類神經網路探詢—以模擬拖車倒車爲例

黃敬仁

摘 要

在現實生活中，拖車倒車入庫是一個高度非線性的問題，所需要考量的變數相當多，以目前的理論而言，尚沒有一個公式能表示出它的運動方程式，而對於一個拖車的駕駛，也必須經由不斷的練習與修正方向盤的角度，才能熟練的將車停至定位。或許，在倒車時，還需要將車子前後的移動方向，才會調整至正確的位置。本文假設拖車只允許倒車；在這個限制下，倒車入庫的困難度也就提高了不少，在我們的研究中，選擇了拖車倒車入庫，作爲我們應用的例子，研究的目標是要學習拖車倒車運動的軌跡，並藉以分析我們所應用的類神經網路架構，是否適合於此類問題。

第一章 導 論

1－1 問題定義

一般而言，將一個問題模組化處理有下列四個步驟（註十一）：

(1) 一個系統要模組化之前，先決定有那些變數需要考慮。
(2) 依據需要考慮的變數與系統環境之間的關係，來分析整個系統的行爲。
(3) 分析整個系統的行爲後，再確定系統模組化的架構。
(4) 找出系統每一參數的值，以完成整個模組化的過程。

在前面的四個步驟中，想要將一個系統正確的描繪出，端賴於一個研究人員對問題的認知程度，以及大量的經驗與知識。否則，建立的模組可能無法表達系統的眞實行爲。

往往一個系統，我們只知道此系統的輸入以及相對於此輸入的輸出值，卻無法知道從輸入轉變成輸出的過程，以及在這個轉變的過程中，到底有多少個相對應的參數在改變而促使這個結果的產生。對於這個系統內部的轉換，也就是所謂的黑盒子問題，一直是許多研究人員所無法完全控制的，這也是我們在研究發展上，一直努力想要克服的目標。如果對於所有的系統，我們都能解開這個令人迷惑的黑盒子；或者我們可以不需要知道黑盒子裡面的過程到底是甚麼，而以一個我們所能理解的模式取而代之，繼續進行我們的研究，而同樣地也能得到一個令人我們滿意的結果，則這個模式將是我們所企盼的，而且是在系統工程研究上的一大突破。

類神經網路(neural network)就是一種能符合上面所提到的模式，它是一種以人類腦部神經系統模式爲基礎的資訊處理結構。以一個更正式的工程解說，將其解釋爲：類神經網路是高度平行動態系統，經由它的內部結構相對於最初狀態的連續性學習回應，而實現了對資訊的處理能力（註七）。

使用類神經網路作爲一個學習的工具，我們並不需要很清楚的知道所要學習的模式是什麼，只要能定義出我們的類神經網路模型，並賦予模型中各單元間連接之權數(weight)，便可以經由不斷的學習，而得到一個可以接受的結果。這個方法使我們降低了對複雜系統的結構未知性而造成的困擾，也比較類似人類眞正的思維模式。

在現實生活中，拖車倒車入庫是一個高度非線性的問題，所需要考量的變數相當多，以目前的理論而言，尚沒有一個公式能表示出它的運動方程式，更遑論建立一個控制機構來執行駕駛的工作；而對於一個拖車的駕駛，也必須經由不斷的練習與修正方向盤的角度，才能熟練的將車停至定位。或許，在倒車時，還需要將車子前後的移動方向，才會調整至正確的位置。本文假設拖車只允許倒車；在這個限制下，倒車入庫的困難度也就提高了不少，在我們的研究中，選擇了拖車倒車，作爲我們應用的例子，研究的目標是要學習拖車倒車運動的軌跡，並藉以分析我們所應用的類神經網路架構，是否適合於此類問題。

1－2 研究方法

Derrick Nguyen與Bernard Widrow在他們所發表的一篇論文（註八）：“ The Truck Backer-Upper：An Example of Self-Learning in Neural Networks”中，以一固定的類神經網路架構，處理拖車倒車入庫問題，在學習上，已經有了足以令人驚訝的結果，在本文中，將參考這篇論文的仿效器(emulator)架構以及Widrow另一篇論文（註十二）的公式推導，作爲本論文的主要研究依據。

前面我們所提到的倒車入庫運動方程式，並沒有一個很明確的公式推導出來，因爲它的運動規則是一個高度非線性的結果，基於這個理由，我們在推導時，作了一些假設，以便符合一項運動規則，作爲類神經網路學習的數據來源。因此，我們可以開始設計研究的方法。

首先，建立拖車倒車入庫的簡化運動方程式，然後，再將推導出來的運動方程式，轉化成電腦程式，作爲拖車學習運動之模擬器(simulator)，所有標準的學習資料，將由模擬器計算出，至於詳細的推導過程，將在3－1節中敘述。

其次，應用Derrick Nguyen與Bernard Widrow所建構的仿效器，將它轉換爲我們所設計的電腦程式，作爲模擬拖車運動的系統程式，仿效器的公式推導，將在3－2節中予以描述。

最後，將我們原先所設計之模擬器與仿效器串連起來，組合成一個總體模組，應用所得到的總體模組，作爲整個系統的學習工具，學習的結果將在本文加以分析。

第二章 Nguyen與Widrow學習模式

Derrick Nguyen 與Bernard Widrow 發表了一篇“ The Truck Backer-Upper：An Example of Self-Learning in Neural Networks”（註八）。在這篇文章中，他們以不同的類神經網路模型，很成功的控制了拖車倒車入庫，這項成果很值得我們深入研究，在本節中，將依據他們所發表的論文，作一番概略性的介紹。

Derrick Nguyen 與Bernard Widrow 的類神經網路模型，主要分爲兩部份，一個是仿效器(emula-tor)，一個是控制器(controller)。依效器主要的功能是學習拖車之移動軌跡，當輸入一個控制信號給仿效器時，仿效器便能由輸入的控制信號與現在的位置狀態，模擬出拖車下一個移動的位置。而控制器主要的功能是藉由本身所產生之控制信號控制仿效器，以決定下一個移動的方向。

圖2－1是整個類神經網路的概略圖，它表示了現在狀態向量S_K是如何送給控制器，及整個學習循環是如何進行。

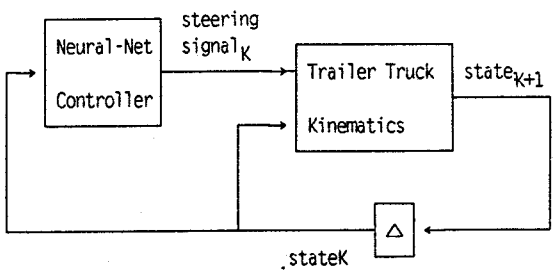


圖 2－1 整個類神經網路的概略圖（註八）

圖2－2是訓練仿效器的流程圖。開始時，拖車由隨機產生之控制信號控制，然後依照控制信號向後倒退，在每一個過程中，仿效器經由不斷的調整內部的權數，來學習拖車是如何移動的；在經過許多的循環後，仿效器學會了拖車移動的行爲模式，然後停止了訓練仿效器的步驟，並記下了仿效器類神網路模型內部所有的權數，如此，當控制器與仿

效器連接後，仿效器便能依照所得到的控制信號，正確地產生拖車的下一個位置。

下圖是訓練仿效器的流程圖。

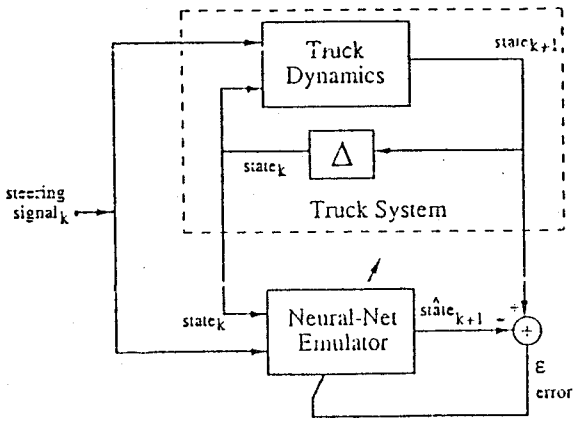


圖2－2 訓練仿效器的流程圖（註八）

圖2－3是訓練控制器的流程圖。圖中所有在方塊中以C標示的是控制器，以T標示的是拖車的仿效器。

假設現在開始進行拖車倒車入庫的學習過程，則我們可以將整個學習的過程，描述如下：

最初，控制器的內部值是任意給定的，當系統將拖車的最初狀態S_0送給控制器及仿效器後，控制器便依照這個最初狀態產生一個控制信號給仿效器，然後仿效器就很快的將拖車移動至下一個位置狀態S_1，並將此時的位置狀態S_1送給控制器，控制器再依照狀態S1產生一個控制信號給仿效器，以決定下一個位置，如此反覆進行。以一個通式表示，控制器接受由仿效器模擬所產生的現在位置狀態，以S_{K-1}表示，然後產生了控制信號K－1；信號的值爲－1或＋1，分別代表向右轉及向左轉，在控制器產生了控制信號以後，便將此控制信號送給仿效器，由仿效器再模擬拖車的運動方式，產生下一個位置狀態S_K，如此持續進行，直到拖車撞到東西後停止爲止。當拖車停止後，將拖車之最後狀態S_K與給定的最後狀態比較，得到一個誤差值 ε_K，在再利用此誤差值 ε_K調整控制器之內部權數，再重新學習，直到成功爲止。

