國立交通大學

資訊科學與工程研究所

碩 士 論 文

高階類神經網路與差分進化法於井測資料反推

Higher Order Neural Networks and
Differential Evolution for Well Log Data
Inversion

研 究 生：陳俊宇

指導教授：黃國源　教授

中 華 民 國 九 十 七 年 六 月

高階類神經網路與差分進化法於井測資料反推
Higher Order Neural Networks and Differential Evolution for Well Log Data Inversion

研 究 生：陳俊宇　　Student：Chun-Yu Chen

指導教授：黃國源 教授　　Advisor：Dr. Kou-Yuan Huang

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年六月

高階類神經網路與差分進化法於井測資料反推

學生：陳俊宇　　指導教授：黃國源 教授

國立交通大學資訊科學與工程研究所碩士班

摘　　要

關鍵字：類神經網路，多層感知器，梯度坡降法，基因演算法，差分進化法，井測資料反推。

我們採用多層感知器的類神經網路於井測資料的反推，網路的訓練方式分別採以梯度坡降法為基礎的倒傳遞學習法以及先利用差分進化法與再使用梯度坡降法的兩階段訓練方式。類神經網路的輸入是井測資料的視導電率 (apparent conductivity, Ca)而輸出是井測資料的地層真實導電率 (true formation conductivity, Ct)。訓練網路時，網路的輸入是由原始的特徵值與其高階的特徵值所組成。

實驗顯示，採用此種擴展型特徵值的倒傳遞學習法網路可以加速網路的學習效率以及降低實際輸出值與期望輸出值之間的絕對值誤差。雖然先利用差分進化法再使用梯度坡降法去訓練高階類神經網路的訓練時間較長，但是卻可以得到一個更精確的結果。

當網路先利用差分進化法再利用梯度坡降法來訓練時，在輸入為十個特徵值且擴展到三階，隱藏節點為八個，以及輸出節點為十個的時候，可以得到與模擬資料間的最小絕對值誤差。接著再把這個網路利用在真實的井測資料上，以測試網路的反推結果。

Neural Networks and Differential Evolution for Well Logging Inversion

Student：Chun-Yu Chen Advisor：Dr. Kou-Yuan Huang

Institute of Computer Science and Engineering
National Chiao Tung University

ABSTRACT

Keywords: Neural networks, multilayer perceptron, gradient descent, genetic algorithm, differential evolution, well log data inversion.

Multilayer perceptron is adopted for well log data inversion. The gradient descent method is used in the back propagation learning rule and a hybrid method, which combines differential evolution (DE) and gradient descent method, are used in training process respectively. The input of the neural network is the apparent conductivity (Ca) of the well log and the desired output is the true formation conductivity (Ct). The higher order of the input features and the original features are the network input for training.

From our experimental results, we find the expanding input features with back propagation learning rule can get fast convergence in training and decrease the mean absolute error between the desired output and the actual output. The hybrid method will provide more precise results, despite it takes a longer training time.

The multilayer perceptron network, which is trained by the hybrid method, with 10 input features, the expanding input features to the third order, 8 hidden nodes, and 10 output nodes can get the smallest average mean absolute error on simulated well log data. And then the system is applied on the real well log data.

誌　　謝

本文承蒙黃國源教授之悉心指點、指正引導方能順利完成，在此致上最誠摯的謝意。黃教授不僅教學態度嚴謹，做研究更是實事求是、講求科學精神，是我在求學上以及將來就業上的典範。同時感謝參加口試的口試委員，劉長遠教授、蘇豐文教授、以及王榮華教授給予學生修改論文上的諸多寶貴建議。

感謝 Chevron Oil Company 以及 Prof. Liang-Chi Shen 提供本論文所使用的井測資料，沒有這些寶貴的井測資料這篇論文便無法完成。

感謝實驗室的同學及好友，在論文研究期間所給予我精神上莫大的鼓勵；最後，謹以此文獻給我摯愛的雙親。

Contents

圖說明

表說明

1. Introduction

井測資料是一種利用儀器來蒐集及辨識地表下資訊的技術，已經被廣泛地使用在油氣探測的工業上。井測資料的反推是為了得到油井周圍的真實結構，是一個困難的問題 [1]。

傳統的井測反推方法，像是 maximum entropy 的方式以及以 Marquardt-Levenberg 為基礎的最小平方誤差法 [2][3]，在油井具有相對較大的斜度時，就需要龐大的計算量。 Goswami [1] 利用了演化計算法來反推井測資料，但是它是一個較沒有效率的方式。 Martin 則使用了 Levenberg-Marquardt (LM) 學習法的類神經網路來反推模擬的井測資料。比較過這些方法之後，類神經網路可以有一個較快的訓練速度以及能夠得到一個可靠的結果。

但是 Martin 並沒有使用由 Pao [5][6] 所提出的具有高階特徵向量的類神經網路 (higher order feature neural network, HONN)。而 Pao 只有建立了單層感知器的函數連結，在這篇論文中，我們使用了原始的輸入向量以及將原始輸入向量擴展到兩次方及三次方再一起當作多層感知器的輸入來做井測資料的反推。高階特徵向量能夠讓視導電率 (apparent conductivity, Ca) 更加非線性地對映成真實地層的導電率 (true formation conductivity, Ct)。

