);
                                H[i][j]= RBF.
                                    Gaussian(
                                    tempA,tempC
                                    ,sigma );
                        }else{
                                H[i][j]= RBF.
                                    Gaussian(
                                    Number(A[i
                                    ]),Number(C
                                    [j]),sigma
                                    );
                        }
                }
        }
}else if (funName == "multi-quadratic"){
        for ( i=0; i<m; i++ ){
                tempA = new RealMatrix(A[i]);
                for ( j=0; j<n; j++ ){
                        if (k!=1){
                                tempC = new
                                    RealMatrix(
                                    C[j]);
                                H[i][j]= RBF.
                                    MultiQuadratic
                                    ( tempA,
                                    tempC,sigma
                                     );
                        }else{
                                H[i][j]= RBF.
                                    MultiQuadratic
                                    ( Number(A[
                                    i]),Number(
                                    C[j]),sigma
                                     );
                        }
                }
        }
}else if (funName == "inverse multi-quadratic")
    {
        for ( i=0; i<m; i++ ){
                tempA = new RealMatrix(A[i]);
                for ( j=0; j<n; j++ ){
                        if (k!=1){
                                tempC = new
                                    RealMatrix(
                                    C[j]);
                                H[i][j]= RBF.
                                    InverseMultiQuadratic
                                    ( tempA,
                                    tempC,sigma
                                     );
                        }else{
                                H[i][j]= RBF.
                                    InverseMultiQuadratic
                                    ( Number(A[
                                    i]),Number(
                                    C[j]),sigma
                                     );
                        }
                }
        }
}else if (funName == "thin-plate-spline"){
        for ( i=0; i<m; i++ ){
                tempA = new RealMatrix(A[i]);
                for ( j=0; j<n; j++ ){
                        if (k!=1){
```

```
                                                tempC = new
                                                    RealMatrix(
                                                    C[j]);
                                                H[i][j]= RBF.
                                                    ThinPlateSpline
                                                    ( tempA,
                                                    tempC );
                                        }else{
                                                H[i][j]= RBF.
                                                    ThinPlateSpline
                                                    ( Number(A[
                                                    i]),Number(
                                                    C[j]) );
                                        }
                                }
                        }
                }

                public function getPhi():RealMatrix{
                        return phi;
                }

                public function getOutput(X:RealMatrix):RealMatrix{
                        weights = phi.solve(X);
                        return phi.timesMatrix(weights);
                }

                public function getWeights():RealMatrix{
                        var X:RealMatrix;
                        if (weights != null){
                                X = weights;
                        }
                        return X;
                }

        }
}
```

E. RBFN 垂直最小平方法

```
package com.RBFPackage
{
        import com.clevr.matrixalgebra.RealMatrix;


        public class RBFN_OLS
        {
                private var input:RealMatrix;

                private var phi:RealMatrix;

                private var sigma:Number;

                private var tolerance:Number;

                private var error:Array;

                private var dd:Number,errSum:Number;

                private var Basis:Array,tempInput:Array;

                private var Q:RealMatrix;

                private var RBFName:String;
```

```
public var Centers:Array;

private var myOutput:RealMatrix;

private var weights:RealMatrix;

public function RBFN_OLS(learningInput:RealMatrix,
    learningOutput:RealMatrix,tol:Number,funName:String
    ="Guassian")
{
        input = learningInput;
        tempInput = input.getArrayCopy();
        tolerance = tol;
        RBFName = funName;
        myOutput = learningOutput;
        dd = myOutput.innearProduct(myOutput);
        creatRBFMatrix(input,funName);
        initOLS();
}

private function getOptimalSigma(centers:RealMatrix):
    Number{
        var n:int = centers.getRowDimension();
        var l:int = centers.getColumnDimension();
        var max:Number=0.0;
        var min:Number=0.0;
        if ( l==1 ){
                max = centers.getColumnMaximun();
                min = centers.getColumnMinimun();
        }else{
                var k:Number;
                var tempA:RealMatrix;
                var tempB:RealMatrix
                for (var i:int=0;i<n;i++){
                        tempA = centers.getRowVector(i)
                            ;
                        for (var j:int=0;j<n;j++){
                                tempB = centers.
                                    getRowVector(j);
                                k = tempA.minus(tempB).
                                    norm2();
                                max = Math.max(max,k);
                                min = Math.min(min,k);
                        }
                }
        }
        return Math.abs(max - min)/Math.sqrt(n);
}

private function creatRBFMatrix(c:RealMatrix,funName:
    String="Guassian"):void{
        var n:int = c.getRowDimension();
        var m:int = input.getRowDimension();
        var A:Array = input.getArrayCopy();
        var C:Array = c.getArrayCopy();
        phi = new RealMatrix(m,n);
        var H:Array = phi.getArray();
        var i:int;
        var j:int;
        sigma = getOptimalSigma(c);
        var tempC:RealMatrix;
        var tempA:RealMatrix;
        if (funName=="Guassian"){
                for ( i=0; i<m; i++ ){
                        tempA = new RealMatrix(A[i]);
                        for ( j=0; j<n; j++ ){
                                if (n!=1){
                                        tempC = new
                                            RealMatrix(
                                            C[j]);
                                        H[i][j]= RBF.
                                            Gaussian(
                                            tempA,tempC
                                            ,sigma );
                                }else{
                                        H[i][j]= RBF.
                                            Gaussian(
                                            Number(A[i
```

