模糊類神經網路學習秘笈

（Tips for Learning Fuzzy-Neural Network）

南台電機 趙春棠 （STUT Chun-Tang Chao） 2008. 9

■ 緣起：“模糊類神經網路”（Fuzzy-Neural Network, FNN）是一門結合“模糊（Fuzzy）”以及“類神經網路（Neural Network）”的技術。可惜在一般有關「模糊理論、控制」或「類神經網路」等方面的課程中，授課教師時常沒有機會好好地詳加說明。另一方面，在 Matlab 中，雖然有 ANFIS 等套裝軟體，可是如何自行寫程式來實現它，卻是個問題。基於以上兩點，寫了這篇短文，希望對苦惱於這方面教學或學習的老師或同學，有所幫助。

■ From “Fuzzy Logic” to FNN

修過“模糊理論（Fuzzy）”的同學，一定知道如何：

1. 將輸入「模糊化」，
2. 對每一「法則」進行「模糊推論」，
3. 「解模糊化」得最後的輸出。

圖 1 是兩個模糊法則的一個模糊推論系統簡例，請注意比較其與一般“模糊理論（Fuzzy）”中的系統之不同所在。比較後，您將會發現，圖 1 所採用的方法，是不是比較簡單？以下列出三個特點：

1. Gaussian Membership Function: 傳統的「三角」隸屬函數，被「高斯」函數所取代。
2. Product Inference Rule: 傳統「Max」、「Min」不見了，用「乘積」取代了。
3. Singleton Fuzzifer: 傳統法則輸出的隸屬函數，被簡單的 Singleton（圖中的 w1,w2）函數所取代。

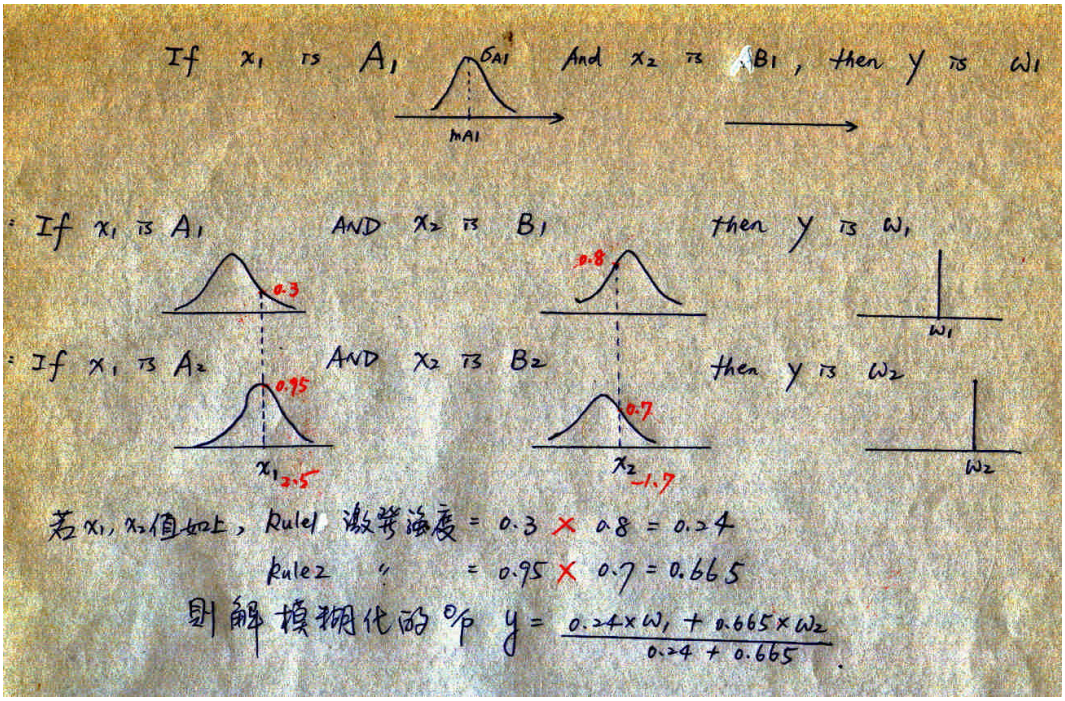


Fig. 1

有了以上的概念後，將圖 1 改畫成圖 2 的型式，就是一個 FNN 了！了解了嗎？而 FNN 比 Fuzzy

Logic 具備的優點就是它的每一個法則是活的，是可以經由學習而調整的。以圖 1 爲例，其中每一個高斯函數是可以變動的（藉由調整 期望值、變異數）；而 w1, w2 也是可以經過學習而調整的。當然，有關「學習」的理論基礎，就是套用「類神經網路」的理論了。至此，讀者應該可以了解爲何稱爲「模糊」「類神經網路」了！

