基於 NBN 演算法改良模糊類神經網路之收斂效率

林哲緯、洪國寶
國立中興大學研究所
death-J@hotmail.com
gbhorng@cs.nchu.edu.tw

摘要

在人工智慧領域中，大部分類神經網路都是以分類器為主要功能，其中又以倒傳遞類神經網路(Error Back Propagation)最為廣泛使用，在之後為了增加收斂的效率，便有人提出把 LM (Levenberg–Marquardt) Algorithm 應用在類神經網路的收斂上。2010 年 Bogdan M. Wilamowski 提出了 NBN (Neuron -By-Neuron) Algorithm，除了改善 LM Algorithm 的幾個缺點，並再次提昇類神經網路的收斂效率。而類神經網路的另一種架構 ANFIS (Adaptive Neuron-Fuzzy Inference Systems)則代表了較為有推理、判斷性的一方。本篇文章的目的則是將 NBN algorithm 與 ANFIS 的優點結合，使得神經網路在擁有推理、預測性質的同時也能有良好的收斂效率。

關鍵詞：Error Back Propagation、LM algorithm、NBN algorithm、ANFIS

1.前言

在類神經網路中，不論是哪種類型的神經網路都是以節點(Node)和權重(Weight)組成，除了部份如 Hopfield 不需要使用倒傳遞運算，其餘大部分都是使用此種方法修正權重，藉此使神經網路的輸出結果盡可能符合我們希望的結果，但是隨著將其應用在資料龐大的分析統計上，如何在有效時間內快速將 weight 收斂到預定值則是很重要的課題。

倒傳遞網路則是最早被提出來的收斂方法，其原理是利用最陡坡降法 (The Gradient Steepest Descent Method)將誤差函數最小化，以期將所有訓練範例輸出的誤差調整在可接受的誤差值範圍內，不過由於訓練速度過於緩慢，即使使用了多種加速收斂的方法，速度上還是無法與二階演算法 NBN Algorithm 相比。

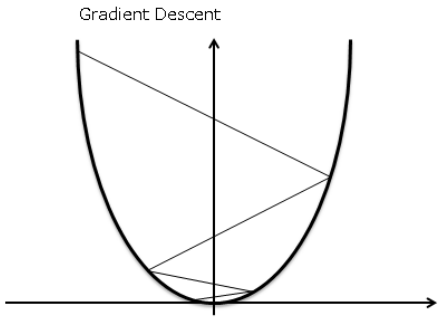
但即使收斂在快，類神經網路終究只是拿來分類樣本的技術，若想達成連續性、推理性、預測性等功能，則必須使用模糊邏輯理論(Fuzzy Logic)，ANFIS[2]便是使用此種方式收斂，本文主要架構就是將 ANFIS 的優點保留，並簡化其原本的架構，在導入 NBN Algorithm [6]的 Without Backpropagation 技術，使其成為一個能快速收斂，並且保有 ANFIS 特性的類神經網路。

本篇的重點在於改變 ANFIS 的架構，使其能更快速的收斂，若是在不同樣本下，還能使用不同的架構以對應不同的規則。

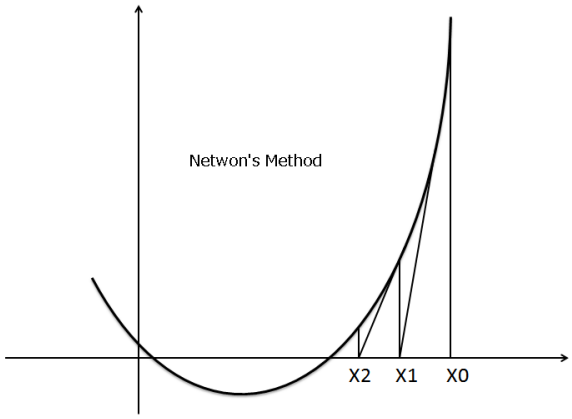
在第二段介紹了 NBN Algorithm，為何能比 EBP 更有效率的收斂；第三段解釋了 ANFIS 的基礎架構精神；第四段說明了如何改進 ANFIS，並推導了 NBN Algorithm 應用在 ANFIS 上的結果；第五段則為結論。