```
                                                ]),Number(C
                                                [j]),sigma
                                                );
                                        }
                                }
                        }
                }else if (funName == "multi-quadratic"){
                        for ( i=0; i<m; i++ ){
                                tempA = new RealMatrix(A[i]);
                                for ( j=0; j<n; j++ ){
                                        if (n!=1){
                                                tempC = new
                                                    RealMatrix(
                                                    C[j]);
                                                H[i][j]= RBF.
                                                    MultiQuadratic
                                                    ( tempA,
                                                    tempC,sigma
                                                     );
                                        }else{
                                                H[i][j]= RBF.
                                                    MultiQuadratic
                                                    ( Number(A[
                                                    i]),Number(
                                                    C[j]),sigma
                                                     );
                                        }
                                }
                        }
                }else if (funName == "inverse multi-quadratic")
                    {
                        for ( i=0; i<m; i++ ){
                                tempA = new RealMatrix(A[i]);
                                for ( j=0; j<n; j++ ){
                                        if (n!=1){
                                                tempC = new
                                                    RealMatrix(
                                                    C[j]);
                                                H[i][j]= RBF.
                                                    InverseMultiQuadratic
                                                    ( tempA,
                                                    tempC,sigma
                                                     );
                                        }else{
                                                H[i][j]= RBF.
                                                    InverseMultiQuadratic
                                                    ( Number(A[
                                                    i]),Number(
                                                    C[j]),sigma
                                                     );
                                        }
                                }
                        }
                }else if (funName == "thin-plate-spline"){
                        for ( i=0; i<m; i++ ){
                                tempA = new RealMatrix(A[i]);
                                for ( j=0; j<n; j++ ){
                                        if (n!=1){
                                                tempC = new
                                                    RealMatrix(
                                                    C[j]);
                                                H[i][j]= RBF.
                                                    ThinPlateSpline
                                                    ( tempA,
                                                    tempC );
                                        }else{
                                                H[i][j]= RBF.
                                                    ThinPlateSpline
                                                    ( Number(A[
                                                    i]),Number(
                                                    C[j]) );
                                        }
                                }
                        }
                }
        }
        private function deleteItemOfArray( A:Array,index:int )
            :Array{
```

```
            /* remove the index item of Array */
            var n:int = A.length;
            if (index+1 == n){
                    A.pop();
                    return A;
            }
            if (index == 0){
                    return A.slice(1,n);
            }
            var tempA:Array = A.slice(0,index);
            var tempB:Array = A.slice(index+1,n);
            var newArray:Array = new Array(n-1);
            for ( var i:int=0;i<index;i++ ){
                    newArray[i] = tempA[i];
            }
            for (i=index;i<n-1;i++){
                    newArray[i] = tempB[i-index];
            }
            return newArray;
    }

    private function Err(A:RealMatrix):Number{
            var h:Number = A.innearProduct(A);
            var g:Number = A.innearProduct(myOutput);
            return (g*g) / (h*dd);
    }

    private function getErrMax(E:RealMatrix):RealMatrix{
            var m:int = E.getRowDimension();
            var errMax:RealMatrix = new RealMatrix(m);
            var e:Array = errMax.getArray();
            for ( var i:int=0; i<m; i++ ){
                    e[i][0] = Err( E.getRowVector(i) );
            }
            return errMax;
    }

    private function initOLS():void{
            /* initial setting */
            Basis = new Array();
            Centers = new Array();
            Q = phi.transpose();
            var q:Array = Q.getArray();
            var initQ:RealMatrix = Q.qr().getQ();//claim to
                get initial error.
            var initErrMax:RealMatrix = getErrMax(initQ);
            var tempErr:Array = initErrMax.getArray();
            for ( var i:int=0; i<tempErr.length; i++ ){
                    tempErr[i] = Number(tempErr[i]);
            }
            errSum = initErrMax.getColumnMaximun();//
                initial error.
            var index:int = tempErr.indexOf(errSum);
            Basis[0] = q[index];
            Centers[0] = tempInput[index];
            tempInput = deleteItemOfArray(tempInput,index);
            Q = new RealMatrix( deleteItemOfArray(q,index)
                );
    }

    private function Gram_Schmidt(basis:RealMatrix,v:
        RealMatrix):Array{
            var m:int = v.getRowDimension();
            var k:int = v.getColumnDimension();
            var n:int = basis.getRowDimension();
            var l:int = basis.getColumnDimension();
            if (k!=1){
                    throw new Error ("v must be one column.
                        ");
            }
            var a:Number;
            var G:RealMatrix;
            if (l==1){
                    a = basis.innearProduct(v) / basis.
                        innearProduct(basis);
                    G = v.minus(basis.times(a));
            }else{
                    var u:RealMatrix;
```