使用坡度遞減 (gradient descent) 的方式有一個明顯的問題就是容易陷入局部最小值 [7]，而全域的演化式演算法則適合用來解決這種問題，並且它也可以拿來訓練類神經網路 [8]-[11]。現今已有很多種的演化式演算法，但是它們都必須滿足四個基本的需求。第一，必須能夠處理不可微分、非線性的函數。第二，必須能夠使用平行式的計算來降低函數的計算量。第三，必須容易使用；只能夠有少數的控制變數來執行最佳化，而且這些變數也必須容易地選擇。最後，必須有良好的收斂效果；這意味著經過一連串獨立的測試，都必須要能夠收斂到全域的最佳解。

基因演算法 (genetic algorithm, GA) 是一種演化式演算法，已經被廣泛地

應用在許多最佳化的問題上 [13]-[15]。但是當 GA 的族群太小時，GA 執行的結果並不令人滿意，會讓 GA 過早收斂以及得到一個非最佳的解。如果一味地增加族群的大小，則會需要一個令人無法忍受的計算時間來執行 GA。為了解決這個問題，Storn 與 Price [12] 提出了一個新的演化演算法，稱作差分進化法 (differential evolution, DE)。DE 已經被發現能夠勝過許多種類的 GA [16]，而且也能夠被應用在許多種類的最佳化問題上 [17]-[19]。最後，我們混和了差分進化法以及高階類神網路來進行井測資料的反推。

2. Neural Networks and Learning

類神經網路的架構分成四個種類，第一種是單層感知器，第二種是高階的單層感知器，第三種是兩層的感知器而第四種是兩層的高階感知器。這裡的訓練方式採用倒傳遞學習法則。

2.1 Single Layer Perceptron

圖 2.1 是一個單層的感知器 [20]。

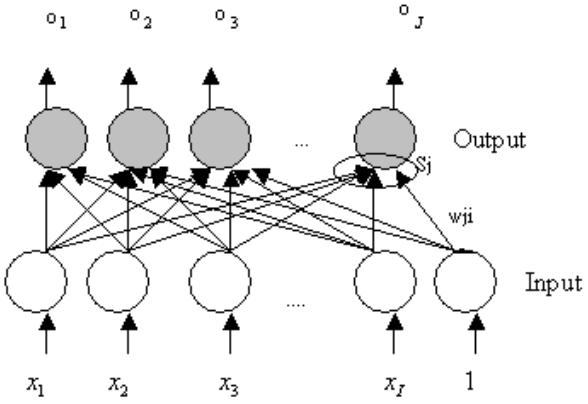


圖 2.1. 單層感知器。

對一個任意的輸出節點 j：

$$s_j = \sum_{i=1}^{I+1} w_{ji} x_i \tag{2.1}$$

i 是輸入層節點的 index，j 是輸出層節點的 index，而 I +1 代表輸入層節點的個數。x_i 是任意一個輸入節點，而 w_{ji} 是連結輸入節點 i 與輸出節點 j 的權重 (weighting)。輸出節點 j 的輸出為 $f(s_j)$，f稱為活化函數 (activation function)，在這裡我們使用雙彎曲函數 (sigmoidal function) 來當作活化函數而感知器的輸出如下：

$$o_j = f(s_j) = \frac{1}{1+e^{-s_j}} \tag{2.2}$$

網路的誤差函數 (error function) E，定義如下：

$$\mathrm{E} = \frac{1}{2}\sum_{j=1}^{J} (d_j - o_j)^2 \tag{2.3}$$

J 代表輸出節點的個數，d_j 表示第 j 個輸出節點的期望輸出值 (desired output)，而 o_j 表示第 j 個輸出節點的實際輸出值 (actual output)，而網路中權重的調整方式是採用梯度坡降法 (gradient descent) 來調整 [21]

$$\Delta w_{ji} = w_{ji}(t+1) - w_{ji}(t) \;=\; -\eta \frac{\partial E}{\partial w_{ji}} \tag{2.4}$$

η 是網路的 learning rate 而 t 是執行的次數。因此，網路調整的公式如下：

$$\Delta w_{ji} = \eta((d_j - o_j) f_j'(s_j) x_i) = \eta \delta_j x_j \tag{2.5}$$

$$\delta_j = (d_j - o_j) f_j'(s_j) \tag{2.6}$$

除此之外，我們另外加入慣性項 (momentum term) 來考慮前一個步驟網路調整所造成的影響 [22]。所以，最終的調整公式為：

$$\Delta w_{ji}(t) = \eta \delta_j(t) x_j(t) + \beta \Delta w_{ji}(t-1) \tag{2.7}$$

β 稱為 momentum coefficient。

2.2 Single Layer HONN

單層的 higher order neural network (HONN) 是將原始的輸入向量擴展至高階項。圖 2.2 是一個單層的 HONN，而其網路權重調整的方式與上述的單層感知器網路相同。

