逆傳遞法則前饋式類神經網路視窗程式設計

Windows Programming of Feedforward Neural Network Using Backpropagation Algorithms

林顯聖　羅國彰　莊興裕

國立臺北科技大學化學工程系, 臺北, 106

摘　要

近年來，類神經網路的蓬勃發展 ，在許多領域都有應用的實例 。藉由類神經網路對實驗數據做非線性回歸擬合 ，可以進行如在天氣預報 、文字辨識等工作 ，用途廣泛 。不過由於大部分類神網路軟體是屬於商業軟體 ，其售價對初學者而 言是一筆沉重的負擔 。因此本研究希望開發一個功能較簡易的類神經網路軟體 ，提供初學者學習時使用 。

本研究使用 Visual Basic 進行程式開發 ，使用逆傳遞法中 Widrow&Hoff 所提出的 Delta 法則做爲參數訓練的方法 。本研究所開發的是簡易型前饋式類神經網路軟體 ，初步開發的功能隱藏層只有一層 ，並且隱藏層的轉換函數可以使用的只有 Hyperbolic Tangent Sigmoid 函數及 Log-Sigmoid 函數，而輸出層的轉換函數則只能使用 Linear 函數。即便如此 ，利用此簡易的類神經網路軟體進行 各種不同的數據測試 ，結果顯示其仍有相當之實用性 。

一、緒論

類神經網路的研究自 40 年代已有不少的成果，早在 1943 年，心理學家 McCulloch 和數學家 Pitts[1]合作提出固定權重的類神經的數學模型(稱之 MP 模型)，雖僅限於邏輯運算而並不具備學習能力，但卻從此開創了神經學理論研究的時代，後來陸續有不少學者提出相關研究。

1949 年，Hebb[2]提出了改變神經元連接強度的 Hebbian 學習法則，利用此學習法則來解釋心理學的實驗結果，至今仍在各種類神經網路模型中佔有重要的地位。而人工智慧網路系統的研究則是在 50 年代末 60 年代初開始。1958 年 Rosenblatt[3,4]首次提出可調整權重具備感知機(Perception)的類神經元，使網路具有接受刺激的能力 ，通常用來作爲分類器 (Classifier)[5]，最主要是用來模擬動物以及人腦的學習能力，1960 年 Widrow 及 Hoff[6]提出了自適應元件 ADALINE (adaptive linear element)，它是連續取值的線性網路，最主要是應用於訊號的處理方面，不過這與當時佔主導地位以順序離散符號推理爲基本特徵的 AI 途徑完全不同 ，因此引起很大的爭議 。1969 年 Minsky 及 Papert[7]提出"perceptron"中提到了類神經網路若加入隱藏層，其學習法則之運算將較爲困難，且由數學上證明得知，感知機無法針對 exclusive OR 的問題作處理，因此導致許多研究類神經網路的學者大受影響，而陷入了低潮。直到 1982 年，美國加州工學院物理學家 Hopfield[8]提出了 HNN 模型，引入了能量函數的概念，給予類神經網路是否穩定的依據，使得類神經網路又活絡了起來，同時也開拓了類神經網路用於聯想記憶和最佳化計算的新途徑，而於 80 年代之中，也有不少學者提出相關的研究。

1982 年，Kohonen[9]提出了自我組織特徵映射網路(self-organizing feature mapping)，爲競爭式學習法則的一種網路，主要用於模式識別、聲納的訊號識別等，1986 年，Rumelhart 和 PDP 研究群 [10] 提出逆傳遞網路 (backpropagation neural network)，至今仍爲最廣泛應用的類神經網路。

1988 年 Kosko[11] 提出雙向聯想記憶 (bidirectional associative memory)，它是最早用於學習的網路；1989 年 Moody 和 Darken[12] 提出了根基函數網路 (radial basis function network)；Rumelhart 和 PDP 於 1986 年所提出之逆傳遞網路已可解決 Minsky 及 Papert 提出無法針對互斥或(exclusive OR)的問題，因此使得類神經網路更加蓬勃發展。至此類神經網路逐漸受到重視與應用 。隨著電腦運算能力提升，類神經網路得以開始展現強大的功能，許多的研究開始以神經網路代替傳統的統計技術，利用其學習、推理、記憶等等能力架構系統模型，以達到估計、預測的作用。