2.NBN Algorithm

在倒傳遞網路中，最原始的方法是藉由計算每一條權重的梯度(Gradient)，並循著其方向逐漸找到最佳解，雖然並不一定能找到全域最佳解(Global Optimum)，但是能有區域最佳解(Local Optimum)即可，於是 EBP 使用了一階最陡坡降法(圖一)，但是此演算法在處理一些複雜的非線性函數時，在越接近極小值時收斂速度將會變得非常緩慢[1]。



圖一:距離最小值越近收斂越慢



圖二:與圖一相反，距離最小值遠時收斂較慢

為了解決這個問題，便有人提出使用 LM Algorithm 取代傳統的收斂方式，其原理是在原本極難收斂的最小值區域轉變收斂方式，使用與最陡坡降法性質相反的牛頓法(Newton's Method) (圖二)，由於牛頓法有著在接近極小值時好收斂而遠離時較難收斂的特性，故大量的縮減了傳統 EBP 的收斂時間[4]。

EBP 對權重的改變方式:

$$\Delta w = -\alpha * g$$

LM Algorithm 對權重的改變方式:

$$\Delta w = -(J^T J + \mu I)^{-1} J^T e$$

α是傳統學習速率，g是梯度，J是 Jacobian 矩陣，I是單位矩陣，e是誤差值，μ為隨時變動的學習參數，當μ極大則接近牛頓法，反之則接近最陡坡降法。

而在 NBN Algorithm 中改善了幾個在 LM Algorithm 裡存在的問題[5]，如記憶體空間的省略，由於 Jacobian 矩陣(圖三)的相乘過於龐大，且計算過於複雜，使得每個 Iteration 會耗費大量時間，所以 NBN Algorithm 使用不需要倒傳遞的演算法以加快收斂速度，這在之後會詳細解釋。

$$\begin{bmatrix}
\overbrace{\begin{matrix}\frac{\partial e_{1,1}}{\partial w_{1,1}} & \frac{\partial e_{1,1}}{\partial w_{1,2}} \\ \frac{\partial e_{1,2}}{\partial w_{1,1}} & \frac{\partial e_{1,2}}{\partial w_{1,2}} \\ \cdots & \cdots \\ \frac{\partial e_{p,1}}{\partial w_{1,1}} & \frac{\partial e_{p,1}}{\partial w_{1,2}} \\ \frac{\partial e_{p,2}}{\partial w_{1,1}} & \frac{\partial e_{p,2}}{\partial w_{1,2}} \\ \cdots & \cdots \\ \frac{\partial e_{np,1}}{\partial w_{1,1}} & \frac{\partial e_{np,1}}{\partial w_{1,2}} \\ \frac{\partial e_{np,2}}{\partial w_{1,1}} & \frac{\partial e_{np,2}}{\partial w_{1,2}}\end{matrix}}^{neuron\ 1} & \cdots & \overbrace{\begin{matrix}\frac{\partial e_{1,1}}{\partial w_{j,1}} & \frac{\partial e_{1,1}}{\partial w_{j,2}} \\ \frac{\partial e_{1,2}}{\partial w_{j,1}} & \frac{\partial e_{1,2}}{\partial w_{j,2}} \\ \cdots & \cdots \\ \frac{\partial e_{p,1}}{\partial w_{j,1}} & \frac{\partial e_{p,1}}{\partial w_{j,2}} \\ \frac{\partial e_{p,2}}{\partial w_{j,1}} & \frac{\partial e_{p,2}}{\partial w_{j,2}} \\ \cdots & \cdots \\ \frac{\partial e_{np,1}}{\partial w_{j,1}} & \frac{\partial e_{np,1}}{\partial w_{j,2}} \\ \frac{\partial e_{np,2}}{\partial w_{j,1}} & \frac{\partial e_{np,2}}{\partial w_{j,2}}\end{matrix}}^{neuron\ 2} & \cdots
\end{bmatrix}$$