```
                        for ( var i:int=0; i<n; i++ ){
                                u = basis.getRowVector(i);
                                a = u.innearProduct(v) / u.
                                    innearProduct(u);
                                G = v.minus(u.times(a));
                        }
                }
                return G.getArray();
        }

        private function OLS():RealMatrix{
                /* return centers */
                var n:int = input.getRowDimension();
                var m:int = n-1;
                var k:int = 1;
                var gramschmidt:RealMatrix
                var g:Array;
                var tempBasis:RealMatrix;
                var errMatrix:RealMatrix;
                var newError:Number;
                var index:int;
                var tempErr:Array;
                var q:Array;
                while( 1-errSum >= tolerance && k<n ){
                        gramschmidt = new RealMatrix(m);
                        g = gramschmidt.getArray();
                        tempBasis = new RealMatrix(Basis);
                        q = Q.getArray();
                        for ( var i:int=0; i<m; i++ ){
                                g[i] = Gram_Schmidt(tempBasis,Q
                                    .getRowVector(i));
                        }
                        errMatrix = getErrMax(gramschmidt);
                        tempErr = errMatrix.getArrayCopy();
                        for (i=0; i<tempErr.length; i++ ){
                                tempErr[i] = Number(tempErr[i])
                                    ;
                        }
                        newError = errMatrix.getColumnMaximun()
                            ;
                        index = tempErr.indexOf(newError);
                        Basis[k] = q[index];
                        Centers[k] = tempInput[index];
                        tempInput = deleteItemOfArray(tempInput
                            ,index);
                        errSum += newError;
                        Q = new RealMatrix( deleteItemOfArray(q
                            ,index) );
                        k++;
                        m--;
                }
                return new RealMatrix(Centers);
        }
        public function getOutput():RealMatrix{
                var RealCenters:RealMatrix = OLS();
                creatRBFMatrix(RealCenters,RBFName);
                weights = phi.solve(myOutput);
                return phi.timesMatrix(weights);
        }

        public function getCenters():RealMatrix{
                return new RealMatrix(Centers);
        }

        public function getWeights():RealMatrix{
                return weights;
        }

    }
}
```

參考文獻

[1] M. D. Buhmann. Radial basis functions. *Acta Numerica*, 2000.

[2] S. Chen, C.F.N. Cowan, and P.M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *Neural Networks, IEEE Transactions on*, 2(2):302 –309, mar 1991.

[3] S.P. Day and M.R. Davenport. Continuous-time temporal back-propagation with adaptable time delays. *Neural Networks, IEEE Transactions on*, 4(2):348 –354, mar 1993.

[4] Mustapha Guezouri. A New Approach Using Temporal Radial Basis Function in Chronological Series, 2008.

[5] Simon Haykin. *Neural Networks: A Comprehensive Foundation (2nd Edition)*. Prentice Hall, 2 edition, July 1998.

[6] Robert J. Howlett and Lakhmi C. Jain. *Radial Basis Function Networks 1: Recent Developments in Theory and Applications*. Physica-Verlag HD; 1 edition, April 27, 2001.

[7] Daw-Tung Lin, Judith E. Dayhoff, and Panos A. Ligomenides. A Learning Algorithm for Adaptive Time-Delays in a Temporal Neural Network. 1992.

[8] D.T. Lin. The Adaptive Time-Delay Neural Network: Characterization and Applications to Pattern Recognition, Prediction and Signal Processing. 1994.

[9] D.T. Lin and J.E. Dayhof. Network Unfolding Algorithm and Universal Spatiotemporal Function Approximation. Technical research report tr95-6, Institute for system research ISR, University of Maryland, 1995.

[10] M. J. D. Powell. Radial basis functions for multivariable interpolation: a review. pages 143–167, 1987.

[11] N.K. Sinha and B. Kuszta. *Modeling and identification of dynamic systems*. Van Nostrand Reinhold, New York, 1983.

[12] C. Wohler and J.K. Anlauf. Real time object recognition on image sequences with adaptable time delay neural network algorithm -application to autonomous vehicles. *Image and Vision*, 19(9–10):593–618, 2001.

[13] P. Yee and S. Haykin. A dynamic regularized radial basis function network for nonlinear, nonstationary time series prediction. *Signal Processing, IEEE Transactions on*, 47(9):2503 –2521, sep 1999.

[14] Paul V. Yee and Simon Haykin. *Regularized radial basis function networks : theory and applications*. Wiley-Interscience; 1 edition, April 2, 2001.

[15] 張斐章、張麗秋、黃浩倫. 類神經網路理論與實務. 東華書局, 2004.

[16] 張麗秋、林永堂、張斐章. Building Radial Basic Function Neural Network by Integrating OLS and SGA for Flood Forecasting. *Journal of Taiwan Water Conservancy*, 2005.

[17] 林永堂. A Study of Combined OLS with SGA to Construct RBF Neural Networks for Flood Forecasting. 2004.

[18] 陳冠廷. The Application of Artificial Neural Networks in a Case-Based Design Wind Load Expert System for Tall Buildings. 2008.

[19] 陳映中. An Rbf Neural Network Method for Image Progressive Transmission. 2000.