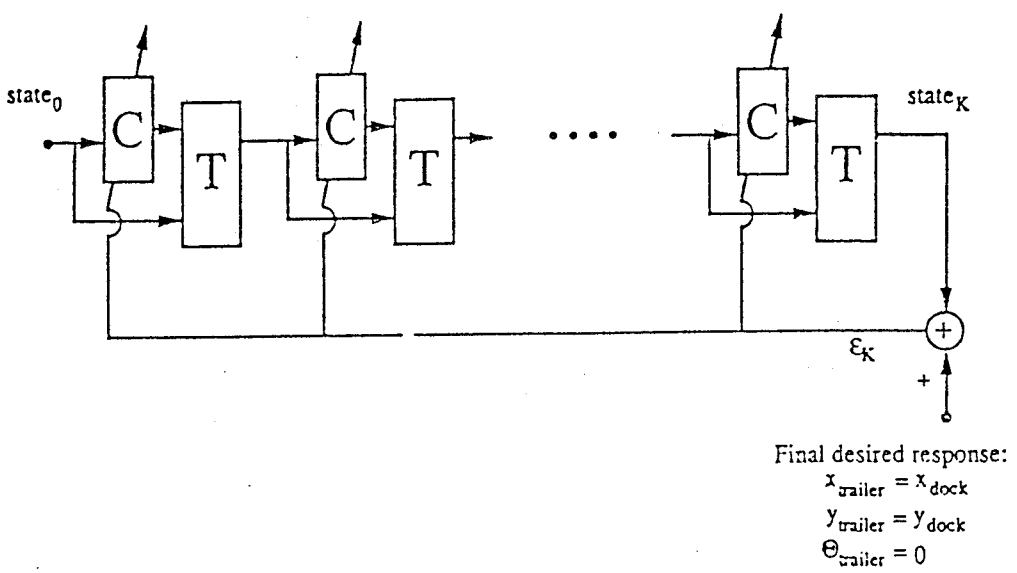


圖2－3 訓練控制器的流程圖（註八）

下圖爲控制器與仿效器之類神經網路模型之簡圖。

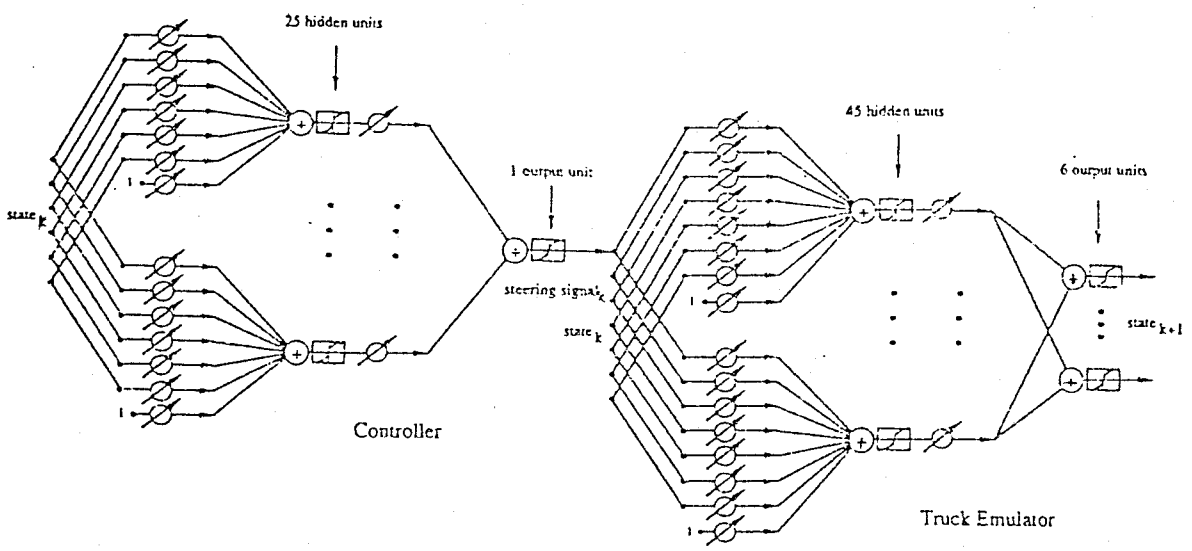


圖2－4　控制器與仿效器類神經網路模型之簡圖（註八）

由上圖我們可以知道，右邊的仿效器是一個具有兩層權數之類神經網路；第一層在仿效器的左邊，它的輸入值爲六個現在的位置狀態，及一個因現在位置由控制器所產生的控制信號，這一層包含有45個隱藏的可調整ADALINE 單元，用以產生下一個將移動的位置；第二層在仿效器的右邊，有六個輸出單元，它主要的功能是由第一層得到調整值，再經由這一層的權數計算，產生拖車的下一個移動位置狀態，並將此新的位置狀態送至仿效器與控制器。

圖形的左邊是控制器的簡圖。它如同仿效器一樣，是一個具有兩層權數之類神經網路；第一層在控制器的左邊，它以六個現在的位置狀態作爲輸入值，這一層包含有25個隱藏的可調整ADALINE 單元；第二層在控制器的右邊，僅包含１個隱藏的可調整ADALINE 單元，而以控制信號作爲它的輸出值。所以，整個系統單元是具有四層的類神經網路模型。如果從學習開始至學習結束共經過Ｋ個狀態，則共有Ｋ個系統單元，也就是說，有４×Ｋ層類神經網路。隨著學習步驟之總次數不同，就有不同的層數值產生。

第三章　系統設計與實驗

3－1　模擬器公式推導

本論文的模擬器(simulator)，將參考 Nguyen 與 Widrow 所定義的座標位置。

3－1－1　變數及符號定義

圖3－1爲拖車之示意圖形。在圖中標示了拖車的位置定義，分別爲：

X_{cab}：拖車箱掛鉤座標之水平位置

Y_{cab}：拖車箱掛鉤座標之垂直位置

$X_{trailer}$：拖車箱尾端中心點座標之水平位置

$Y_{trailer}$：拖車箱尾端中心點座標之水平位置

X_{dock}：車庫中心點標之水平位置

Y_{dock}：車庫中心點標之水平位置

θ_{cab}：拖車頭相對於地面之旋轉角

$\theta_{trailer}$：拖車箱相對於地面之旋轉角

θ_{wheel}：前輪相對拖車頭θ_{cab}之旋轉角

圖3－2　爲推導拖車運動方程式之符號示意圖形。各符號的意義分別爲：

a：拖車頭寬度

b：拖車頭長度

c：拖車頭與拖車箱之間的長度，其值大於 $\frac{a}{2}$

d：拖車箱長度

e：拖車箱寬度，其值等於a

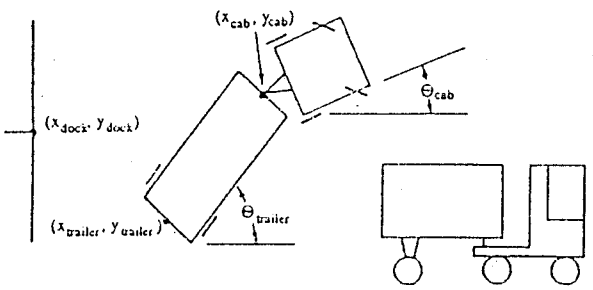
A：拖車前車輪中心聯線之中心點

B：拖車箱掛鉤位置（水平位置B_x，垂直位置B_y）

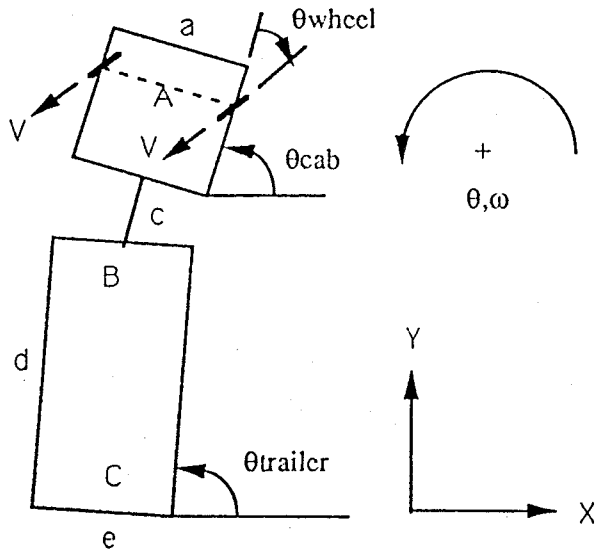
C：拖車箱尾端之中心點

V：拖車移動之速度

△T：拖車移動之時間區間



3－1 拖車之示意圖形（註八）



3－2 拖車運動方程式之符號示意圖形

3－1－2 基本假設

(1) 拖車在移動時並沒有額外之能量消耗，所有之能量用於移動拖車。

(2) 拖車移動的時間區間非常小。

(3) 對於每一時間循環，拖車移動相同的距離，即以定速 V 移動。

3－1－3 公式推導

以拖車車頭做力系分析，並假設前輪輪胎之移動速度V ，可以平移至拖車前車輪中心聯線之中心點，即點A。令$V_1=V$，我們直接以 V 表示 V_1，並且繪出其速度分量，則可得到拖車車頭之速度分離體圖(freebody diagram)。

下圖爲拖車車頭之速度分離體圖：

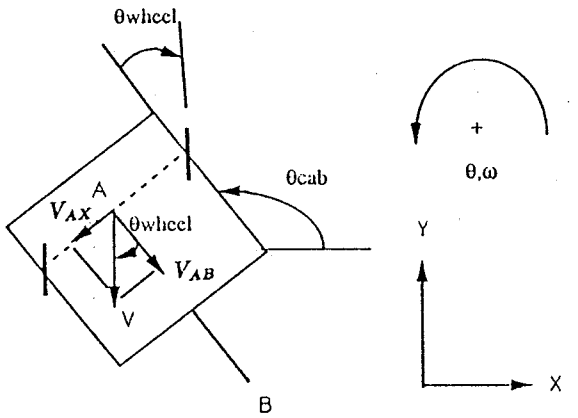


圖3－3 拖車車頭之速度分離體圖

在上圖中，我們使用的符號定義如下：

V_{AB}：速度 V 在AB線上之速度分量。

V_{AX}：速度 V 在拖車頭輪心連線之速度分量。

因此我們可得：

$$V_{AX}=V\sin(\theta_{wheel})$$
$$V_{AB}=-V\cos(\theta_{wheel})$$

點 A 對點 B 之角速度ω_{cab}：

$$\omega_{cab}=-\frac{V_{AX}}{AB}$$

由VAX所造成之點 A 的旋轉角：

$$\triangle\theta_{cab}=\omega_{cab}\triangle T$$

則θ_{cab}受拖車移動影響後之新角度爲：

$$\theta_{cabnew}=\theta_{cab}+\triangle\theta_{cab} \qquad (3\text{-}1)$$

假設拖車箱掛鉤位置B ，因受速度V 之作用，而移至一新的定點，爲求B 之新位置，假設兩種情況，其一爲拖車頭先後移再旋轉，其一爲拖車頭先旋轉再後移。將這兩種情況所產生的位移取平均值，以此平均值作爲B 點之位移量。

考慮情況一，拖車頭先後移再旋轉，則

$$\triangle X_1 = - V_{AB}\sin(\theta_{cab} - \frac{\pi}{2})\triangle T$$

$$\triangle Y_1 = V_{AB}\cos(\theta_{cab} - \frac{\pi}{2})\triangle T$$

考慮情況二，拖車頭先旋轉再後移，則

$$\triangle X_2 = - V_{AB}\sin(\theta_{cabnew} - \frac{\pi}{2})\triangle T$$

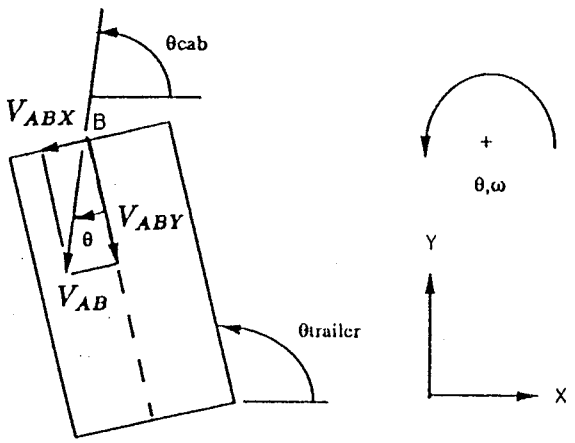
$$\triangle Y_2 = V_{AB}\cos(\theta_{cabnew} - \frac{\pi}{2})\triangle T$$