二、前饋式類神經網路系統

類神經網路是由大量的簡單處理單元或

稱神經元(如圖 1 所示)所組成的一個複雜網路，其基本原理是模擬人類腦細胞對外界訊號處理之方式，它具有如同人腦般的學習能力。類神經網路的運作模式，即接收外界輸入的信號並傳入網路中，經由配重值(weight)、門檻值(basis)的加權處理會產生出一活化能量(activation)，接著將活化能量再由網路中選定的轉換函數做轉換後，可得一轉換值，並傳送至下一層神經元，經過不斷的轉換與傳送之後，可得到類神經網路的輸出值。

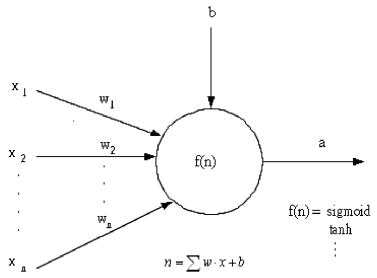

圖 1 神經元基礎單元

前饋式類神經網路的基本結構如圖 2 所示，可分成三個部份：

1. 輸入層：一般輸入層可分爲兩種形式，第一種輸入層中的神經元有轉換函數、內部門檻值及配重值；第二種輸入層中的神經元沒有轉換函數，沒有內部門檻值，也沒有配重值，因此輸出值就是輸入變數值。而本研究是採用第二種。
2. 隱藏層(可有可無)：用來接受輸入層神經元訊號以及輸出經由轉換所得之轉換值，通常可能是一層、二層甚至是多層，可依問題的複雜度作調整，而神經元的數目並沒有明確的方法可以決定，多半是以測試的方式來決定其最佳的數目。一般在隱藏層中，通常是以非線性轉換函數來做計算。
3. 輸出層：用來接受隱藏層的轉換值，通常神經元的數目與輸出值的變數個數相同，轉換函數的選擇將影響得到的輸 出值。

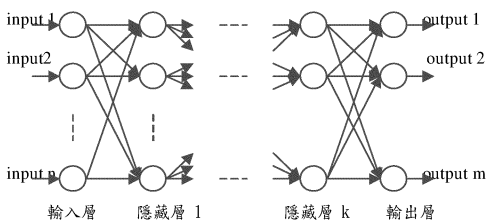

圖 2 前饋式類神經網路架構

由以上敘述可知，類神經網路神經元的數學運算流程可用以下三個步驟來表示：

步驟一：計算神經元的活化能

$$n_i^m = \sum_{j=1}^{p^{m-1}} w_{i,j}^m \cdot x_j^m + b_i^m \qquad (1)$$

步驟二：計算神經元的輸出值

$$a_i^m = f(n_i^m) \qquad (2)$$

步驟三：輸出值等於下一層神經元輸入值

$$x_j^{m+1} = a_i^m, \quad i = j \qquad (3)$$

式中上標 m 代表第 m 層神經元，下標 i 代表第 i 個神經元，而 j 則是神經元所接收的第 j 個輸入值，p^{m-1} 是第 $m-1$ 層神經元的個數。而 $f(n_i^m)$ 則依選擇的轉換函數而定。

在本研究的程式中隱藏層 可使用的轉換函數有雙曲正切 S 型函數(Hyperbolic Tangent Sigmoid)以及對數 S 型函數(Log-Sigmoid)，而輸出層則是線性函數 (Linear)，其計算式分別表示如下：

(1) Hyperbolic Tangent Sigmoid 函數：

$$\tan sig(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad (4)$$

(2) Log-Sigmoid 函數：

$$\log sig(x) = \frac{1}{(1+e^{-x})} \qquad (5)$$

(3) Linear 函數：

$$purelin(x) = x \qquad (6)$$

由於是前饋式網路架構，所以代入網路中的輸入值會一層一層往後傳遞，並在每一層的神經元內作上述的運算步驟，最後就可得到我們想要的對應輸出值了。

三、逆傳遞學習法則

由於我們所選擇的初始參數 (配重值及門檻值)未必能充分表達輸入與輸出的關係 ，所以我們需要使用學習法則來 修正神經元間的配重值及門檻值，使得網路輸出值的誤差函數(MSE)能達到最小值。而本研究使用的是修改的「最陡坡降法」來調整配重值及門檻值。