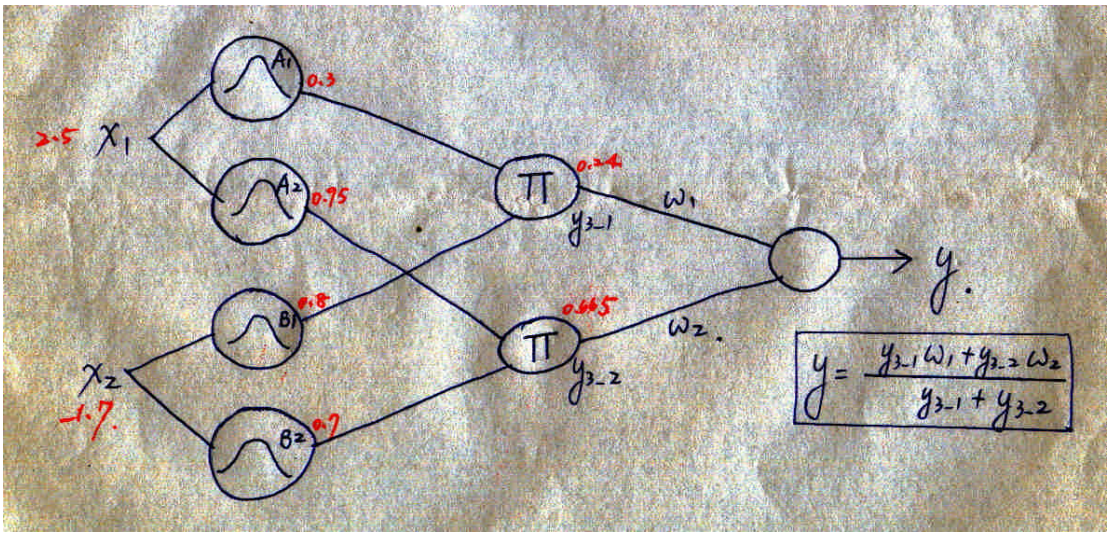


Fig. 2

■ 學習範例：以下是一個「非線性」函數，我們希望用一個 FNN 來學習出這個函數。這是在文獻[1]中的一個範例，筆者過去是撰寫 C 程式語言來實現，此次則利用 Matlab 來實現之。

$$y = f(x_1, x_2) = \frac{\sin(\pi x_2)}{2 + \sin(\pi x_1)} \qquad -1 <= x1 <= 1, \quad 0 <= x2 <= 1$$

程式： fnn2_3_3_9_1.m

■ FNN Structure:

有類神經網路學習經驗的工程人員應該都知道，類神經的網路架構與所要學習的系統息息相關，結構太大或是太小，都不適宜。結構太大，可能造成過度學習，結果反而不好，此時好比使用太多階的多項式來完成曲線擬合（curve fitting）問題一樣；另一方面，如果結構太小，系統參數不夠多，則學習的結果勢必不夠精準。圖 3 是本文中所採用的架構，簡單來說，x1 變數使用 3 個 term nodes，x2 變數使用 3 個 term nodes，如此一來，這些 term nodes 彼此連結，就產生 3×3=9 rule nodes。這種架構，也類似 ANFIS 所採用的架構。