所以，我們可以得到點 B 的新位置：

$$B_{xnew} = B_x + \frac{\triangle X_1 + \triangle X_2}{2} \qquad (3\text{-}2)$$

$$B_{ynew} = B_y + \frac{\triangle Y_1 + \triangle Y_2}{2} \qquad (3\text{-}3)$$

取拖車車箱作爲分析，以拖車箱掛鉤位置B 作速度分析圖。



3—4 拖車車箱之掛鉤位置 B 作速度分析圖

由於拖車箱受速度V_{AB} 之作用而產生旋轉，改變了拖車箱原先之角度。將速度V_{AB} 分解爲一沿拖車箱前沿之速度分量與一沿拖車箱箱身縱線之分量。

則沿拖車箱前沿之速度分量爲

$$V_{ABX} = V_{AB}\sin(\theta)$$

$$= V_{AB}\sin(\theta_{trailer} - \theta_{cab})$$

另外沿拖車箱箱身縱線之分量

$$V_{ABY} = V_{AB}\cos(\theta)$$

$$= V_{AB}\cos(\theta_{trailer} - \theta_{cab})$$

因此，拖車箱的角速度$\omega_{trailer}$爲：

$$\omega_{trailer} = - \frac{V_{ABX}}{BC}$$

在拖車箱上的點 B 之角位移：

$$\triangle\theta_{trailer} = \omega_{trailer}\triangle T$$

拖車箱角度$\theta_{trailer}$之最後角度爲：

$$\theta_{trailernew} = \theta_{trailer} + \triangle\theta_{trailer} \qquad (3\text{-}4)$$

由方程式(3-1)、(3-2)、(3-3)、(3-4)，我們可得到拖車運動一微小時間△T後之移動位置。

3—2 仿效器架構及公式推導

在前面的章節中，我們曾經提到 Nguyen 與 Widrow 所發表的一篇“ The Truck Backer-Upper : An Example of Self-Learning in Neural Networks”，在本節中，仿效器將藉用他們所發展的結構，但是所應用的仿效器與原作者稍有不同，主要的差別在仿效器的第二層權數位置，原作者所繪圖形表示由第一層的隱藏單元透過第二層權數，再連接至六個輸出單元，對每一個隱藏單元而言，僅以單一權數值同時連接六個輸出單元，依據原作者所繪製之圖形，則將產生六個輸出單元所收到的輸入強度值均相同的情形，如此，六個輸出單元便會有相同的輸出值，此點與作者的說明，不無可議之處；基於這個原因，我們將仿效器的類神經網路模型稍做修改，並且減少了輸出單元的數目，將其減爲四個輸出單元，把每一個隱藏單元，各以不同的權數值分別連接至相對應的輸出單元，也就是說，對一個隱藏單元而言，將有四個權數分別連接至

四個相對應的輸出單元，則每個輸出單元將有可能得到不同的輸入强度，這種接法或許較符合作者的原意。

下面的公式推導，以雙曲線正切函數作爲隱藏單元之致動函數，以線性函數 (linear function) 作爲輸出單元之致動函數。

以下爲公式中所使用的符號標示說明。我們以依效器的第一層作爲輸入層，每一輸入層有P 個輸入，而第一層共有q 個輸入層，則每一輸入層的相對應的權數指標以pq表示；第二層爲輸出層，由於每一個輸入層有一相對應的輸出分別輸出至輸出層，因此每一個輸出層之輸出單元分別可收到q 個輸入，但輸出層共有m 個輸出單元，所以輸出層之權數指標以qm表示；經過符號的定義後，我們可以開始進行公式的推導。

公式推導

假　設

X_p：輸入單元

其中 $p = 1,2,\cdots\cdots P$ 表示輸入單元

X_8：1

考慮在仿效器的第一層內的每一組ADALINE，隱藏單元自任一組ADALINE 所得到的輸入强度爲：

$$Net_q = \sum_{p=1}^{P} X_p W_{pq}$$

$$q = 1,2,\cdots\cdots,Q \qquad (3\text{-}5)$$

其中 q 表示隱藏單元

第一層內，隱藏單元的輸入值，經過致動函數（雙曲線正切函數）轉換後，得到每一隱藏單元之輸出值爲

$$Sgm_q = \tanh(Net_q)$$

第二層內的輸出單元，由第一層隱藏單元所得到的輸入强度爲

$$Net_m = \sum_{q=1}^{Q} Sgm_q W_{qm}$$

$$m = 1,2,\cdots\cdots,M \qquad (3\text{-}6)$$

其中 m 表示輸出單元總數

假設任一輸出單元與系統目標值的誤差爲

$$\varepsilon_m = (D_m - Net_m)$$

則對於系統誤差值平方和爲

$$\varepsilon^2 = \sum_{m=1}^{M} \varepsilon_m^2$$

$$= \sum_{m=1}^{M} (D_m - Net_m)^2$$

假設輸出值之誤差平方和，對任一層的第 j 個ANALINE 輸入强度之導數定義爲：

$$\delta_j = -\frac{1}{2}\frac{\partial \varepsilon^2}{\partial S_j}$$

推導第二層之瞬間平方誤差(instantaneous square error)；利用連鎖率得

$$\delta_m = -\frac{1}{2}\frac{\partial \varepsilon^2}{\partial Net_m}$$

$$= -\frac{1}{2}\frac{\partial \{\sum_{m=1}^{M} (D_m - Net_m)^2\}}{\partial Net_m}$$

$$= (D_m - Net_m)$$

而第一層之瞬間平方誤差爲

$$\delta_q = -\frac{1}{2}\frac{\partial \varepsilon^2}{\partial Net_q}$$

$$= -\frac{1}{2}\frac{\partial \{\sum_{m=1}^{M} (D_m - Net_m)^2\}}{\partial Net_q}$$

$$= \left(\sum_{m=1}^{M} (D_m - Net_m)\frac{\partial Net_m}{\partial Net_q}\right)$$

$$= \left(\sum_{m=1}^{M} (D_m - Net_m)\frac{\partial Net_m}{\partial Sgm_q}\right)\left(\frac{\partial Sgm_q}{\partial Net_q}\right)$$

$$= \left(\sum_{m=1}^{M} (D_m - Net_m)\, W_{qm} \right) (1 - Sgm_q^2)$$

假設在時間K 時，網路中任一ADALINE 的瞬間誤差梯度(instantaneous error gradient)爲系統輸出值與目標值差之平方和相對ADALINE 中任一權數之導數，即

$$\widehat{\nabla} k = \frac{\partial \varepsilon_k^2}{\partial W_k}$$

則我們可推導出第二層之瞬間誤差梯度爲

$$\widehat{\nabla}_{gm} = \frac{\partial \varepsilon^2}{\partial W_{gm}}$$

$$= \left(\frac{\partial \varepsilon^2}{\partial Net_m}\right) \left(\frac{\partial Net_m}{\partial W_{qm}}\right)$$

$$= \left(\frac{\partial \{\sum_{m=1}^{M} (D_m - Net_m)^2\}}{\partial Net_m}\right) \left(\frac{\partial Net_m}{\partial W_{qm}}\right)$$

$$= -2(D_m - Net_m) Sgm_q$$

將 δ_m 值與上式比較得第二層之瞬間誤差梯度爲

$$\widehat{\nabla} q_m = -2\delta_m Sgm_q \qquad (3\text{-}7)$$

推導第一層之瞬間誤差梯度，可以得到

$$\widehat{\nabla} pq = \frac{\partial \varepsilon^2}{\partial W_{pq}}$$

$$= \left(\frac{\partial \varepsilon^2}{\partial Net_q}\right) \left(\frac{\partial Net_q}{\partial W_{pq}}\right)$$

$$= \left(\frac{\partial \{\sum_{m=1}^{M} (D_m - Net_m)^2\}}{Net_q}\right) \left(\frac{\partial Net_q}{\partial W_{pq}}\right)$$

$$= -2 \left(\sum_{m=1}^{M} (D_m - Net_m) \frac{\partial Net_m}{\partial Net_q} \right) \left(\frac{\partial Net_q}{\partial W_{pq}}\right)$$

$$= -2 \left(\sum_{m=1}^{M} (D_m - Net_m) \frac{\partial Net_m}{\partial Sgm_q} \right) \left(\frac{\partial Sgm_q}{\partial Net_q}\right) \left(\frac{\partial Net_q}{\partial W_{pq}}\right)$$

$$= -2 \left(\sum_{m=1}^{M} (D_m - Net_m) W_{qm} \right) (1 - Sgm_q^2) X_p$$

將 δ_q 值與上式比較得第一層之瞬間誤差梯度爲

$$\widehat{\nabla}_{pq} = -2\delta_q X_p \qquad (3\text{-}8)$$

考慮在時間 k 與 k+1 時，由最陡下降法(steepest descent method)觀念知

$$W_{(k+1)} = W_{(k)} + \mu \left(-\nabla_{(k)}\right)$$

其中 W 表示權數，μ 爲學習速率，$\widehat{\nabla}(k)$爲瞬間誤差梯度。

將方程式(3-7) 所得到的結果代入上式，可計算出第二層權數變化公式爲

$$W_{qm(k+1)} = W_{qm(k)} + \mu_{m(k)} \left(-\widehat{\nabla}_{qm(k)}\right)$$

$$= W_{qm(k)} + 2\mu_{m(k)} \delta_{m(k)} Sgm_{q(k)}$$

同時

$$\Delta W_{qm(k)} = 2\mu_{m(k)} \delta_{m(k)} Sgm_{q(k)} \qquad (3\text{-}9)$$

將方程式(3－8)所得到的結果代入上式，可計算出第一層權數變化公式爲

$$W_{pq(k+1)} = W_{pq(k)} + \mu_{q(k)} \left(-\widehat{\nabla}_{pq(k)}\right)$$

$$= W_{pq(k)} + 2\mu_{q(k)} \delta_{q(k)} X_{p(k)}$$

同時

$$\Delta W_{pq(k)} = 2\mu_{q(k)} \delta_{q(k)} X_{q(k)} \qquad (3\text{-}10)$$

在後推法則中，μ 值在系統學習的過程裡，可用以減少對權數的梯度干擾；如果考慮應用動量技術，則可將$W_{qm(k+1)}$改寫爲：

$$\Delta W_{qm(k)} = 2\mu_{m(k)}(1-\eta)\delta_{m(k)} Sgm_{q(k)} + \eta \Delta W_{qm(k-1)} \qquad (3\text{-}11)$$

$$W_{qm(k+1)} = W_{qm(k)} + \Delta W_{qm(k)} \qquad (3\text{-}12)$$

而$W_{pq(k+1)}$改寫爲：

$$\triangle W_{pq(k)} = 2\mu_{q(k)}(1-\eta)\delta_{q(k)}X_{p(k)} + \eta\triangle W_{pq(k-1)} \quad (3\text{-}13)$$

$$W_{pq(k+1)} = W_{pq(k)} + \triangle W_{pq(k)} \quad (3\text{-}14)$$

上式中 $0 \leqq \eta < 1$，一般選擇 $0.8 \leqq \eta \leqq 0.9$。

以下推導在權數改變時，動態學習率 μ 值的變化情形。

首先考慮第二層

$$\frac{\partial \varepsilon^2}{\partial Net_m} = \frac{\partial \{\Sigma_{m=1}^{M}(D_m - Net_m)^2\}}{Net_m} = -2(D_m - Net_m)$$