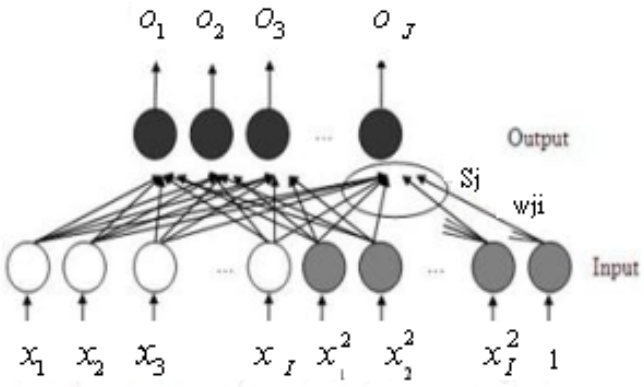


圖 2.2. 單層的 HONN。

2.3 Multilayer Perceptron with One Hidden Layer

2.3.1 Introduction

圖 2.3 是一個兩層的類神經網路。

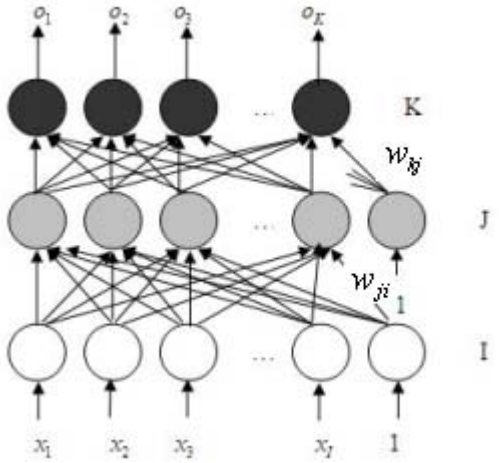


圖 2.3. 兩層的類神經網路。

i 是輸入層節點的 index，*j* 是隱藏層節點的 index，*k* 則是輸出層節點的 index，I+1 是輸入層節點個數，J+1 是隱藏層節點個數，而 K+1 是輸出層節點

個數。網路權重的調整採用梯度坡降法。其中，網路的輸入為 x_i ， w_{kj} 是連結隱藏層節點 j 與輸出層節點 k 的權重， w_{ji} 是連結輸入層節點 i 與隱藏層節點 j 的權重。網路中的活化函數 f 皆採用雙彎曲函數，所以隱藏層節點的輸出 o_j 與輸出層節點的輸出 o_k 分別為：

$$o_j = f(s_j), \quad s_j = \sum_{i=1}^{I+1} w_{ji} x_i \tag{2.8}$$

$$o_k = f(s_k), \quad s_k = \sum_{J=1}^{J+1} w_{kj} o_j \tag{2.9}$$

隱藏層與輸出層之間的權重調整公式如下：

$$\Delta w_{kj} = \eta((d_k - o_k) f_k'(s_k) o_j) = \eta \delta_k o_j \tag{2.10}$$

$$\delta_k = (d_k - o_k) f_k'(s_k) \tag{2.11}$$

輸入層與隱藏層之間的權重調整公式如下：

$$\delta_k = (d_k - o_k) f_k'(s_k)$$

$$\Delta w_{kj} = \eta((\sum_{k=1}^{K} \delta_k w_{kj}) f_j'(s_j) x_i) = \eta \delta_j x_i \tag{2.12}$$

$$\delta_j = (\sum_{k=1}^{K} \delta_k w_{kj}) f_j'(s_j) \tag{2.13}$$

在這裡，我們也加入了慣性項來考慮前一個步驟網路調整所造成的影響，所以最終的調整公式為：

$$\Delta w_{kj}(t) = \eta \delta_k(t) o_j(t) + \beta \Delta w_{kj}(t-1) \tag{2.14}$$

$$\Delta w_{ji}(t) = \eta \delta_j(t) x_i(t) + \beta \Delta w_{ji}(t-1) \tag{2.15}$$

η 是網路的 learning rate 而 β 稱為 momentum coefficient。

2.3.2 On hidden nodes for neural nets

關於兩層感知器網路的隱藏層的節點個數，我們使用由 Mirchandani 與 Cao [23] 所推導出來的定理，來決定隱藏層的節點個數。

定理 2.1. 在網路的輸入個數為 d 時 (意味著有 d 度空間)，而隱藏的數目為 H 時，則可得最多線性可分割的區域數 M 為

$$\mathrm{M(H,d)}=\sum_{k=0}^{d} C(H,k) = C(H,0)+C(H,1)+...+C(H,d) \qquad (2.16)$$

where $C(H,k)=0,\ H<k$

根據此一定理，由最多可分割的區域數可推得訓練樣本 (training patterns) 的數目 T 的 lower bound。

假設網路為 10-H-10，H 為隱藏層節點的個數，訓練樣本數為 500 個 (T = 500)，而網路的輸入節點數為 10 (d = 10)，根據定理，我們可以得到最多分割區域數 M 為：

∵ H < d

H= 1, M(1,10) = $2^1 = 2$ regions

H = 2, M(2,10) = $2^2 = 4$ regions

H = 3, M(3,10) = $2^3 = 8$ regions

H= 4, M(4,10) = $2^4 = 16$ regions

H = 5, M(5,10) = $2^5 = 32$ regions

H = 6, M(6,10) = $2^6 = 64$ regions

H= 7, M(7,10) = $2^7 = 128$ regions

H = 8, M(8,10) = $2^8 = 256$ regions

H = 9, M(9,10) = $2^9 = 512$ regions

理想上，H = 9 時就足夠能解決這個問題。

2.4 Multilayer HONN with One Hidden Layer

兩層的 higher order neural network (HONN) 是將原始的輸入向量擴展至高階項。圖 2.4 是一個兩層的 HONN，而其網路權重調整的方式與上述的兩層感知器網路相同。