圖三: Jacobian 矩陣，行數為權重個數，列數為樣本數*輸出個數

3.ANFIS

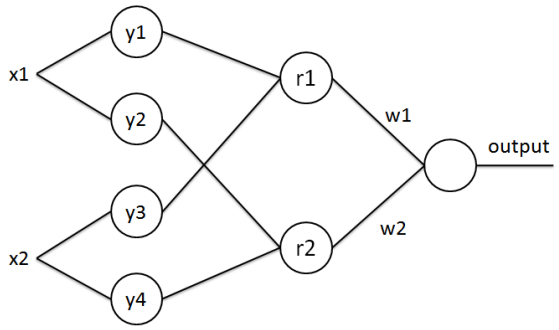
模糊神經網路與一般神經網路有著不同概念，主要是藉由模糊函數與規則達到 EBP 無法做到的模糊分類，而 ANFIS 則是 EBP 和模糊神經網路的結合，在分類方面需要倚賴的規則藉由類神經網路自我學習，且需要調整的權重與模糊函數也藉由 EBP 修正，簡單分類可分為三個階段(圖四):

輸入層:主要是將所有輸入的值藉由模糊函數分配到下一層規則層，以達到推論的效果。

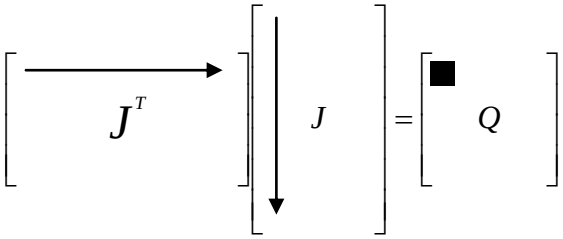
規則層:將輸入層傳遞過來的值進行模糊運算，如圖四的$r1 = y1$。$y3$的合成運算。

推論層:解模糊化後將最有可能的結果輸出。

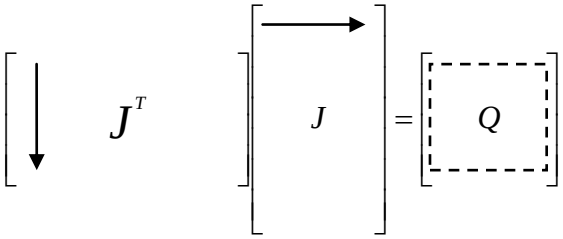
若是最終結果不符合預期，則藉由調整權重、模糊函數，直到結果符合即為收斂。



圖四:NFNFIS 架構圖



(a)



(b)

圖五:不同的 Jacobian 計算方式

4.NBNFIS

NBNFIS 主要目的是使用少量的空間換取計算時間，因為在 ANFIS 中使用的是 EBP，在每個 Iteration 因為需要大量的時間處理梯度的運算，於是這裡改變了傳統的倒傳遞方法。

我們定義誤差函數為(1)

$$E = \frac{1}{2}(d - o)^2 \tag{1}$$

d 為期望值，而 o 為實際輸出值，而 o 為推論層解模糊化後的輸出結果，函數為(2)

$$o = \frac{\sum r_n w_n}{\sum r_n} \tag{2}$$

w 為權重，而 r 為模糊化進入規則層的模糊運算結果，而模糊化函數則使用高斯函數(3)

$$y = e^{-\left(\frac{x-m}{\sigma}\right)^2} \tag{3}$$

x 為樣本的輸入值，m 跟σ為更改高斯函數的函數型態所需要變動的值，由於要更改的值有 w、m、σ，所以 Jacobian 矩陣第一列如(4)

$$J_{11} = \left[\frac{\partial E}{\partial \sigma_1}\frac{\partial E}{\partial m_1}\cdots\frac{\partial E}{\partial \sigma_n}\frac{\partial E}{\partial m_n}\frac{\partial E}{\partial w_1}\cdots\frac{\partial E}{\partial w_n}\right] \tag{4}$$