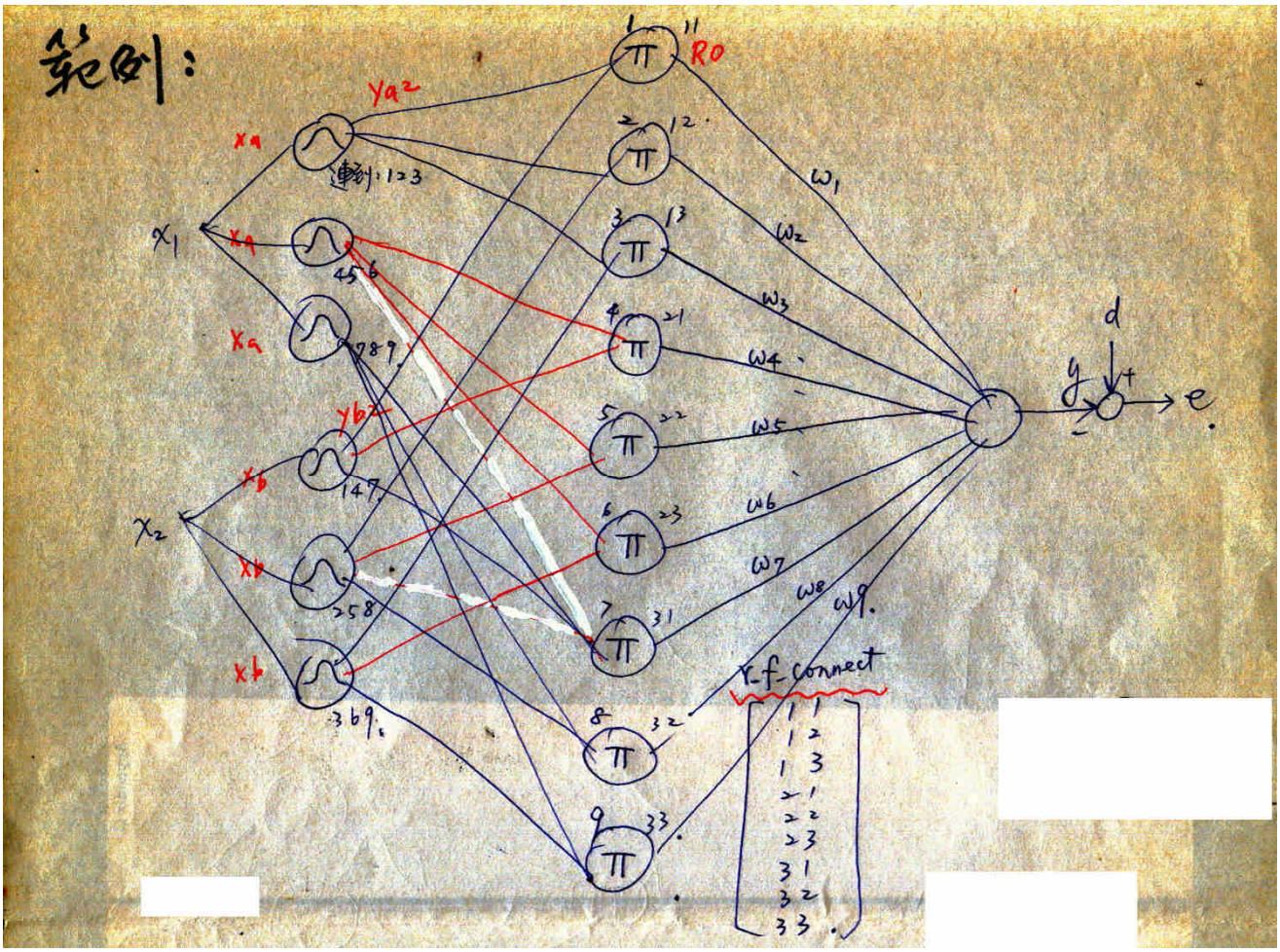


Fig. 3

另外一種有名的架構，見於 Li-Xin Wang 教授的著作，或[2]中亦可見，這種架構的特性是

Fuzzy 法則數目 = 每一輸入變數的 term nodes 數目

如圖 4 所示，如此一來，FNN 的連接就簡單得多，不過 … ，這部份就先暫不討論了！

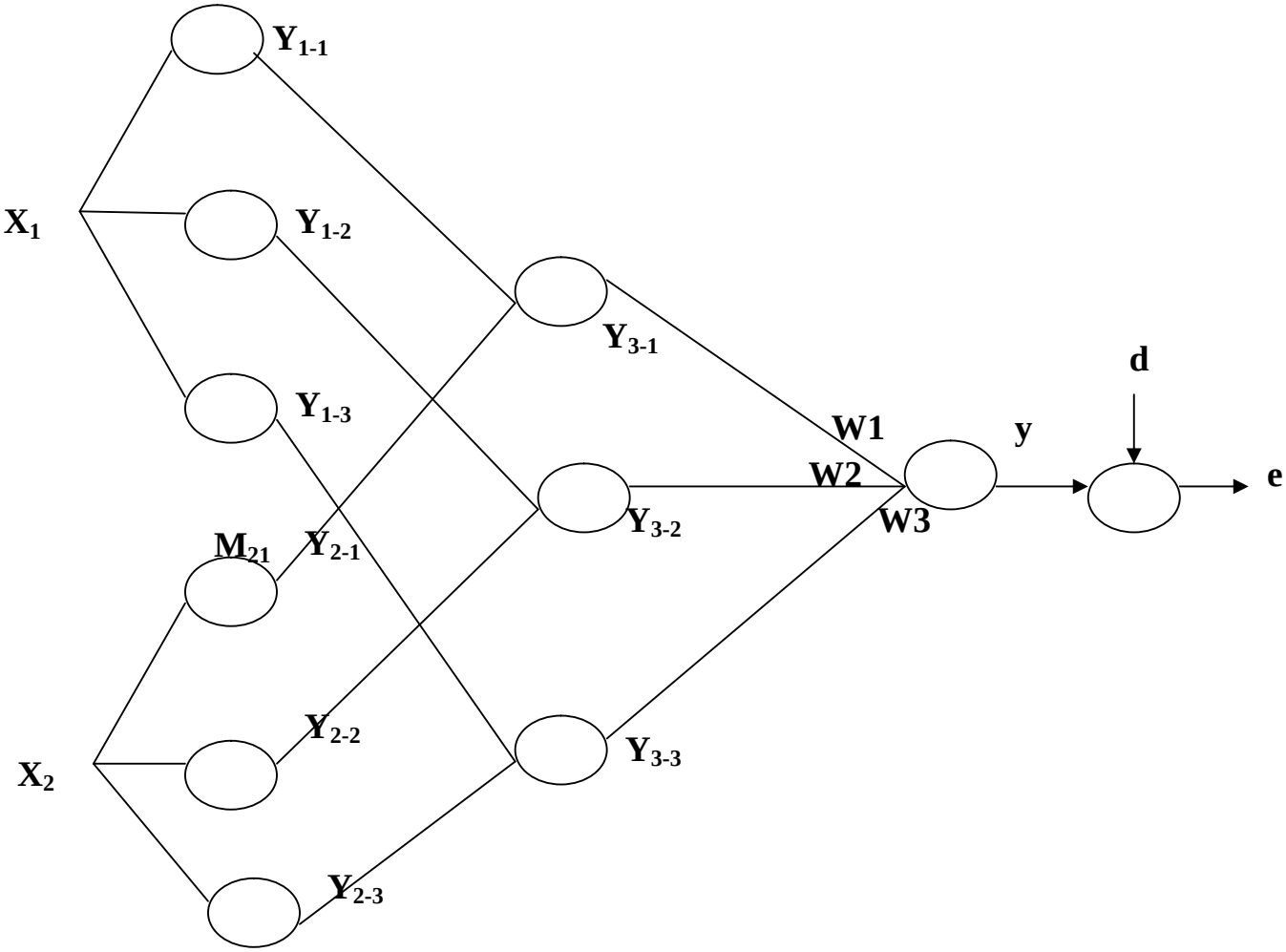


Fig. 4

【蔡智倫 先生繪圖，特此致謝！ ^_^】

■ FNN Initialization and Forward Operation

建立好 FNN 架構後，接下來就是先給定 FNN 參數的初值。以下設定方式可以參考：

1. 根據輸入變數 x1, x2 的範圍，概略設定其 term nodes 之期望值、變異數初值
   （本例： -1<= x1 <= 1, 0<= x2 <= 1）
2. 考慮隸屬度爲 1 的輸入值，代入學習範例函數，將輸出作爲 FNN 輸出層的權重值 wn
   （本例： y(0.6,0.75) = 0.2396 ← 此值作爲 w1 的初值）