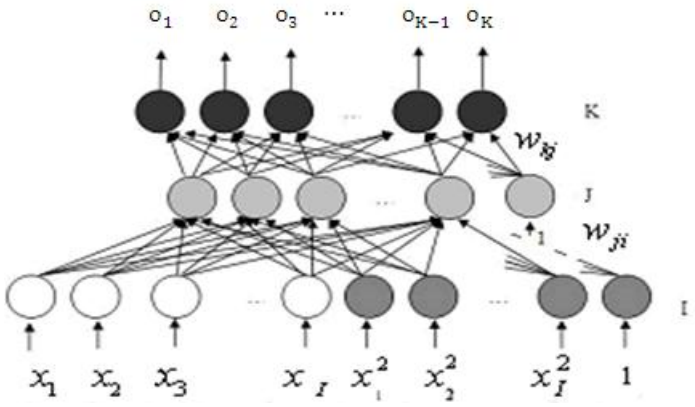


圖 2.4. 兩層的 HONN。

2.5 Training Process

實驗中，網路的輸入是視導電率 (apparent conductivity, Ca)，即視電阻率 (apparent resistivity, Ra) 的倒數，而網路的期望輸出是地層真實導電率 (true formation conductivity, Ct)。圖 2.5 為這些性質的轉換圖。

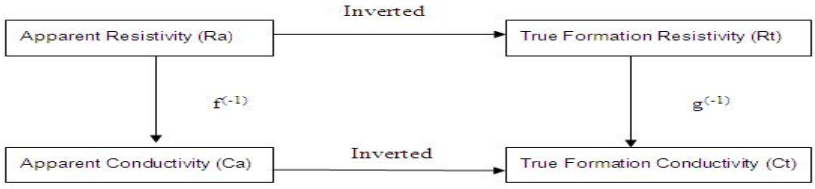


圖 2.5. 視電阻率、視導電率、真實電阻率與真實導電率之間的轉換圖。

我們一共有三十一組模擬的井測資料，拿第一組到第二十五組的模擬資料來做網路的訓練，第二十六組到第三十一組來做測試。每一組模擬的資料的油井深度是從 490 英呎到 589.5 英呎，而取樣區間 (sample interval) 是 0.5 英呎。所以每一組模擬的井測資料一共有兩百個模擬的視導電率與其對應的地層真實導電率。

我們將 200 個模擬的視導電率分別分成 1 個、2 個、4 個、5 個、10 個、

20 個、40 個、50 個、100 個、200 個訓練樣本，接著分別將這些訓練樣本正規化 (normalized) 在 0.1 到 0.9 之間，再將這些訓練樣本當作單層感知器網路與具有高階特徵向量的高階類神經網路 (HONN) 的輸入。接著再比較每一種網路在訓練過程的平均絕對值誤差 (mean absolute error, MAE)，找出最適合的網路架構。平均絕對值誤差定義如下：

$$\text{MAE}=\frac{1}{PK}\sum_{p=1}^{P}\sum_{k=1}^{K}\left|d_{pk}-o_{pk}\right| \qquad (2.17)$$

P 是訓練樣本的總數目，K 是輸出節點的總數目，d_{pk} 與 o_{pk} 分別代表第 p 個樣本與第 k 個輸出節點的期望輸出值與實際輸出值。網路的執行步驟設定為 20,000，學習速率為 0.6，動量係數為 0.4，而誤差值的門檻 (error threshold) 則設定為 0.002。訓練完之後，再分別對第二十六組至第三十一組模擬資料去做測試，每一個測試的資料都可以得到一組由類神經網路得出來的輸出值。最後再將這些輸出值經過還原數值 (re-scaled) 的步驟，即可以得到由類神經網路反推出來的一組真實地層導電率 (true formation conductivity, Ct)。

3. Experimental Results on Well Log Inversion with Neural Networks

感知器網路的實驗分成四個部分，第一部分為利用模擬的井測資料對單層的感知器網路做訓練。第二部分為利用模擬的井測資料對單層的 HONN 做訓練。藉由第一部分及第二部分的實驗，找出最適合的單層網路架構。第三部分為利用模擬的井測資料對兩層的感知器網路做訓練，第四部分為利用模擬的井測資料對兩層的 HONN 做訓練。藉由第三部分及第四部分的實驗，找出最適合的兩層 HONN 架構。這篇論文中，我們使用的電腦 CPU 為 Intel Pentium IV 3.0 GHz，作業系統為 Windows Vista。

3.1 Results of Single Layer Neural Networks

在這個實驗中，我們使用了單層的感知器網路來執行模擬的井測資料反推。單層感知器的網路架構如圖 3.1 所示。網路的輸入節點個數 I 分別為 1、2、4、5、10、20、40、50、100、與 200。圖 3.2 (a) -(e)為二十五組訓練資料其中的五個資料，而訓練的結果則記錄在表 3.1 中，圖 3.3 為網路大小為 10-10 時的訓練過程，圖 3.4 為網路大小為 50-50 時的訓練過程。