由於直接計算 Jacobian 矩陣相乘會浪費太多記憶體，於是採用一次計算一列的方式，也只需要儲存一列(圖五)：

$$(J^T J + \mu I)^{-1} J^T e = (Q + \mu I)^{-1} g \tag{5}$$

$$Q = J_{11} + \cdots + J_{1q} + J_{21} + \cdots + J_{p1} + \cdots + J_{pq}$$

p 跟 q 是輸出的節點數跟樣本的個數，若直接使用圖五(a)方法計算，因為無法一次取得整行的數值，所以必須將整個 Jacobian 矩陣儲存起來，最後才計算出 Q，而圖五(b)則是在每個樣本進來時都可以計算出部份的 Q，如此即可算出高斯矩陣 Q，在梯度方面也使用一列 Jacobian 矩陣即可計算，而每一列的各元素計算如下：

$$\frac{\partial E}{\partial w_n} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial w_n} \tag{6}$$

對於權重的改變，只需要使用(6)計算從輸出層的錯誤誤差即可。

$$\frac{\partial E}{\partial \sigma_i} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial \sigma_i} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial y_i}\frac{\partial y_i}{\partial \sigma_i} \quad (7)$$

$$\frac{\partial E}{\partial m_i} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial m_i} = \frac{\partial E}{\partial o}\frac{\partial o}{\partial y_i}\frac{\partial y_i}{\partial m_i} \quad (8)$$

對於模糊函數的改變，若屬於同一個函數，則使用(7)(8)改變輸入層的數值，故J_{11}可寫成：

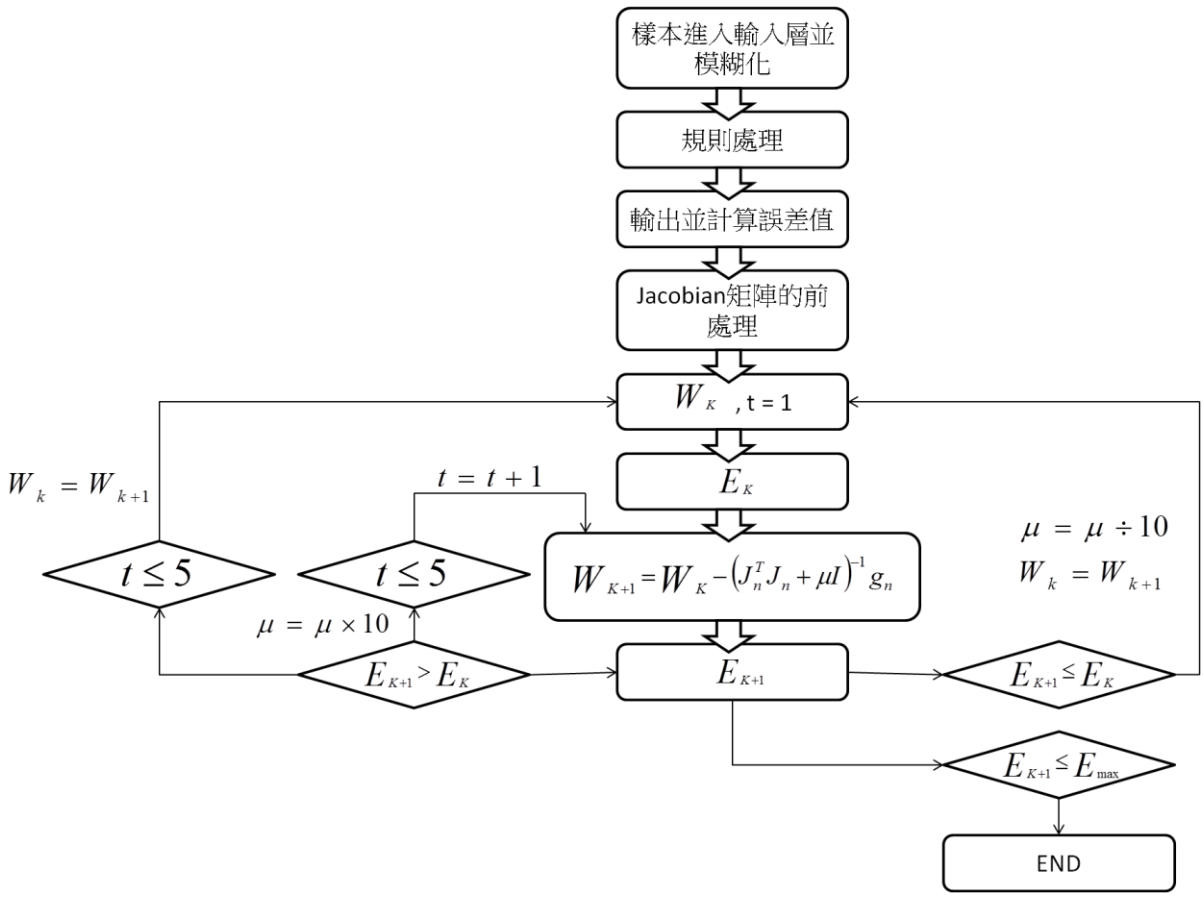
$$J_{11} = \left[\frac{\partial E}{\partial o}\frac{\partial o}{\partial y_1}\left[\frac{\partial y_1}{\partial \sigma_1} \quad \frac{\partial y_1}{\partial m_1}\right]\cdots \right.$$
$$\left. \cdots \frac{\partial E}{\partial o}\frac{\partial o}{\partial y_i}\left[\frac{\partial y_i}{\partial \sigma_i} \quad \frac{\partial y_i}{\partial m_i}\right]\frac{\partial E}{\partial o}\left[\frac{\partial o}{\partial w_1}\cdots\frac{\partial o}{\partial w_n}\right]\right]$$