因爲

$$\triangle Net_m \propto -\frac{\varepsilon^2}{Net_m}$$

所以得到$\triangle Net_m$值爲

$$\triangle Net_m = \mu_m^* \{2(D_m - Net_m)\} \quad (3\text{-}15)$$

又由方程式(3－6)知

$$\triangle Net_m = \sum_{q=1}^{Q} Sgm_q(W_{qm} + \triangle W_{qm}) - \sum_{q=1}^{Q} Sgm_q W_{qm} = \sum_{q=1}^{Q} Sgm_q \triangle W_{qm}$$

所以，由方程式(3－9)得

$$\triangle Net_m = \sum_{q=1}^{Q} Sgm_q(2\mu_m \delta_m Sgm_q) \quad (3\text{-}16)$$

由方程式(3－15)與(3－16)得

$$\mu_m^* \{2(D_m - Net_m)\} = \sum_{q=1}^{Q} Sgm_q(2\mu_m \delta_m Sgm_q)$$

將 δ_m 值代入上式，可以得到

$$\mu_m^* \{2(D_m - Net_m)\} = \sum_{q=1}^{Q} Sgm_q \{2\mu_m (D_m - Net_m) Sgm_q\}$$

將上式化簡得

$$\mu_m^* = \sum_{q=1}^{Q} Sgm_p^2 \mu_m$$

故動態學習率 μ_m 之值爲

$$\mu_m = \frac{\mu_m^*}{\Sigma_{q=1}^{Q} Sgm_q^2} \quad (3\text{-}17)$$

接下來考慮第一層

$$\frac{\partial \varepsilon^2}{\partial Net_q} = \frac{\partial \{\Sigma_{m=1}^{M}(D_m - Net_m)^2\}}{Net_q}$$

$$= -2\left(\sum_{m=1}^{M}(D_m - Net_m)\frac{\partial Net_m}{\partial Net_q}\right)$$

$$= -2\left(\sum_{m=1}^{M}(D_m - Net_m)\frac{\partial Net_m}{\partial Sgm_q}\right)\left(\frac{\partial Sgm_q}{\partial Net_q}\right)$$

$$= -2\left(\sum_{m=1}^{M}(D_m - Net_m)W_{qm}\right)(1 - Sgm_q^2)$$

因爲

$$\triangle Net_q \propto -\frac{\partial \varepsilon^2}{\partial Net_q}$$

所以得到$\triangle Net_q$值爲

$$\triangle Net_q = \mu_m^* \left(2\sum_{m=1}^{M}(D_m - Net_m)W_{qm}\right)(1 - Sgm_q^2) \quad (3\text{-}18)$$

又由方程式(3－5)知

$$\triangle Net_q = \sum_{p=1}^{P} X_p(W_{pq} + \triangle W_{pq}) - \sum_{p=1}^{P} X_p W_{pq} = \sum_{p=1}^{P} X_p \triangle W_{pq}$$

所以，由方程式(3－10)得

$$\triangle Net_q = \sum_{p=1}^{P} X_p(2\mu_q \delta_q X_p) \quad (3-19)$$

由方程式 (3－18)與(3－19)得

$$\mu_q^* \left(2\sum_{m=1}^{M}(D_m - Net_m)W_{qm}(1 - Sgm_q^2)\right) = \sum_{p=1}^{P} X_p(2\mu_q \delta_q X_p)$$

將 δ_q 值代入上式，可以得到

$$\mu_q^* \left(2\sum_{m=1}^{M}(D_m - Net_m)W_{qm}(1 - Sgm_q^2)\right)$$

$$= \sum_{p=1}^{P} X_p(2\mu_q(\sum_{m=1}^{M}(D_m - Net_m)W_{qm})(1-Sgm_q^2)X_p)$$

將上式化簡得

$$\mu_q^* = \sum_{p=1}^{P} X_p^2 \mu_q$$

故動態學習μ_q之值爲

$$\mu_q = \frac{\mu_q^*}{\sum_{p=1}^{P} X_p^2}$$

我們可假設$\mu_m^* = 1$與$\mu_q^* = 1$。

在本篇論文的實際拖車行爲模擬中，將採用階段式靜態學習率，而不採用動態學習率。

3－3 學習系統架構

由我們所定義的結構，將學習系統架構繪製如下：

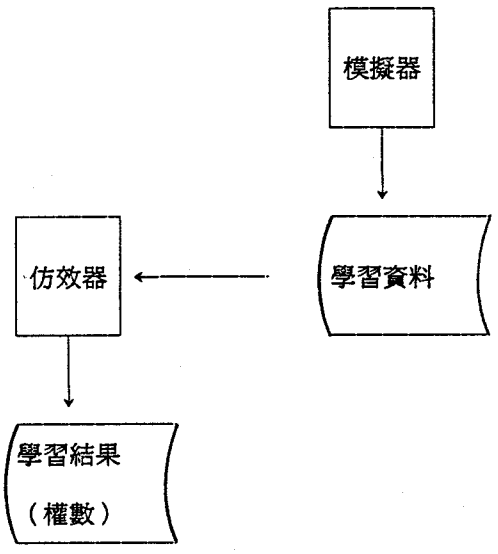


圖 3－5 系統權數學習流程

在系統的建構過程中，我們先將3－1節所推導出來的拖車運動方程式，轉換爲C 語言程式，即圖中之模擬器；再由模擬器依初始位置及前輪輪胎轉動角度之不同，而產生各組的測試資料，將這些測試資料自仿效器中讀入，供仿效器作學習用；最後將產生之學習結果，即一組權數值存檔，以便以後總體模組讀取（總體模組乃模擬器與仿效器串聯後之模組）。

總體模組的產生方式，是將模擬器中計算拖車下一移動位置的副程式，以仿效器之前導公式取代，其餘部份不予改變。因此我們可將總體模組流程圖繪製如下：

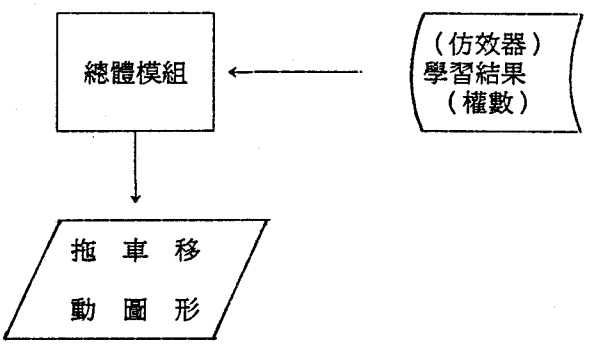


圖 3－6 總體模組使用流程圖

依圖中可知，在圖3－5中仿效器所產生之學習結果（權數值），爲總體模組之輸入數值，總體模組在讀入權數值後，經由模組中之前導公式以及外部輸入之前輪輪胎角度，便可得到拖車之下一移動位置。

3－4 學習結果

在學習的過程中，爲了確定所建構的類神經網路架構，能否達成學習的目的，我們使用了三個函數，作爲測試。這三個函數分別爲：

$$case1: Z = 0.2\exp(X) + \sin(Y^2)$$
$$case2: Z = 0.2XY + 1.2\sin(X+Y)$$
$$case3: Z = \sin(X) + 1.2\cos(Y^2)$$

對於這三個測試函數，我們各取100 點資料，供作系統學習的樣本，以RMSE值(root mean squre of error)與IA值(index of accuracy)作爲判斷學習結果是否良好的依據指標。RMSE與IA值分別定義如下：

$$RMSE = \sqrt{\frac{\Sigma_{n=1}^{N} (D_n - O_n)^2}{N-1}}$$

$$I.A. = \frac{RMSE}{\sqrt{\frac{(\Sigma_{n=1}^{N} D_n^2) - (\frac{(\Sigma_{n=1}^{N} D_n)^2}{N})}{(N-1)}}}$$

在下面的測試中，所使用的η值，爲應用動量技術時，此次權數調整量與上次權數調整量的分配比例，一般爲0.8～0.9，我們也以這個範圍作爲主要的學習區間。茲將學習的過程及發現描述如下。

爲確定所建立之公式是否能達到學習的目的，我們使用了如前所示的三個例子，並將輸出單元之致動函數改爲線性函數，因爲這三個函數之輸出值均不在雙曲線正切函數區間－1至＋1之間。

範例一：

(1) 學習函數：

$Z = 0.2\exp(X) + \sin(Y^2)$

(2) 函數類別：

可分割函數：

所謂可分割函數以case1 而言，可以分割爲兩個函數$0.2\exp(X)$與$\sin(Y^2)$，各爲單變數之函數，而函數Z 可以由此兩予函數線性組合而成，故我們稱Z 爲一可分割函數。