此學習 法則 對於配重值與門閥值的修改，如以下所示：

(1) 配重值修正公式：

$$w_{i,j}^m(k) = w_{i,j}^m(k-1) - (1-mc)\cdot lr \cdot s_i^m \cdot x_j^{m-1} + mc \cdot \Delta w_{i,j}^m(k-1) \qquad (7)$$

(2) 門閥值修正公式：

$$b_i^m(k) = b_i^m(k-1) - (1-mc)\cdot lr \cdot s_i^m + mc \cdot \Delta b_i^m(k-1) \qquad (8)$$

式中 $w_{i,j}^m(k)$ 是指第 m 層第 i 個神經元與前一層第 j 個神經元間的配重值，$b_i^m(k)$ 則是第 m 層第 i 個神經元的門檻值，而括弧內的 k 則是第 k 次改變後的值，另外 lr 是學習速率，mc 是

動量係數，s_i^m 是靈敏度。

隱藏層與輸出層的靈敏度分別有不同計算方式，在隱藏層因爲沒有目標值可計算誤差，所以必須先算出輸出層的靈敏度，然後再一層一層往回求出各層神經元的靈敏度，也因爲它的這個特性，所以此法被稱爲逆傳遞學習法則。

輸出層與隱藏層的靈敏度計算公式分別表示如下：

(1) 輸出層靈敏度計算：

$$s_i^O = -2(t_i - a_i^O)\frac{\partial f^O(n_i^O)}{\partial n_i^O} \tag{9}$$

(2) 隱藏層靈敏度計算：

$$s_j^m = s_i^{m+1} w_{i,j}^{m+1} \frac{\partial f^m(n_j^m)}{\partial n_j^m} \tag{10}$$

式(9)中上標O表示輸出層的意思，而t_i則是該組輸入值的相對目標值。等號右邊的微分項依轉換函數的選擇而有所不同，分別如下所示：

(1) Hyperbolic Tangent Sigmoid 函數微分：

$$\frac{\partial(\tan sig(n))}{\partial n} = 1 - \tan sig^2(n) \tag{11}$$

(2) Log-Sigmoid 函數微分：

$$\frac{\partial(\log sig(n))}{\partial n} = \log sig(n)(1 - \log sig(n)) \tag{12}$$

(3) Linear 函數微分：

$$\frac{\partial(purelin(n))}{\partial n} = 1 \tag{13}$$

四、軟體介紹

我們所開發的前饋式類神經網路視窗軟體，輸入層不做任何計算，而隱藏層有一層，且轉換函數有 Hyperbolic Tangent Sigmoid 函數及 Log-Sigmoid 函數兩種，另外，輸出層轉換函數則只使用 Linear 函數一種。

(一) 程式訓練流程說明：

(1) 設定網路隱藏層之神經元數目及隱藏層轉換函數。
(2) 設定學習速率與動量係數及輸出層轉換函數。
(3) 輸入訓練資料。
(4) 以均勻隨機亂數產生網路各個參數值。
(5) 計算第 I 組數據之網路輸出變數值。
(6) 計算第 I 組數據輸出值與目標值的誤差值。
(7) 計算靈敏度。
(8) 根據第 I 組數據更新各神經元的配重值與門檻值。
(9) 重複步驟(5)到(8)依次從第一組數據訓練至最後一組數據完成爲一個訓練回合。
(10) 重複第(9)步直到收斂。