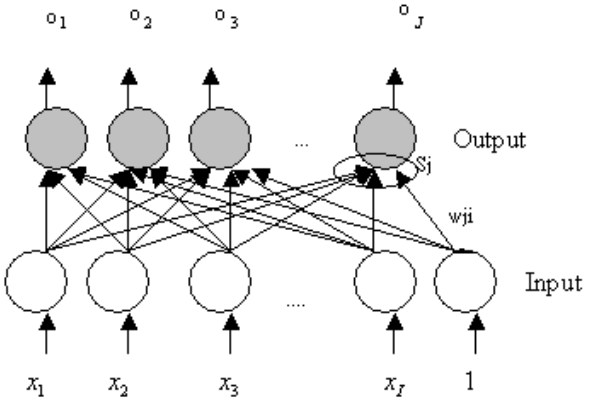
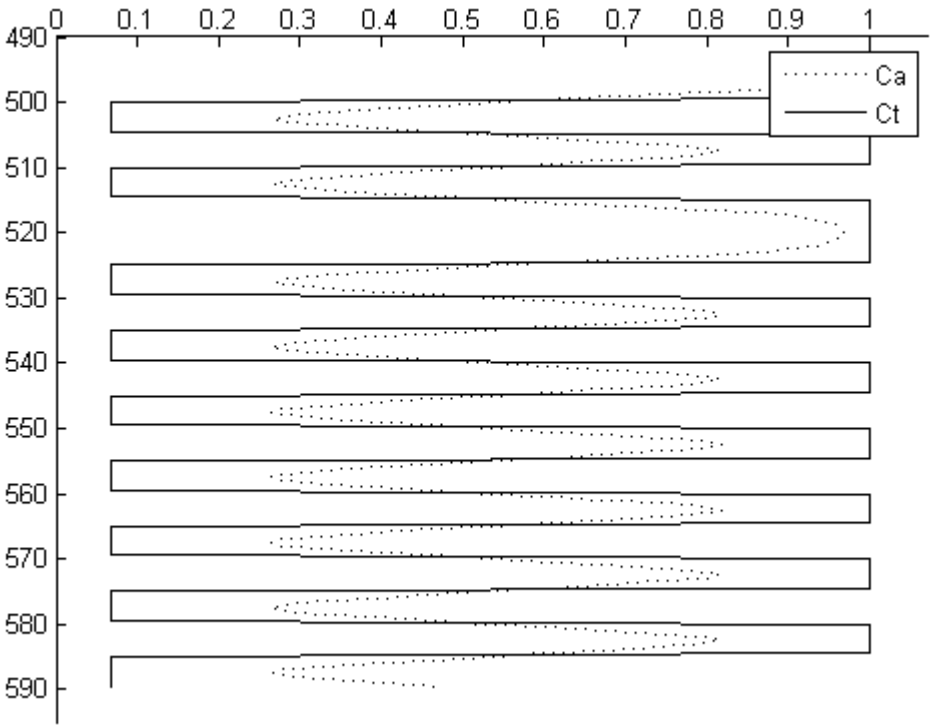
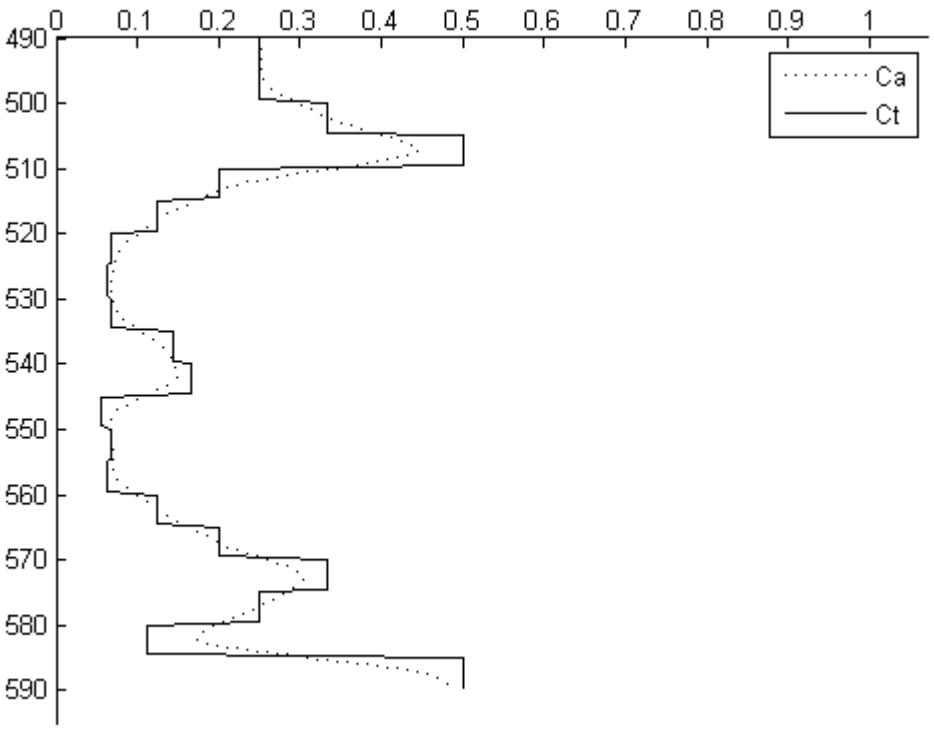