由於做了類似前處理的動作，在每個 Iteration 就不需要重新計算，一個 Iteration 的過程如圖六：

每次 Iteration 在數次以內若是無法找到比前一次更好的錯誤誤差，則不取代原本的權重，這邊使用 5 次為一個 Iteration 的上限。

5.結論

本文結合了 NBN Algorithm 的優點，改善了 ANFIS 的收斂效率，並且不失去原本對於資料的推理、預測性，因為使用些微空間換取時間，在前處理與 Jacobian 矩陣的計算優化之下，記憶體使用空間也降低許多，而且在規則層方面也不侷限於傳統的連接規則，可以彈性的調整適合的類神經網路架構[3]以對應不同的資料樣本。



圖六:NBNFIS 的流程圖

參考文獻

[1] R. Hecht-Nielsen, “Theory of the back propagation neural network,” in Proc. IEEE IJCNN, Washington D.C., pp.593 – 605, Jun. 1989.

[2] J-S. R. Jang, “ANFIS : adaptive-network-based fuzzy inference system,” IEEE Trans. Systems, Man and Cybernetics, Vol. 21, No. 11, p1793-1803, Nov. 2010.

[3] B. M. Wilamowski, D. Hunter, and A. Malinowski, “Solving parity- N problems with feedforward neural networks,” in Proc. IEEE IJCNN, Piscataway, NJ: IEEE Press, pp. 2546–2551, 2003.

[4] B. M. Wilamowski, N. J. Cotton, O. Kaynak,, and G. Dündar, “Computing Gradient Vector and Jacobian Matrix in Arbitrarily Connected Neural Networks,” IEEE Trans. Industrial Electronics, Vol. 55, No. 10, pp.3784 – 3790, Oct. 2008.

[5] B. M. Wilamowski and H. Yu, “Improved computation for Levenberg– Marquardt training,” IEEE Trans. Neural Netw., vol. 21, no. 6, pp. 930–937, Jun. 2010.

[6] B. M. Wilamowski and H. Yu, “Neural Network Learning without Backpropagation,” IEEE Trans. Neural Network, Vol. 21, No. 11, pp.930 – 937, Nov. 2010.