程式訓練流程圖如圖 3 所示。

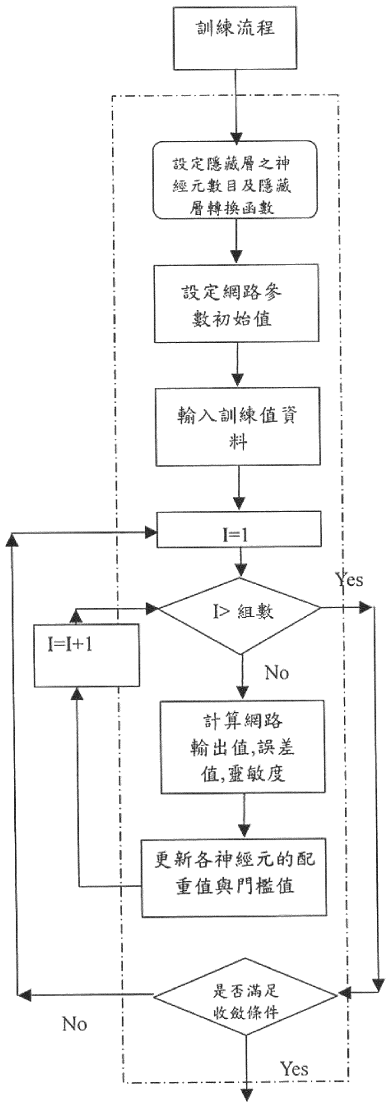

圖 3 訓練流程圖

(二) 軟體視窗說明：視窗介面如圖 4 所示

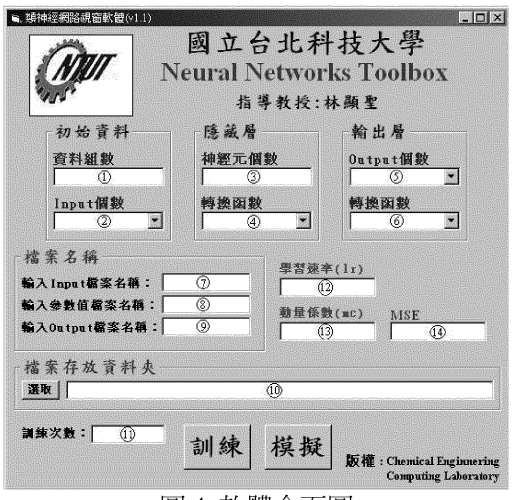

圖 4 軟體介面圖

①鍵入資料對的組數
②選取每組資料對中 input 的個數
③鍵入隱藏層的神經元個數
④選取隱藏層的轉換函數
⑤選取輸出層的神經元個數，也就是每組資料對中 output 的個數
⑥選取輸出層的轉換函數
⑦鍵入儲存資料對檔案的檔名
⑧鍵入儲存參數值檔案的檔名
⑨鍵入程式運算完成後，儲存數據的檔案名稱
⑩鍵入⑦⑧⑨三個檔案的存檔路徑，也可按左邊的 選取 鈕選擇，按下 選取 會出現如圖 5 的視窗，視窗中位置(1)可以選擇磁碟機，位置(2)可以選擇資料夾，而位置(3)是用來確認檔案是否在此資料夾中，最後位置(4)則是顯示選取的資料夾的完整路徑
⑪輸入希望訓練的次數
⑫輸入學習速率
⑬輸入動量係數
⑭顯示輸出值的 MSE 值

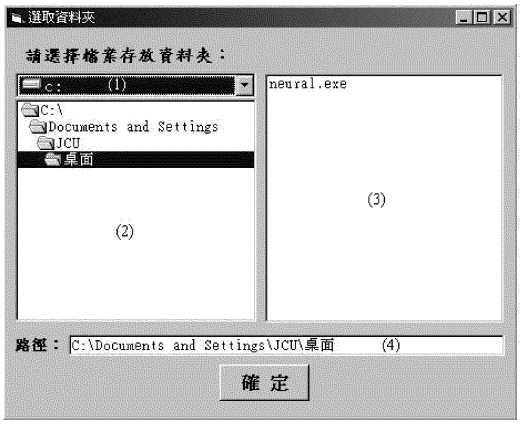

圖 5 選取檔案存放路徑

五、數據測試

範例一：(一個輸入數據產生一個輸出值)
在此範例中，我們用式(14)這個公式來測試。

$$Y = \sin(2\pi X) \qquad (14)$$

首先將 X 範圍-1~1 分成 100 等分，可得 101 個點，再用式(14)求出這 101 個 X 值的相對目標值(Y)，可以得到 101 組訓練對。另外再將 X 範圍-1~1 分成 200 等分，可得 201 個 X 值，扣除訓練的 101 個，剩下的 100 個作爲測試數據。

另外，我們在訓練時所使用的各項條件如下列所示：

1. 隱藏層轉換函數--TANSIG
2. 輸出層轉換函數--PURELIN
3. 學習速率--0.0009
4. 動量係數--0.5

而訓練結果的 MSE 值如表 1 及圖 6 所示。

隱藏層神經元數目	訓練數據 MSE
1	3.60121E-01
2	2.90717E-01
3	4.31332E-03
4	1.38101E-03
5	2.59012E-05
6	1.42388E-05
7	1.26377E-05
8	9.36899E-06
9	8.98548E-06