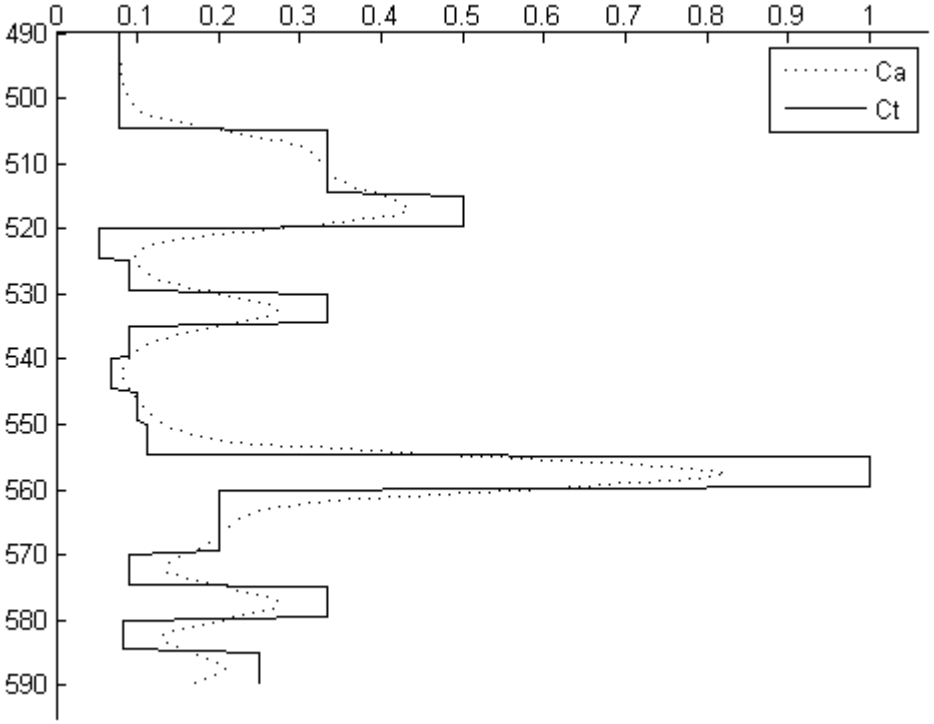
圖 3.1. 單層感知器。



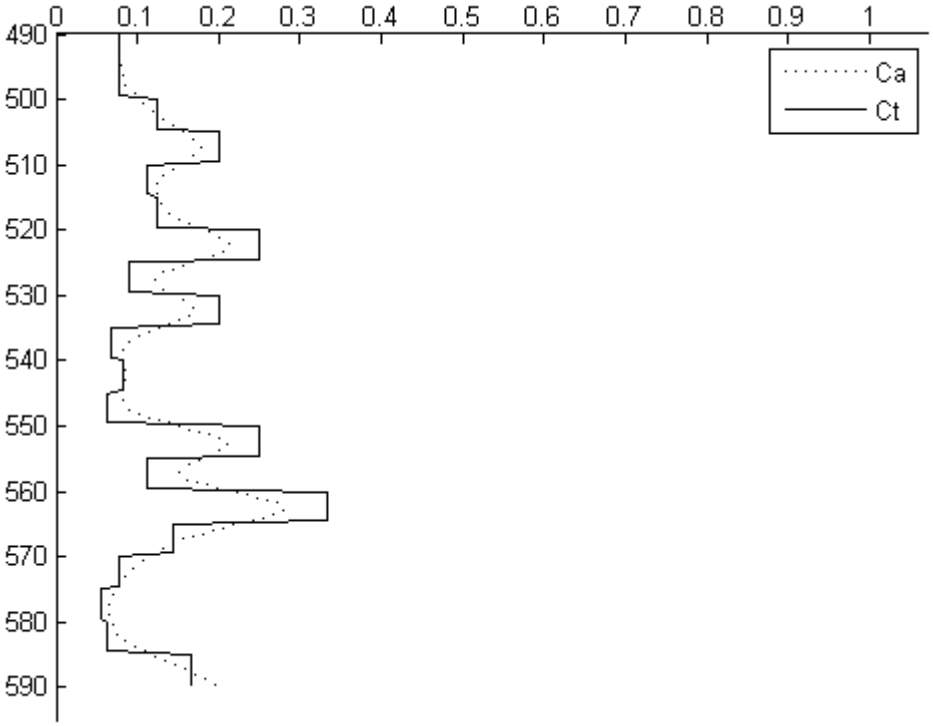
(a) 第一組訓練資料。



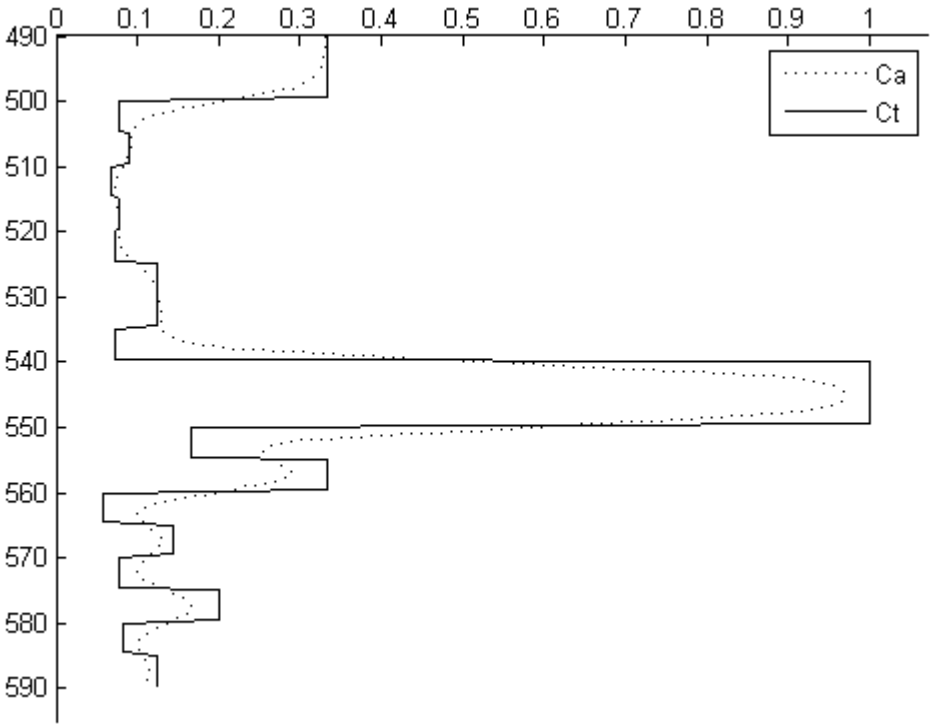
(b) 第七組訓練資料。



(c) 第十三組訓練資料。



(d) 第十九組訓練資料。



(e) 第二十五組訓練資料。

圖 3.2. 二十五組訓練資料中的五組訓練資料。

表 3.1. 每一種單層感知器網路的實驗結果。

Network size	Number of training patterns	Error MAE at 20,000 iterations
1-1	5000	0.055739
2-2	2500	0.055430
4-4	1250	0.046294
5-5	1000	0.026866
10-10	500	0.018248
20-20	250	0.021283
40-40	125	0.022017
50-50	100	0.023346