筆者將本例 FNN 初值設定，以圖形方式，顯示於圖 5，程式 fnn2_3_3_9_1.m 就是依這些參數進行初始化的。可以想像得到，好的初值設定，對於未來 FNN 的學習調整，會有事半功倍的效果，讀者可以試著更改初值，做實驗測試。

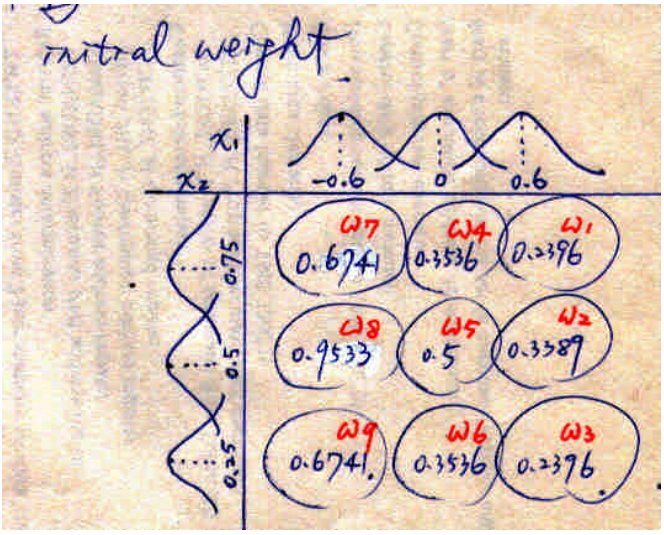


圖 5

FNN 的特點，就是能一步一步地修正圖 5 的這些參數，使 FNN 最終能學習出整個系統。所以接下來我們必須準備 Training Patterns（訓練樣本），在本例中，我們利用「均勻分布亂數」的概念，在輸入變數 x1 及 x2 的輸入範圍內讓電腦自由地選定亂數，共取了 247 筆（x1 及 x2 值代入以上「學習範例」，即得理想輸出 d）。

此時讀者應自行思考，在圖 3 的 FNN 架構中，每當輸入一筆變數 x1 及 x2 的值，您是否有能力自行寫程式，計算出此時的輸出 y（即 Forward Operation）？在程式 fnn2_3_3_9_1.m 中，讀者可以發現，有關 FNN 架構的設定，還沒有達到 100%全自動，也就是不能讓 user 隨意地設定 FNN 架構。不過也正因爲如此，程式比較簡短，比較容易讀懂。

■ Back-Propagation（倒傳遞）

學過類神經網路的讀者，對於 Back-Propagation 應該不陌生，在每一筆 Training Pattern 輸入 FNN 後，隨即就利用 Back-Propagation，對每一參數進行修正調整（adjustment），以下簡單說明原理。

首先定義誤差函數 E（error function）來表現網路的學習品質，如下所式：

$$E = \frac{1}{2}(d - y)^2 \qquad (1)$$

其中 d 爲希望的輸出（desired output）。有了誤差函數 E，就可以利用倒傳遞演算法推導網路加權值的修正公式，假設 $w(i, j)$ 是網路中的一個加權值，則修正法則如下：

$$w(i, j)[k+1] = w(i, j)[k] - \eta \frac{\partial E}{\partial w(i, j)} + \alpha \Delta w(i, j)[k] \qquad (2)$$

其中 η 爲學習率（learning rate），而 α 爲慣性因子（momentum parameter）。上式中，讀者不妨先令 α =0，如此也是可以進行參數修正的，只是學習收斂會慢些。有了以上的觀念，我們發現只要有辦法推導出 $\frac{\partial E}{\partial w(i, j)}$，再套用式(2)，問題就解決了，以下推導依此原則。