(3) 數值範圍：

輸入：

X：$-1 \leq X \leq 3.95$，增量：0.05。

Y：$0 \leq Y \leq 1.98$，增量：0.02。

輸出：

Z：$0.073576 \leq Z \leq 9.684642$

(4) 樣本數 N：

N=100，分佈情形與 X、Y 增量產生之 Z 相同。

(5) 測試次數 T ：

T ：600。

(6)使用之η值：

$\eta = 0.95$

(7) 靜態學習率μ ：

$T < 200$：$\mu = 0.5 * 0.5 * 0.01$

$200 \leq T < 400$：$\mu = 0.25 * 0.5 * 0.01$

$400 \leq T \leq 600$：$\mu = 0.0625 * 0.5 * 0.01$

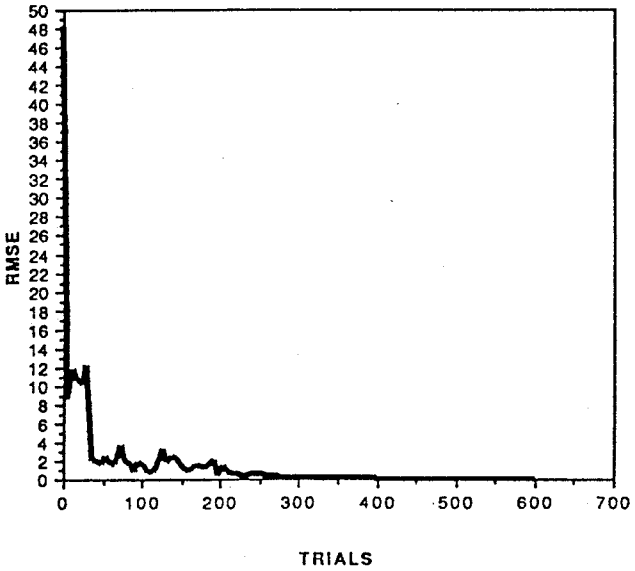
(8) 初始權數值範圍：

第一層：$0.01 \leq W < 0.1$

第二層：$0.1 \leq W < 0.5$

(9) 學習結果：

$Z = 0.2^{*}\exp(X) + \sin(Y^{*}Y)$



3－7 case1當$\eta = 0.95$之RMSE學習曲線圖

下圖爲在上列數據下之I.A.學習曲線圖。

$Z = 0.2^{*}\exp(X) + \sin(Y^{*}Y)$

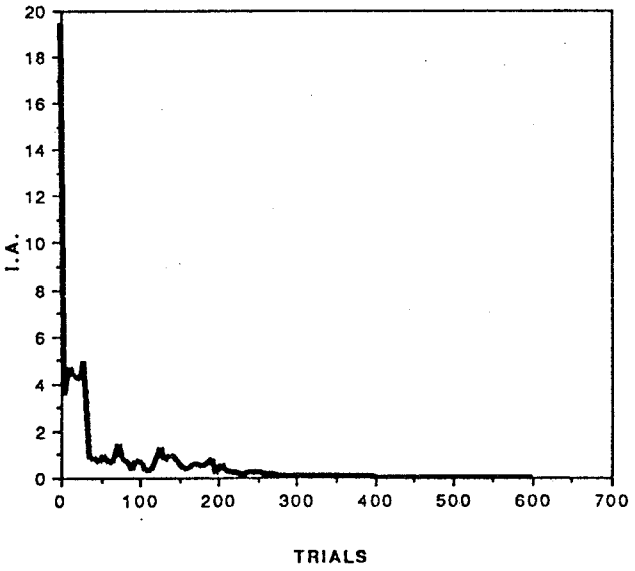


圖3－8 case1當$\eta = 0.95$之I.A.學習曲線圖

(10)討論：

由圖3－7與圖3－8中，我們可以知道網路在經過600 次的學習後，不管是RMSE或IA值，幾乎已達到穩定的狀態，此時的數值

RMSE ≈ 0.009

I.A. ≈ 0.004

由於數值樣本的分佈較廣，所以造成I.A.值比RMSE值小。由圖中的學習曲線可以看出，在學習的過程中有振盪的情形，這與我們所定義的靜態學習率、類神經網路的架構、及樣本資料的範圍有關；當給與不同的學習率，就會產生不同的振盪位置。

在下面的範例中，我們探討η對系統學習情形的影響。

範例二：

(1) 學習函數：

Z = 0.2XY +1.2sin(X+Y)

(2) 函數類別：

不可分割函數：

所謂不可分割函數以case2 而言，函數Z由0.2XY及1.2sin(X+Y)線性組合而成，無法分割爲個別之X 函數或Y 函數，故我們稱Z 爲一不可分割函數。

(3) 數值範圍：

輸入：

X：-1 ≦ X ≦ 3，增量：0.2。

Y： 0 ≦ Y ≦ 1，增量：0.25。

輸出：

Z：-1.438447 ≦ Z ≦ 1.330783

(4) 樣本數 N：

N =273

(5) 測試次數 T ：

T = 600。

(6)使用之η值：

0.8 ≦ η ≦0.99

(7) 靜態學習率μ ：

T<200：μ =0.5*0.5*0.01

200 ≦ T < 400：μ =0.25*0.5*0.01

400 ≦ T ≦ 600：μ =0.0625*0.5*0.01

(8) 初始權數值範圍：

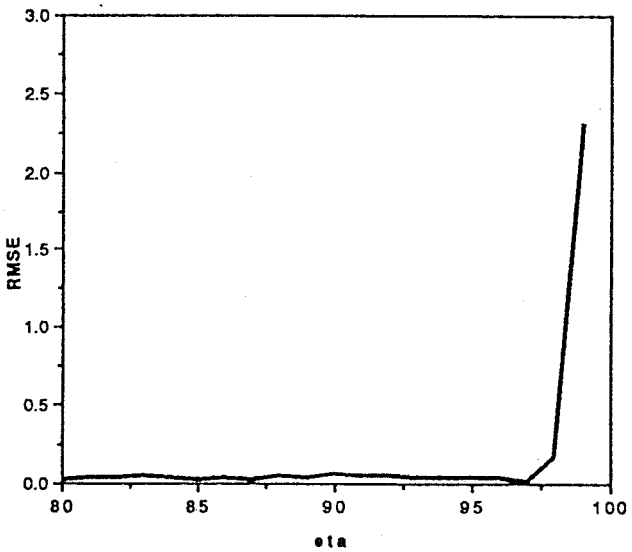
第一層：0.01 ≦ W < 0.1

第二層：0.1 ≦ W < 0.5

(9) 學習結果：

下圖爲在上列數據下之RMSE學習曲線圖。

Z = 0.2XY +1.2sin(X+Y)



3－9 case2當0.8<η<0.99之RMSE學習曲線圖

下圖爲在上列數據下之I.A.學習曲線圖。

Z = 0.2XY +1.2sin(X+Y)

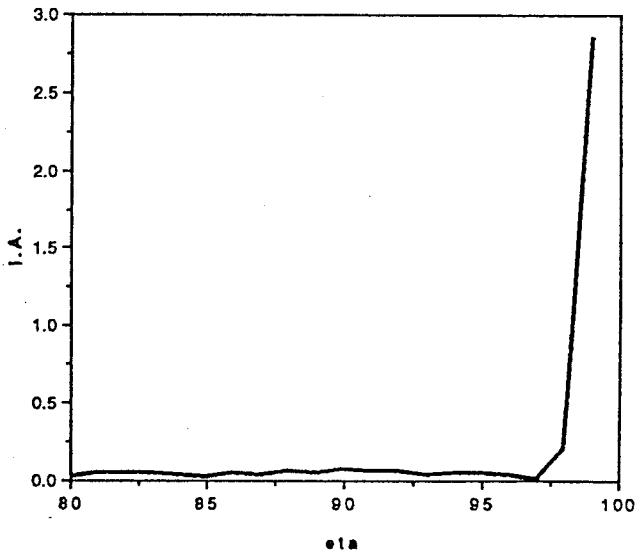


圖3－10 case2當0.8<η<0.99之I.A.學習曲線圖

(10)討論：

在經過600 次的學習後，在η=0.97時有

最佳之RMSE與I.A.。

RMSE ≈ 0.009248

I.A. ≈ 0.012258

且I.A.值比RMSE值大。此因樣本資料取樣範圍小，且取樣又多的原因。由此次的結果可知，當取樣的數量多，且學習目標值的範圍減小，學習的效果較佳。對於學習的結果，曾經作過多次實驗，得到的經驗是，當所給與的初始權數值不同時，發生最佳RMSE與I.A.值之η值並不一定相同。

範例三：

(1) 學習函數：

$Z = \sin(X)+1.2\cos(Y^2)$

(2) 函數類別：

可分割函數：

(3) 數值範圍：

輸入：

X：-1 ≦ X ≦ 3.95，增量：0.05。

Y： 0 ≦ Y ≦ 1.96，增量：0.04。

輸出：

Z：-1.881809 ≦ Z ≦ 2.199782

(4) 樣本數 N：

N ＝100，分佈情形與 X、Y 增量產生之Z 相同。

(5) 測試次數 T ：

T ＝ 600。

(6)使用之η值：

$0.8 \leq \eta \leq 0.99$

(7) 靜態學習率μ：

T＜200：$\mu = 0.5*0.5*0.01$

200 ≦ T ＜ 400：$\mu = 0.25*0.5*0.01$

400 ≦ T ≦ 600：$\mu = 0.0625*0.5*0.01$

(8) 初始權數值範圍：

第一層：0.01 ≦ W ＜ 0.1

第二層：0.1 ≦ W ＜ 0.5

(9) 學習結果：

下圖爲在上列數據下之RMSE學習曲線圖。

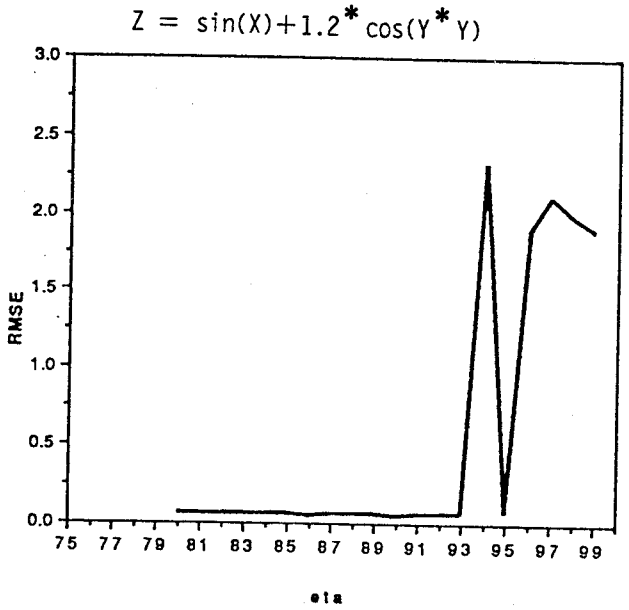


圖3－11 case3η之RMSE學習曲線圖

下圖爲在上列數據下之I.A.學習曲線圖。

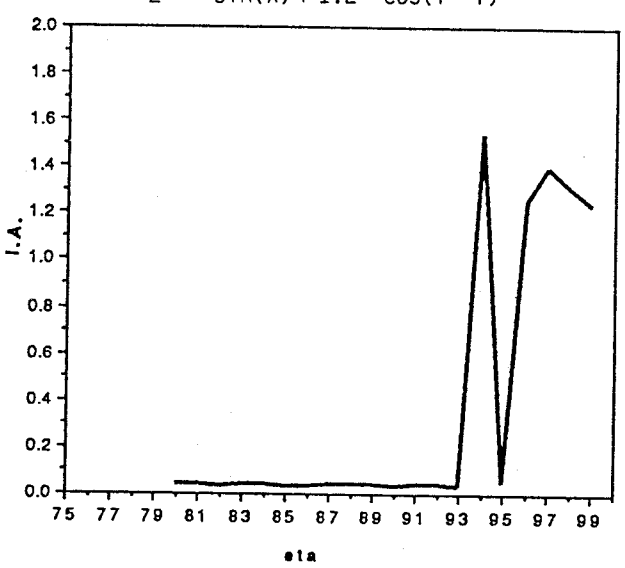


圖3－12 case3η之I.A.學習曲線圖

(10)討論：

由圖中可知，在η＝0.95時有最佳之RMSE與I.A.。

RMSE 0.050953

I.A.　0.033775

且I.A.值比RMSE值小。當η值超過93時，系統學習的曲線會有劇烈振盪的情形。如果，我們以一組不同的初始權數，重新學習，則會有不同的學習曲線圖，振盪的情形也會有所不同，當然最佳RMSE與I.A.之η值也不一定發生在η＝0.95的位置。