表 1 範例一訓練結果 MSE

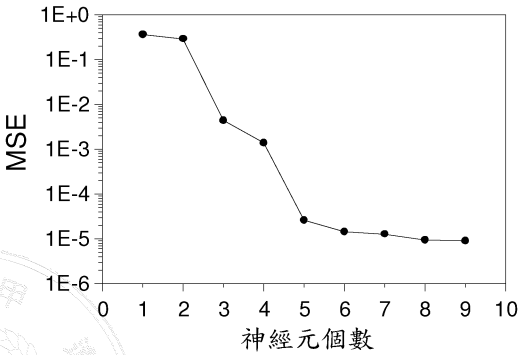

圖 6 範例一訓練結果

從圖 6 可以看出，當隱藏層神經元數目增加到五個以上，其 MSE 值的變化已經很小了，所以我們認爲隱藏層使用五個神經元就可以很充分的表示式(14)的關係式，故我們用測試數據代入訓練出來的網路結構中求輸出值，所得結果如圖 7 中的實線所示，而圖中的點則是 $Y = \sin(2\pi X)$ 的理論值。

從圖 7 中可以看出，理論值的點與我們用軟體模擬出來的線相當的接近，這說明了我們用軟體訓練出來，隱藏層含五個神經元的網路結構，可以對 $Y = \sin(2\pi X)$ 有很不錯的擬合效果。

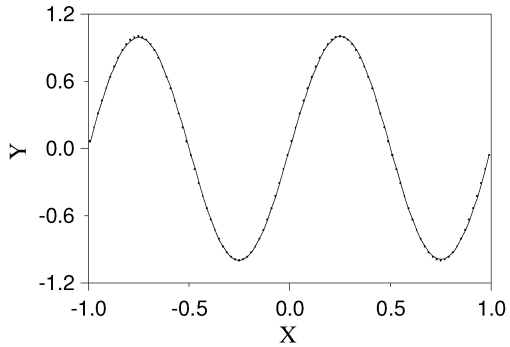

圖 7 範例一模擬數據與理論值比較圖

範例二：(二個輸入數據產生一個輸出數據)
第二個範例，我們用式(15)來測試。

$$Y = X_1 \cdot \sin(2\pi X_2) \qquad (15)$$

這個範例中 ，訓練數據我們取 X_1 分別等於 0.5、1、1.5、2、2.5、3、3.5、4 八個點，而 X_2 則是在範圍-1~1 之間分成 40 等分，可得 41 個點，所以共有 328 組數據，再將這 328 組數據代入式(15)求出相對目標值(Y)，即得到 328 組訓練對。另外測試數據則是分別取 X_1 及 X_2 每兩個相鄰數據點的中間值 ，所以共有 $280(7\times 40)$組。

再來說明其它網路條件的設定，首先是轉換函數 ，隱藏層用 TANSIG，而輸出層則是 PURELIN；另外，動量係數我們設定爲 0.5，至於學習速率 ，則是隨著訓練過程一直作修正。
最後訓練結果的 MSE 值如表 2 及圖 8 所示。

隱藏層神經元數目	訓練數據 MSE
5	2.66273E-01
10	7.54324E-02
13	9.35760E-03
15	9.14273E-03
20	8.96422E-03

表 2 範例二訓練結果 MSE

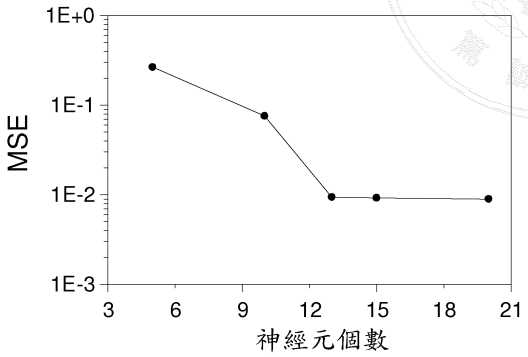

圖 8 範例二訓練結果

從圖 8 中觀察，我們選擇用隱藏層有 13 個神經元的網路結構來測試先前求出來的280組測試數據，而將得到的輸出值(Y)與 X_2 的關係，分別用圖 9 中的七條線表示，而每一條線分別代表相同 X_1 值時的關係。另外，圖中的點是用公式 $Y = X_1 \cdot \sin(2\pi X_2)$ 所求出的理論 Y 值。