100-100	50	0.023534
200-200	25	0.744285

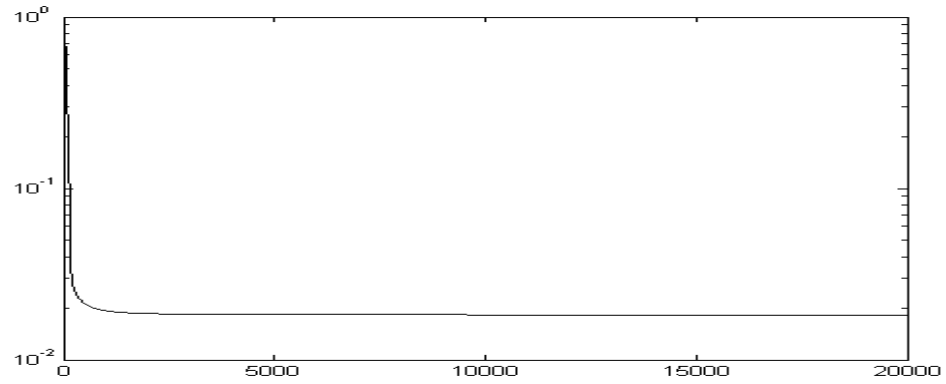


圖 3.3. 網路為 10-10 時的訓練過程。

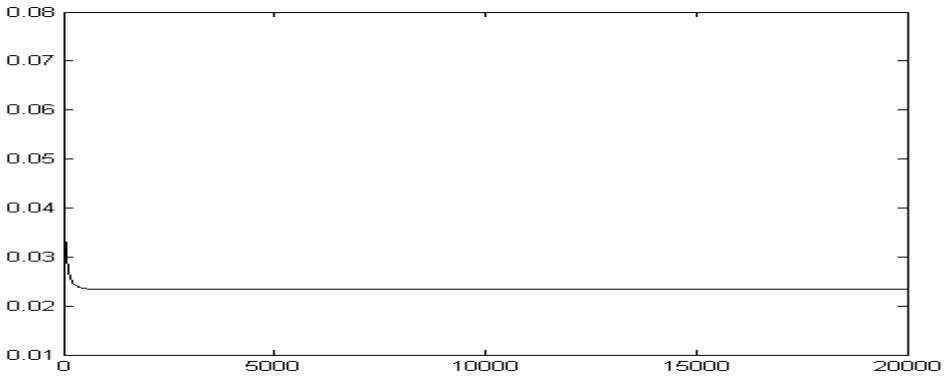


圖 3.4. 網路為 50-50 時的訓練過程。

當所有的網路訓練完之後，我們分別將第二十六組到第三十一組模擬資料當作網路的測試資料。由於單層感知器網路中以 10-10 訓練完可得到最小的絕對值誤差，我們利用 10-10 網路來進行模擬資料的測試。在這裡我們秀出第二十六組及第三十一組模擬的井測資料利用 10-10 網路做反推的結果。圖 3.5 為第二十六組井測資料，圖 3.6 則為第三十一組井測資料。圖 3.7 為第二十六組井測資料反推後的結果，其反推出來的 Ct 與預期的 Ct 之間的平均絕對值誤差為 0.021042。圖 3.8 為第三十一組井測資料反推後的結果，其反推出來的 Ct 與預期的 Ct 之間的平均絕對值誤差為 0.015689。