經由上面的測試後，開始進行拖車倒車行爲之模擬。如3－2節公式推導，將原先模擬器所輸出的學習資料，乘上一放大倍數，以此作爲眞正的學習資料，在學習過一定次數後，取得學習後的權數，當應用此權數產生輸出時，只需將輸出數值除上原先的放大倍數便得到實際的輸出值，此點效果較應用增益方式爲佳。以下是我們的實驗過程。

範例四：

(1) 學習函數：

應用3－2節公式推導學習拖車倒車行爲。

(2) 數值範圍：

輸入與輸出

θ_{cab}：$-\pi \leq \theta_{cab} \leq \pi$

X_{cab}：$0 \leq X \leq 1$

Y_{cab}：$0 \leq Y \leq 1$

$\theta_{trailer}$：$-\pi \leq \theta_{trailer} \leq \pi$

$$\theta_{wheel}：-\frac{\pi}{3} \leq \theta_{wheel} \leq \frac{\pi}{3}$$

(4) 樣本數 N：

N ＝340。

(5) 測試次數 T ：

T ＝ 5000。

(6)使用之η值：

η＝0.85。

(7) 靜態學習率μ ：

T<200：μ＝0.5＊0.01

200 ≦ T < 400：μ＝0.25＊0.01

400 ≦ T ≦ 600：μ＝0.125＊0.01

T >600：μ＝0.0625＊0.01

(8) 初始權數值範圍：

第一層：0.01 ≦ W < 0.1

第二層：0.01 ≦ W < 0.1

(9) 學習結果：

在測試時，將原先之輸出值放大5倍，得到之數值如下：

Avg－RMSE　0.028604

Avg－I.A.　0.035063

θ_{cab}－RMSE　0.090851

θ_{cab}－I.A.　0.074253

X_{cab}－RMSE　0.004692

X_{cab}－I.A.　0.026229

Y_{cab}－RMSE　0.004231

Y_{cab}－I.A.　0.022905

$\theta_{trailer}$－RMSE　0.014644

$\theta_{trailer}$－I.A.　0.016893

(10)討論：

在本次實驗之前，曾以276 組測試資料放大五倍作測試，經過5000次訓練，所得到的權數值，應用至整體模組以後，觀察拖車移動的軌跡，能符合所給與之測試資料移動的方式，表示所建構的系統已經具有學習的功能；但爲求更加的精確，此次使用了340 組測試資料，擴大組數，但所得到的結果與276 組的結果，差異不大，主要的原因是拖車移動之位置有相當多組，要有相當多的測試資料才能得到眞正的運動圖形。