Case 1: 輸出層 wk 的倒傳遞修正

以下以修正 w_2 爲例，列式推導，一旦導出，自能推得 w_k 的修正“通式”。所需要的數學基礎，僅僅是“偏微分”以及“chain-rule”技巧，讀者一定要試著自行推導，首先：

$$y=\frac{y_{3-1}w_1+y_{3-2}w_2+\ldots+y_{3-8}w_8+y_{3-9}w_9}{\left(y_{3-1}+y_{3-2}+\ldots+y_{3-8}+y_{3-9}\right)} \tag{3}$$

故 $$\frac{\partial \mathrm{E}}{\partial w_2}=\frac{\partial E}{\partial y}\times\frac{\partial y}{\partial w_2}=\ldots=\left[-\left(d-y\right)\right]\times\frac{y_{3-2}}{\left(y_{3-1}+y_{3-2}+\ldots+y_{3-9}\right)} \tag{4}$$

有了以上結果，我們可以推廣得：

$$\frac{\partial \mathrm{E}}{\partial w_k}=\frac{\partial E}{\partial y}\times\frac{\partial y}{\partial w_k}=\ldots=\left[-\left(d-y\right)\right]\times\frac{y_{3-k}}{\left(y_{3-1}+y_{3-2}+\ldots+y_{3-9}\right)} \tag{5}$$

一般教科書，都是寫成以上的通式，而我們在學習參數修正理論的時候，應該拿支筆，拿張紙，自行針對某一變數來推導才是，否則永遠無法真正理解數學式子所代表的涵義。

Case 2: Term Node 層，平均值 m_{ij} 的倒傳遞修正

以下以修正 m_{21}（變數 x2 的第一個 term node）爲例。首先

$$y_{2-ij}=e^{-\left(\frac{x_i-m_{ij}}{\sigma_{ij}}\right)^2} \tag{6}$$

故 $$\frac{\partial E}{\partial \mathrm{m}_{21}}=\frac{\partial E}{\partial y_{2-1}}\times\frac{\partial y_{2-1}}{\partial \mathrm{m}_{21}}=\left(\frac{\partial E}{\partial y}\times\frac{\partial y}{\partial y_{2-1}}\right)\times\frac{\partial y_{2-1}}{\partial \mathrm{m}_{21}} \tag{7}$$

$=\cdots$（略）

$$=\frac{-(d-y)}{(y_{3-1}+\cdots+y_{3-9})}\times\left[(w_1-y)\times y_{3-1}+(w_4-y)\times y_{3-4}+(w_7-y)\times y_{3-7}\right]\times\frac{2(x_2-\mathrm{m}_{21})}{\sigma_{21}^2} \tag{8}$$

觀察上式，我們發現中間中括弧所包含的項，包含了 y_{3-1}, y_{3-4}, y_{3-7}，您可知爲什麼？因爲這些都是「變數 x2 的第一個 term node」所連接到的 rule node，所造成的輸出呀！這可是很重要的「秘密」喔！如果您懂了，這時才真的了解 Back-Propagation 的意涵喔！對了，請試著寫出式(8)的通式吧！哈，一般教科書，論文，就是把它寫成通式啦！所以粗心的讀者，始終看不懂囉！

Case 3: Term Node 層，標準差 σ_{ij} 的倒傳遞修正

以下以修正σ_{21}（變數 x2 的第一個 term node）爲例。

$$\frac{\partial y_{2-1}}{\partial \sigma_{21}} = \cdots$$

$$\frac{-(d-y)}{(y_{3-1}+\cdots+y_{3-9})} \times \left[(w_1-y)\times y_{3-1}+(w_4-y)\times y_{3-4}+(w_7-y)\times y_{3-7}\right] \times \frac{2(x_2-\mathrm{m}_{21})^2}{\sigma_{21}^2} \qquad (9)$$

比較式(8)、(9)，我們可以發現，兩者的差別僅在於後者最後一項的分子，多取了平方。關於此點，其實不難，讀者只需比較 $\frac{\partial y_{2-1}}{\partial \mathrm{m}_{21}}$ 與 $\frac{\partial y_{2-1}}{\partial \sigma_{21}}$ 就可以發現其間的差異了！