最後，我們從圖 9 中可以看出，軟體訓練出來，隱藏層含 13 個神經元的網路結構，其對於方程式 $Y = X_1 \cdot \sin(2\pi X_2)$ 亦有不錯的擬合效果。

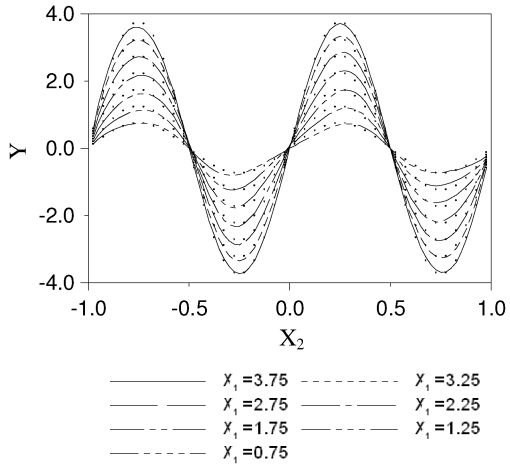

圖 9 範例二模擬數據與理論值比較圖

六、結論

從上述範例一、二可以知道本研究所設計的類神經網路軟體已具備了基本的功能，在一些數據的擬合上已可有不錯的結果，而未來我們還會繼續加入更多可以使用的轉換函數，以及增加可以使用的隱藏層層數，以期望軟體可以做的事能更多更廣。另外，本研究最主要是希望設計一個教學用類神經網路軟體，以提供初學者使用，所以在開發介面以及操作上，都朝簡單易懂來著手，所以對於第一次使用本軟體的人來說，應該能很快上手。

七、參考文獻

[1] McCulloch, W. and Pitts, W., “A logical calculus of the ideas immanent in nervous activity,” *Bulletin of Mathematical Biophysics*., Vol.5, pp.115-133, 1943.
[2] Hebb, D. O., *The Organization of Behavior*, New York: Wiley, 1949.
[3] Rosenblatt, F., “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychological Review*, Vol.65, pp.386-408, 1958.
[4] Rosenblatt, F., *Principles of Neurodynamics*, New York: Spartan, 1962.
[5] Hopfield J. J. and Tank, D. W., “Neural computation of decisions in optimization problems,” *Biological Cybernetics*, Vol.52, pp141-152, 1985.
[6] Widrow, B. and Hoff, M. E., “Adaptive switching circuits,” *1960 IRE WESCON Convention Record*, New York: IRE Part 4, pp.96-104, 1960.
[7] Minsky, M. and Papert, S., “*Perceptions：An Introduction to Computational Geometry*,” Cambridge, MA: The MIT Press, 1969.
[8] Hopfield, J. J., “Neural networks and

physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci. USA*, Vol.79, pp.2554-2558, 1982.

[9] Kohonen, T., "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, Vol.43, pp.59-69, 1982.

[10] Rumelhart, D. E. and McClelland J. L., eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol.1, Cambridge, MA: MIT Press, 1986.

[11] Kosko, B., "Feedback stability and unsupervised learning," *IEEE ICNN*, Vol.1, pp.141-152, 1988.

[12] Moody, J. and Darken, C. J., "Fast Learning in Networks of Locally-Tuned Processing Units," *Neural Computation*, Vol.1, pp.281-294, 1989.

[13] Johansson, E. M., Dowla, F. U. and Goodman, D. M., "Backpropagation for Multilayer Feed-Forward Neural Networks Using the Conjugate Gradient Method," *International Journal of Neural System*, Vol.2, No.4, pp.291-301, 1992.

[14] Leonard, J. and Kramer, M. A., "Improvement of the Backpropagation Algorithm for Training Neural Networks," *Computers Chm. Engng.*, Vol.14, No.3, pp.337-341, 1990.

[15] Baughman, D. R. and Liu, Y. A., *Neural Networks in Bioprocessing and Chemical Engineering*, San Diego: Academic Press, 1995.

[16] Hagan, M. T., Demuth, H. B. and Beale, M., *Neural Network Design*, Boston: PWS Publishing Company, 1996.

[17] Lopatin, C. M., Pizziconi, V., Alford, T. L. and Laursen, T., "Hydroxapatite Powders and Thin Films Prepared by a Sol-Geltechnique," *Thin Solid Films*, Vol.326, pp.227-232, 1998.