(11)學習結果圖形：

以下爲(9)學習結果之圖形。

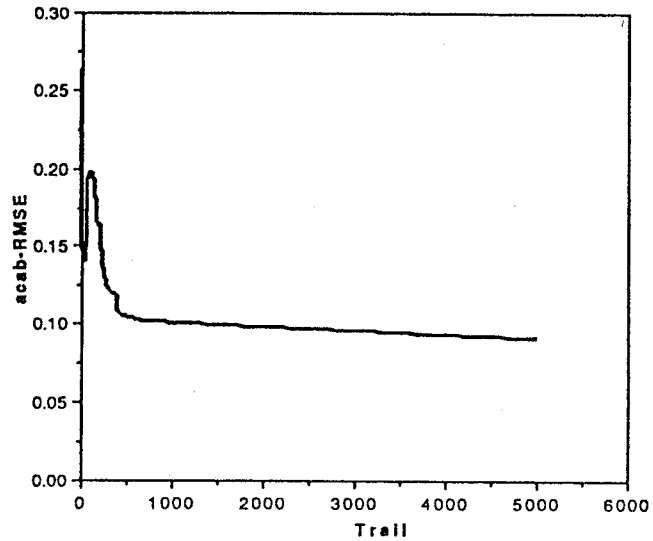


圖3－13 θ_{cab}之RMSE學習曲線圖

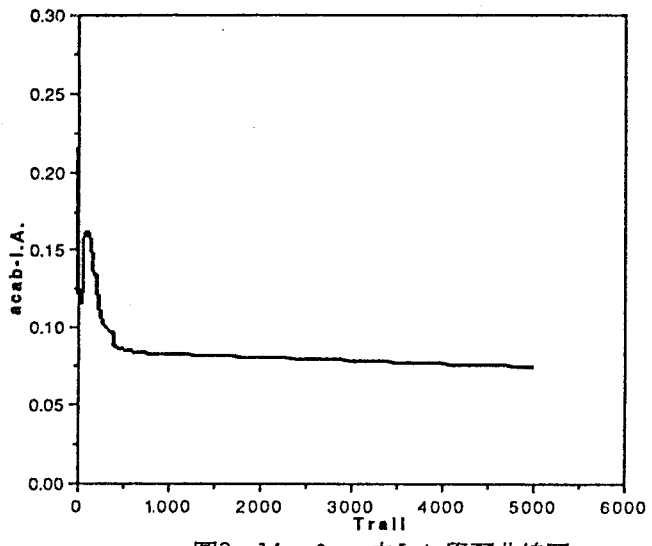


圖3－14 θ_{cab}之I.A.學習曲線圖

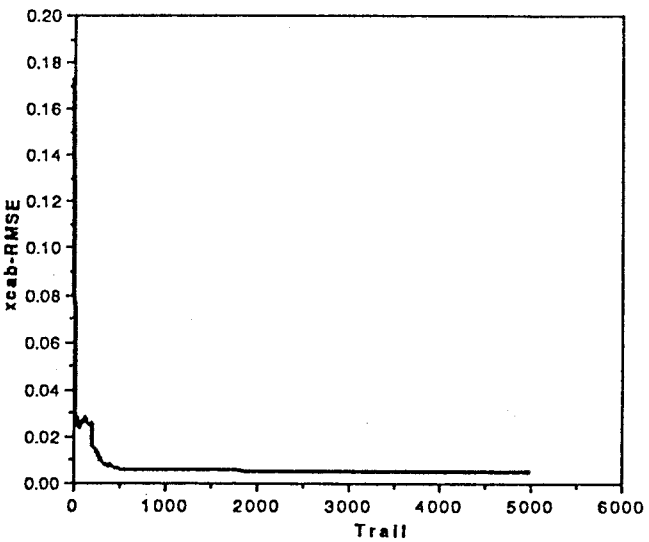


圖3－15 X_{cab}之RMSE學習曲線圖

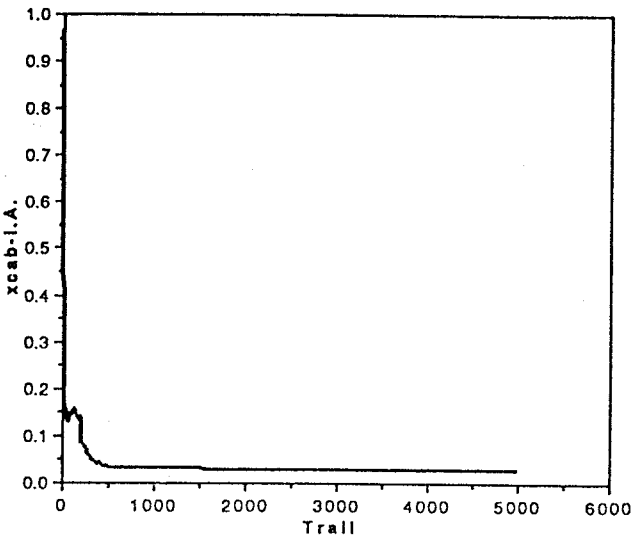


圖3－16 X_{cab}之I.A.學習曲線圖

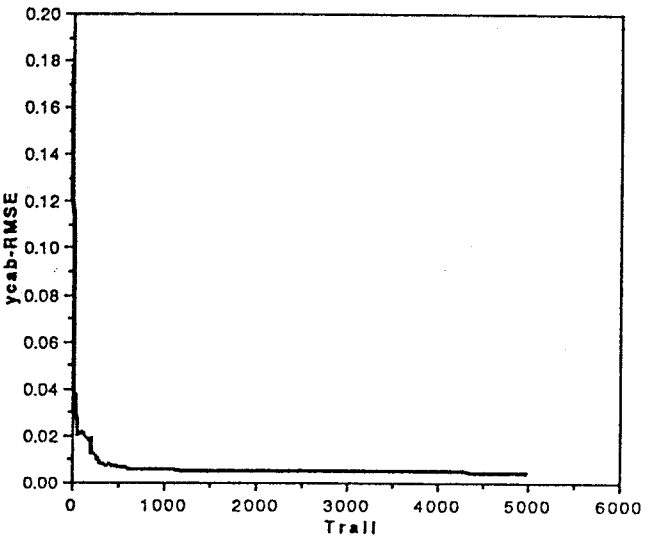


圖3－17 Y_{cab}之RMSE學習曲線圖

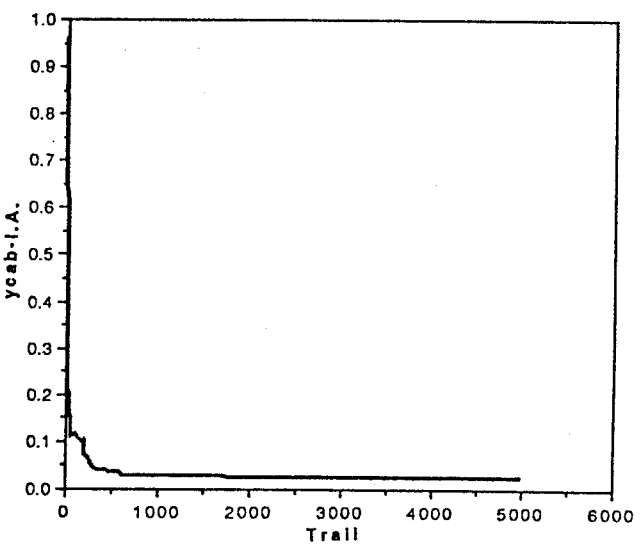


圖3－18 Y_{cab}之I.A.學習曲線圖

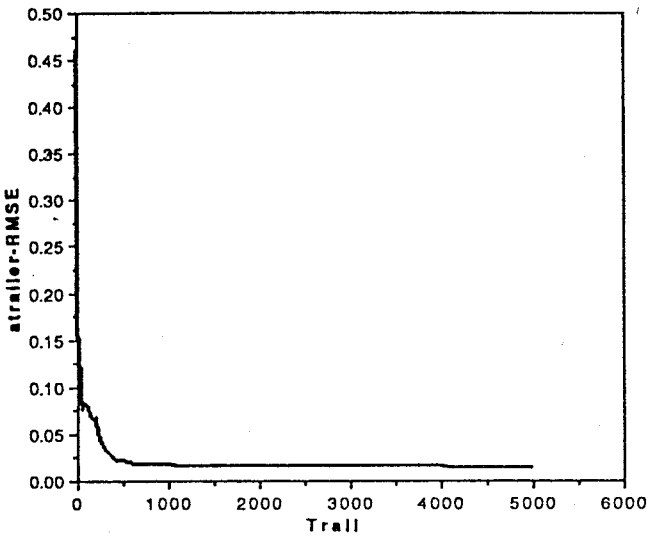


圖3－19 $\theta_{trailer}$之RMSE學習曲線圖

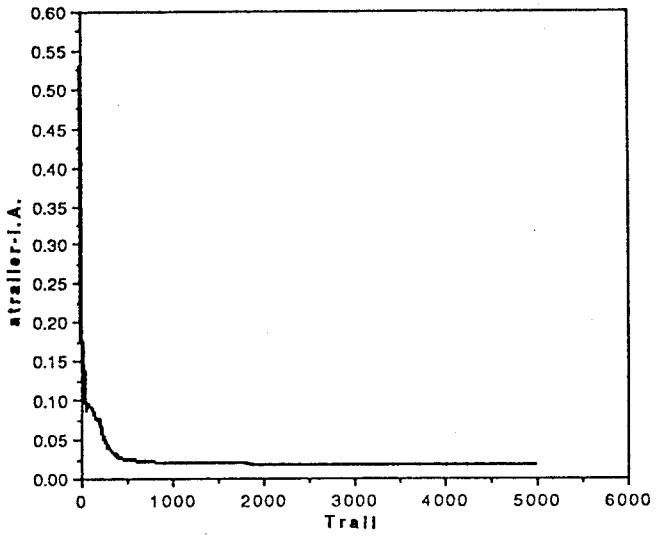


圖3－19 $\theta_{trailer}$之I.A.學習曲線圖

第四章　討論與結論

4－1　結　論

綜合第三章的實驗過程與結果，可以得到下列結論：

(1) 在實際學習時，儘可能以隨機抽樣方式學習，因爲拖車在任一位置，前輪的轉動角度有相當多的組合，單針對某一角度作學習，將無法學習到拖車眞正的行爲。

(2) 學習的目標值的值域將會影響學習的效果。要考慮網路的特性

(3) 在考慮到學習的效果時，可以使用放大誤差值的作法，以增加系統學習的敏感度。

(4) 在學習的曲線中之振盪情形，與所定義的靜態或動態學習率、類神經網路的架構、及樣本資料的範圍有關。如果學習樣本之行爲變化差異(dynamics)巨大時，則權數調整不易。若學習率不適合，則會發生系統振盪。

(5) 動態學習率提供的乃一學習速率之參考，但並非一定是最佳之學習速率；另一較簡化的方式可採用階段式靜態學習率，依系統的測試數、學習之收斂情形，而給與適當的學習速率。

(6) 當使用雙曲線正切函數(tanh)爲輸出致動函數時，必須注意初始權數值，如果初始權數值過大，將使得隱藏單元或輸出單元得到較大之輸入强度，會造成單元之致動函數微分項爲零，系統便有學習停滯的現象。

(7) 以線性函數作致動函數時，要考慮線性函數之特性，線性函數並沒有收斂值。以雙曲線正切函數作致動函數時，要考慮資料的分佈性，如果只使用雙曲線正切函數之上半部或下半部做學習工作，將降低學習的效果。

(8) 同一種學習函數，發生最佳RMSE與I.A.之η值，隨著所給與的初始權數值不同，而產生不同之η值。因爲η值對於初始權數值有較大之敏感度，此與Kolen及Pollack（註六）所得之結果相同。

(9) 初始權數值的給定，確實會影響系統學習的收斂速度，與經過一定測試次數後之最後學習結果（註六）。

(10)學習可分割函數較學習不可分割函數容易。

4－2　貢獻與建議

在經過本論文結果的討論後，我們認爲本論文有下列幾點貢獻可供後續研究人員作一個參考：

(1) 提供一個建立類神經網路的細部過程。本結果可作爲後續有意從事類神經網路的研究人員一個基礎及參考範例。

(2) 使用階段式靜態學習率取代一般使用的靜態（固定）學習率及動態學習率作爲學習的速率。

在本論文的實驗過程中，原本使用動態學習率，後經改用階段式靜態學習率後，學習的效果反而有改善，此點可提供應用學習率時的一個思考方向；唯靜態學習率的值究竟應多少，並沒有一定的規範可供參考，尙賴研究者的經驗與試誤去尋找。

接下來我們提出以下幾點建議：

(一)系統建構：

(1) 依據所要學習的目的物，定義類神經網路的架構。例如在第三章的實驗中，由於輸出角度值分別在$-\pi$與$+\pi$之間，我們嘗試以增益(gain)、線性輸出致動函數、放大誤差值等方式從事學習。

(二)系統學習：

(1) 在學習目標函數前，可先以已知函數作測試，如我們使用的 case1、case2、case3，從測試的過程中吸取經驗。

(2) 學習資料的取樣，必須分佈均勻，且盡可能考慮各種情況，如此才能達到學習正確資訊的功能。

(3) 當學習效果不是很理想時，試著改變系統的參數值，如學習率、初始權數值、誤差分配比例。對於一個系統中，學習率的值對於系統的學習過程會有較大的影響，可以先調整學習率，其次初始權數值、誤差分配比例。

(4) 對於本論文之η，如果想找到一個學習結果較佳的η值，必須以相同的初始權數值做測試，才能得到合理的結果。因爲最佳之η會因初始權數值不同而不同。

(5) 如果使用非線性致動函數作學習，需要考慮誤差由各層處理之分配比例的學習過程。

(三)結果分析：

(1) 對於RMSE與I.A.值之學習曲線圖，必須用所有之測試結果爲分析點，不可取樣分析，以免失眞。分析時以察看學習之I.A.值爲主。因爲RMSE值爲系統輸出值與給與目標值差之均方根，當所給與系統之行爲變化差異(dynamics) 巨大時，學習誤差值自然加大，即RMSE值就變大，但這部份因素可能因樣本數值偏差大所造成，故不採用RMSE值作爲系統學習評價的標準。而I.A.值爲RMSE值除以樣本之標準差，如果我們樣本數值偏差大，則相對的RMSE與樣本之標準差也就加大，由此兩項相除，我們可以得到學習效果RMSE值，與給與樣本之標準差之偏差量，以此爲學習指標，也才能得到眞正的學習效果。

4－3 未來之研究方向

Nguyen 與Widrow 對於拖車倒車入庫的學習，已經有了相當大的貢獻，值得我們參考。在本節中，我們提出另一個架構供作未來研究的方向。

Scott E. Fahlman 與Christian Lebiere在他們的一篇論文（註三）“The Cascade-Correlation Learning Architecture”（以下簡稱CCLA ）中，提供一個通用的類神經網路結果，它突破了以往以固定架構作爲學習結構的方法，而以動態的方式依需要建構隱藏單元，很值得我們探討，因此將這篇文章的摘要性整理放在附錄。

對於 CCLA 之結構，在文獻中將其應用於雙螺旋問題 (two-spirals problem) 及 N 輸入同位(n-input parity) 問題上，所得到的結果較其他種類神經網路模型爲佳。

由於CCLA類神經網路的架構方式可以依問題的複雜性，而建構出不同數量的隱藏單元，而拖車倒車入庫又是一個高度非線性的問題，我們可以試著將CCLA的架構，與本論文已建構好的總體模組連接，來從事拖車倒車入庫的學習工作。

在學習的整合過程中，值得注意的是，原作者所設計之CCLA系統程式，並非與時間具有連續性的關係，也就是說，現在的狀態只與前一個狀態有關，與前一個狀態之前的狀態無關；這與拖車倒車入庫之情形並非完全吻合，因爲拖車之倒車入庫是與時間呈連續性之變化，倒車入庫成功與否，並非只與前一個運動狀態有關，而是與此狀態之前的所有狀態均有關係。

另外，由於CCLA的隱藏單元之建構方式，是一個接著一個的加入網路中，因此所需要的學習時間相當長，這點將會影響系統的學習效率；我們不禁想問，是不是我們在初始狀態時，能先決定由多少個隱藏單元開始學習起？如果決定了由多少個隱藏

單元開始學習，那這些隱藏單元的初始值與初始權數值又應是多少？這幾點將留在以後再做研究。

參考文獻

1. Becker, S. and leCun, Y. "Improving the Convergence of Back-Propagation Learning with Second-Order Methods" Proceedings of the 1988 Connectionist Models Summer School, Morgan Kaufmann.
2. Fahlman, Scott E. "Faster-Learning Variations on Back-Propagation: An Empirical Study" Proceedings of the 1988 Connectionist Models Summer School, Morgan Kaufmann.
3. Fahlman, Scott E. and Christian Lebiere "The Cascade-Correlation Learning Architecture", CMU-CS-90-100, 1990.
4. Graubard, Stephen R. "The Artificial Intelligence Debate False Starts, Real Foundations", The MIT Press, Cambridge, Massachusetts.
5. Jordan, Michael I. "Forward models:Supervised learning with a distal teacher", Occasional Paper # 40.
6. Kolen, John F. and Pollack, Jordan B. "Backpropagation is sensitive to initial condition", Complex system 4(1990) 269-280.
7. Miller, Richard K., CMfgE, CEM "NEURAL NET-WORKS: Implementing Associative Memory Models in neurocomputers".
8. Nguyen, Derrick and Widrow, Bernard "The Truck Backer-Upper: An Example of Self-Learning in Neural Networks", IEEE Journal on Neural Network, Vol. 2, 1989.
9. Plaut, D. C., Nowlan, S. J., and Hinton, G. E. "Experiments on learning by Back-Propagation. "Technical report CMU-CS-86-126, Carnegie-Mellon University, Computer science Dept., Pittsburgh, PA, 1986.
10. Rumehart, David E. and McClelland, Jame L. editors. Parallel Distributed Processing, volume 1, The Mit Press, 1986.
11. Wang, Wei-Hua "A Nonlinear Function Approximator-The Cooperation of Nonparametric Concepts and A Neural Network Approach", PhD dissertation, Ohio state University, 1989.
12. Widrow, Bernard and Hoff, Marcian E. "Adaptive switching circuits". 1960 IRE WESCON Convention Record. New York: IRE, pp. 96-104.
13. Widrow, Bernard, fellow, IEEE, and Lehr, Michael A. "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation", Proceedings of the IEEE, Vol. 78, No.9, September 1990.