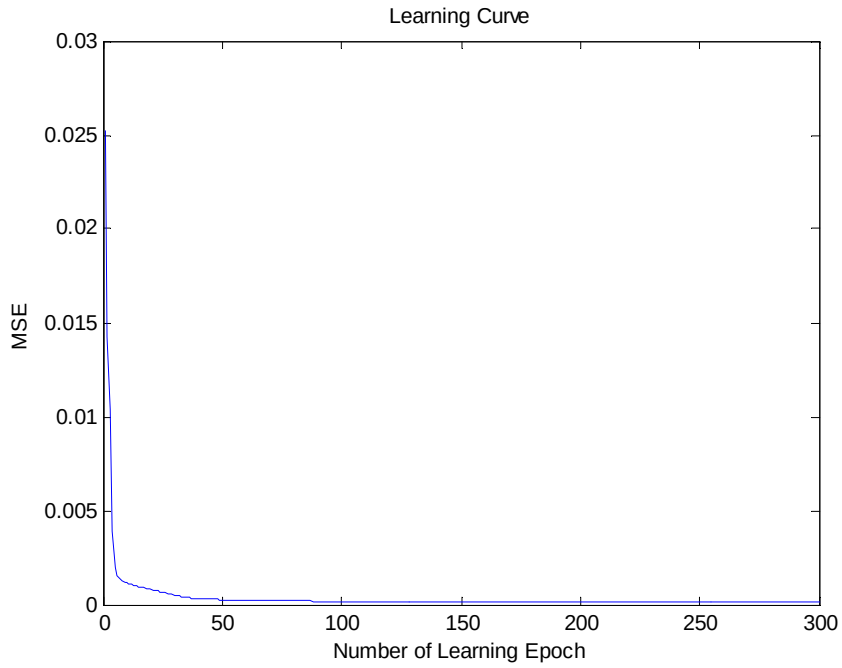
■ 程式 fnn2_3_3_9_1.m 執行結果：

epoch=1, MSE=0.025200.
epoch=2, MSE=0.014252.
epoch=3, MSE=0.010382.
epoch=4, MSE=0.003886.
epoch=5, MSE=0.001947.
epoch=6, MSE=0.001558.
……（略）……
epoch=295, MSE=0.000090.
epoch=296, MSE=0.000090.
epoch=297, MSE=0.000090.
epoch=298, MSE=0.000089.
epoch=299, MSE=0.000089.
epoch=300, MSE=0.000089.

Learning Results O/P: 【以下是學習結束的每個參數值喔！】

mx1(1)=0.415389, stdx1(1)=0.282738
mx1(2)=-0.108031, stdx1(2)=0.771812
mx1(3)=-0.456439, stdx1(3)=0.243602
mx2(1)=1.224470, stdx2(1)=0.418209
mx2(2)=0.517610, stdx2(2)=0.397358
mx2(3)=-0.205133, stdx2(3)=0.404506

w(1)=-0.071464
w(2)=0.256671
w(3)=-0.044561
w(4)=-0.147919
w(5)=0.558083
w(6)=-0.093846
w(7)=-0.385213
w(8)=1.522813
w(9)=-0.215161



以上已經將 FNN 的學習要點，和盤托出，希望有興趣的讀者，好好繼續研究。fnn2_3_3_9_1.m 程式，是慢慢一行一行長出來的，重要的地方，筆者都加了註解，看之前，先自己想一下如果自己寫程式，會遇到那些問題？如此應該可以加速看懂程式喔！　　祝：

Happy Learning!

■ References:

1. C.T. Chao and C.C. Teng*, “Implementation of a fuzzy inference system using a normalized fuzzy neural network,” Fuzzy Sets and Systems, vol.75, no.1, pp. 17-31, October 1995. (SCI)
2. C.T. Chao, Y.J. Chen and C.C. Teng*, “Simplification of fuzzy-neural systems using similarity analysis,” IEEE Trans. Systems Man Cybernetic Part B: Cybernetics, vol.26, no.2, pp. 344-354, April 1996. (SCI)
3. **J.-S. R. Jang（清大 張智星 教授）,** `` ANFIS: Adaptive-Network-based Fuzzy Inference Systems," IEEE Trans. on Systems, Man, and Cybernetics, vol. 23, pp. 665-685, May 1993.