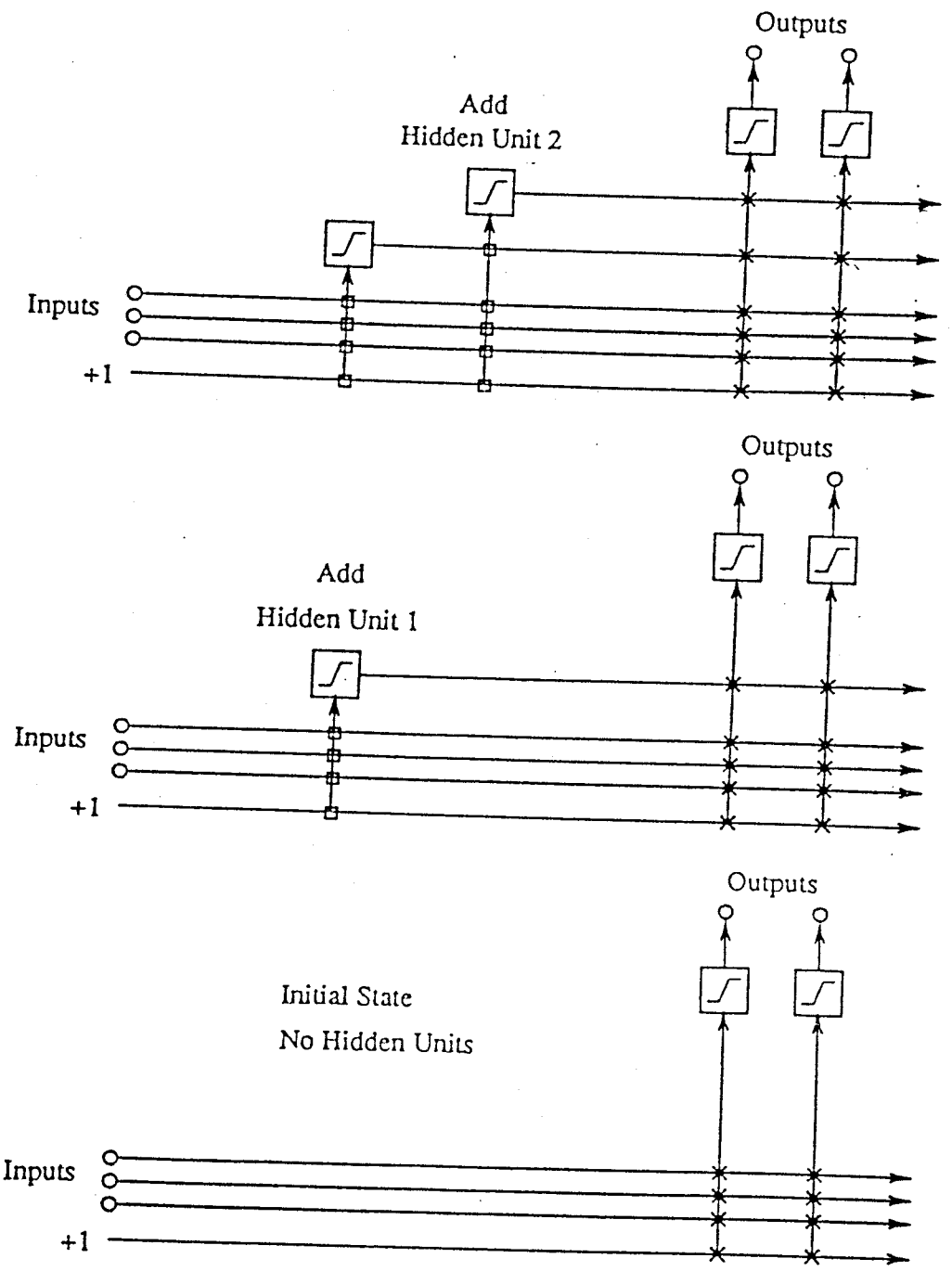
附錄：串聯關係學習結構

對於許多研究類神經網路的工程人員而言，常常會有一個疑問，究竟我們應該在什麼情況的考量下，才能建立一個適合這個問題的類神經網路結構，究竟在這個結構中，單元(unit)與單元間，應該如何連接比較恰當，究竟需要多少個單元才能滿足系統的要求，這些問題沒有一個標準答案；而且常常是依據研究人員本身的知識與經驗來決定要建立那一種的架構；如果有一位研究人員想要知道，何種類神經網路架構對於那類問題較爲恰當，這將是一個很令人困擾的問題。是不是在應用上，我們都需要事先設定所有的類神經網路架構？是不是需要先知道系統中要有多少個單元，才能開始做學習的工作？這個答案並不全然是肯定的。如果在應用類神經網路時，能克服了上列的疑慮，這不就是一個令人稱羨的發現嗎？

Scott E. Fahlman 與Christian Lebiere發表了CCLA架構，它提供了一個通用的類神經網路結構，突破了以往以固定架構作爲學習結構的方法，取而代之的是以一個標準的架構作爲類神經網路發展的模式，本附錄中主要是針對這篇文獻作一個重點的闡述。

CCLA結合了兩個重要的概念：其一爲串連的結構(cascade architecture)，在這個模式中，隱藏單元加入結構的邏輯是在同一時間只加入一個，一旦這個隱藏單元加入結構後，它就不再改變。其二爲學習的演算法(learning algorithm)，CCLA依據這個演算法產生並建構隱藏單元。對每一個新隱藏單元的加入，所用的演算法是，當新單元之輸出值與我們想要減少之殘餘誤差之間的相互關係值最大時，則選擇這個新的隱藏單元。

圖附錄 (1) 描繪出CCLA之類神經網路結構；圖中包含了架構之最初狀態與增加兩個隱藏單元後的狀態，垂直線部份表示此線爲各種輸入之加總，方塊狀接點表示此處之權數值爲固定，X 狀接點表示此處之權數值爲反覆受訓練，可加以更改。下圖爲經簡化的CCLA之類神經網路結構。



圖附錄_(1) CCLA之類神經網路結構（註三）

綜合作者的說明，我們可以將CCLA之特性歸納爲下列幾點：

(1) 開始時，並沒有隱藏單元，只以一些輸入單元與若干輸出單元爲主。輸入單元與輸出單元之個數，依所要學習的系統來定義應有的數量。

(2) 對每一個輸入，都有一個相對應的可調整權數連接至輸出。另外系統有一個對角輸入(bias input)，它的值恆爲＋1。

(3) 可以使用不同的致動函數。文中系統之輸出致動函數爲對稱S 形致動函數(hyperbolic tang-ent) ，它的輸出值限定爲－1.0至＋1.0。

(4) 系統一次只加入一個隱藏單元。每一個剛加入的隱藏單元，都與網路原先存在的輸入單元和隱藏單元有一權數連接。每一個隱藏單元在加入網路後，它的輸入權數就已經固定，只有輸出權數可以反覆的調整。因此，對於整個網路而言，加入一個隱藏單元，就等於加入一層僅一個單元的網路。

(5) 沒有後推(back-propagate)的情形給隱藏單元，故適用其它已知調整單層網路的演算法。

(6) 以快速傳播演算法 (quickprop)（註二）訓練輸出權數。

(7) 當經過內設次數的訓練循環後，如果沒有可觀的誤差值降低，就再執行整個網路一次，並重新計算誤差值，如果此誤差值是可接受的，就停止學習；否則，將新的隱藏單元加入網路，再重新學習，直到誤差值是可接受的或者放棄學習爲止。

接下來，將說明作者是以何種演算法訓練候選單元 (candidate units)，以及如何挑選候選單元，加入網路的規則。

(1) 建立一個後選單元，大約 4 至 8 個，它們的起始權數都是隨機產生各不相同；每一個候選單元都由網路中得到相同的輸入值，且察看到相同的殘餘誤差。候選單元間並沒有相互連接。

(2) 候選單元從網路之外部輸入及已經存在之隱藏單元的輸出得到它的輸入值，它的輸出並未連接至網路，因此不會影響正在學習中的網路的權數值。

(3) 在網路每一次訓練的過程中，都調整候選單元的權數，調整的目的在於將S 值最大化。有關S 值之定義如下：

$$S = \sum_{o} \left| \sum_{p} (V_p - \bar{V})(E_{p,o} - \bar{E}_o) \right|$$

在式中，註腳o 表示網路的輸出，在此點量測誤差值；p 爲訓練樣本(training pattern)指標；V 爲候選單元的值；E_o爲在單元o 所觀察到的殘餘輸出誤差值；V 及E_o是所有訓練樣本之V 與E_o的平均值。

(4) 爲求S 之最大值，計算S 相對於每一候選單元之輸入權數W_i的導數值，即 $\frac{\partial S}{\partial W_i}$ 。計算導數得：

$$\frac{\partial S}{\partial W_i} = \sum_{p,o} \sigma_o (E_{p,o} - \bar{E}_o) f'_p I_{i,p}$$

式中，σ_o爲候選單元的值與輸出單元 o 間的相互關係值的符號；f'_p 是候選單元之致動函數相對於它的輸入總合的訓練樣本p 之導數；$I_{i,p}$對訓練樣本p 而言，候選單元自單元i 所得到的輸入值。

(5) 爲求收斂速度快，使用快速傳播演算法調整權數值。

(6) 當S 值沒有改善時，則加入新的隱藏單元到網路中，並且固定它的輸入權數值，繼續訓練網路。同時重新建立候選單元集，回到(1) ，重新開始。

在以上的兩種說明中，必須特別注意的是，當輸出層的權數在調整時，網路中的其他權數將保持固定；當候選單元的權數在調整時，網路中沒有任何權數值被更改。

相對於目前正在使用的類神經網路訓練演算法，作者認爲CCLA具有下列的優點：

(1) 不需要事先便決定網路的大小、深度與連接的樣式。它可以自動產生合理大小的網路，同時單元間也可以使用不同型式的非線性函數作爲

致動函數。

(2) 在CCLA中，每一個單元只著手解決一個固定的問題，因此學習的速度很快。學習的時間大約爲NlogN，其中 N 爲解決這個問題最後所需要的隱藏單元數。

(3) CCLA可以建立深層的網路，而不會有像後推網路因層數加大便大大降低效能的情形。

(4) 在同一時間僅訓練一層的權數，其他在網路中的權數並沒有改變，因此可以儲存它們的值。

(5) 網路不需要將誤差值後推，權數僅向一個方向傳送，除去了後推可能產生的誤差。

(6) 候選單元間並沒有相互連接。在同一時間，每一個候選單元都由網路中得到相同的輸入與誤差信號，這個限制有助於平行處理。

由以上的結果，可以發現CCLA與我們傳統上所建構的類神網路模型有一些差異，至於其差異處，可以歸納爲下面幾點：

(1) 傳統類神經網路的架構於網路開始學習時，便已經建構完成；而CCLA採用動態的方式，依系統當時的情形加入隱藏單元，並非一開始網路便已建構好。

(2) 傳統類神經網路並沒有自動增加隱藏單元的功能，而CCLA有一套法則，決定新單元應何時加入網路。

(3) 傳統類神經網路在網路中的權數均可調整，而CCLA於加入新單元後，新單元輸入部份所接連的權數值便固定而輸出部份的權數值可調整。

(4) 使用後選單元集，以爬山式(hill-climbing)的方法訓練新單元，使候選單元與殘餘誤差的相互關係最大。

(5) CCLA在同一時間只訓練一個單元，而不是整個類神經網路，與傳統類神經網路不同。

(6) 前導式模型有誤差的後推計算，而CCLA沒有誤差信號的後推。

(7) 前導式模型隱藏單元間並沒有權數相接，而CCLA隱藏單元間有權數相接，且不管是否有隱藏單元，輸入單元與輸出單元間均有直接的權數相連。