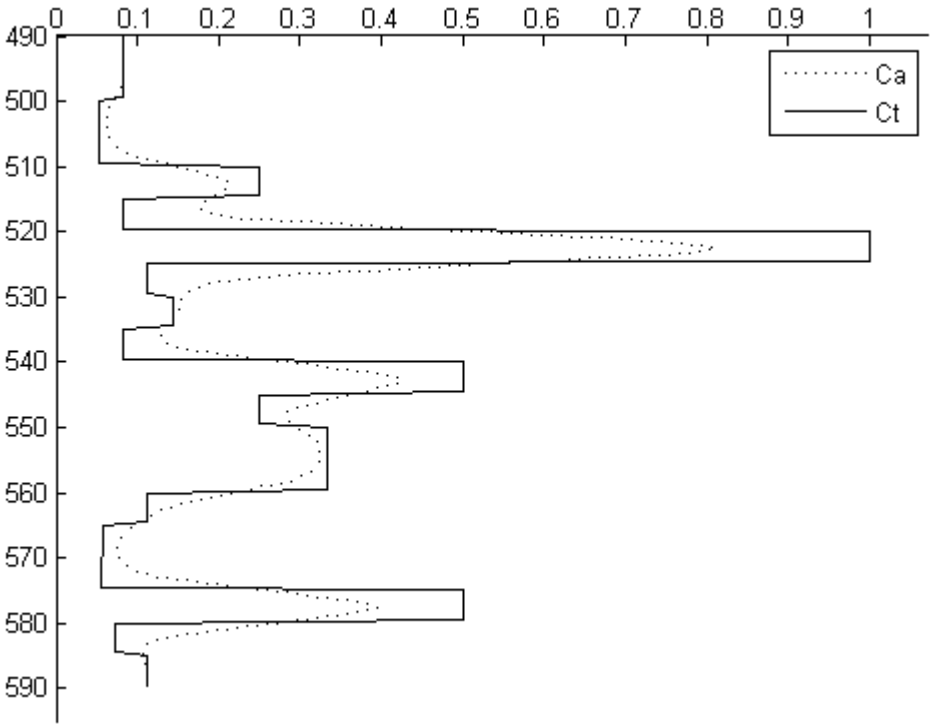


圖 3.5. 第二十六組井測資料的 Ca 與 Ct。

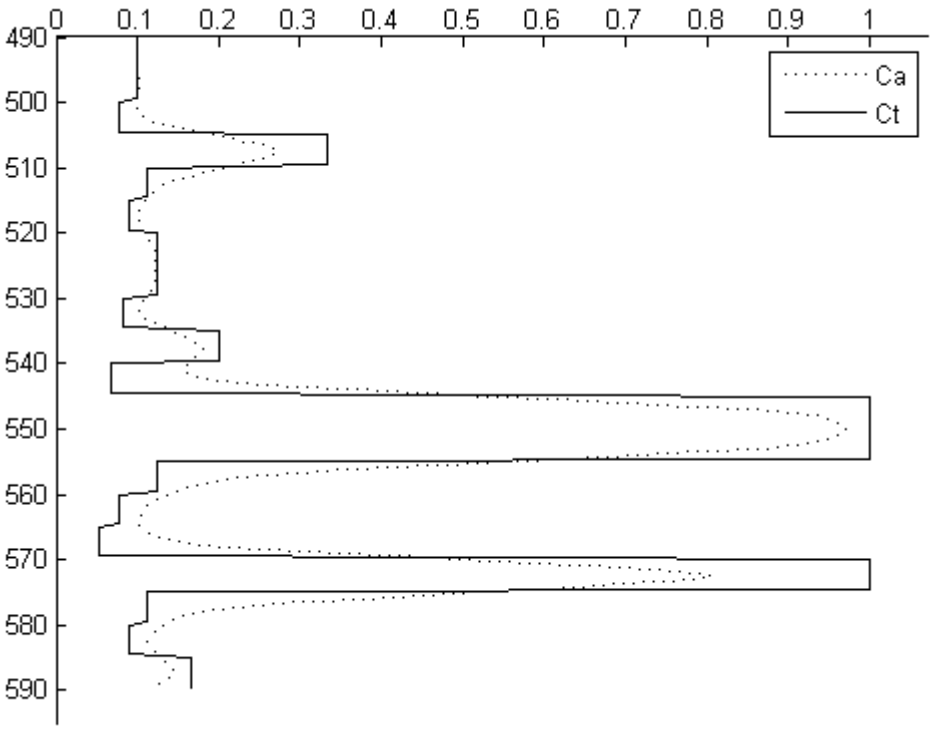


圖 3.6. 第三十一組井測資料的 Ca 與 Ct。

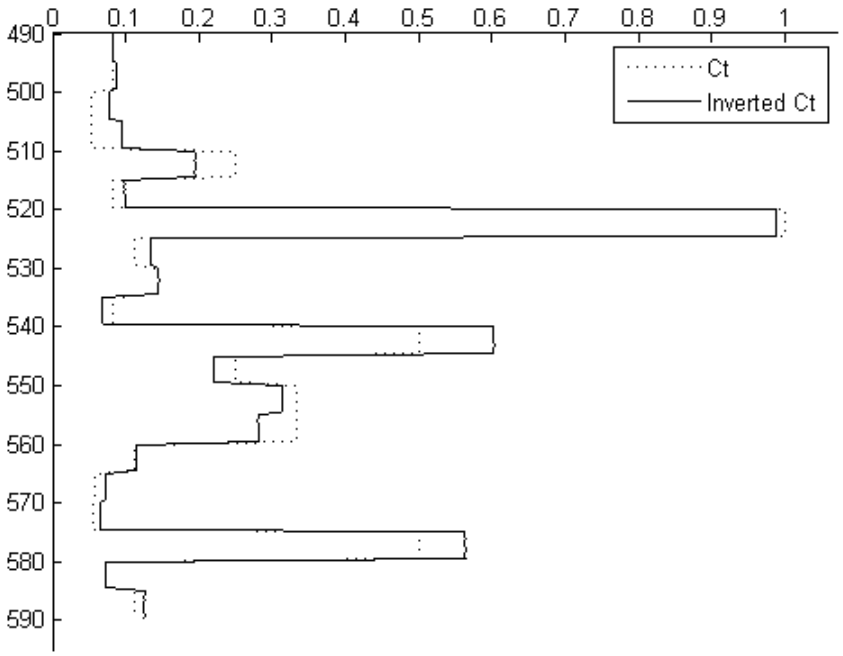


圖 3.7. 第二十六組井測資料利用 10-10 網路反推的 Ct 與期望的 Ct。

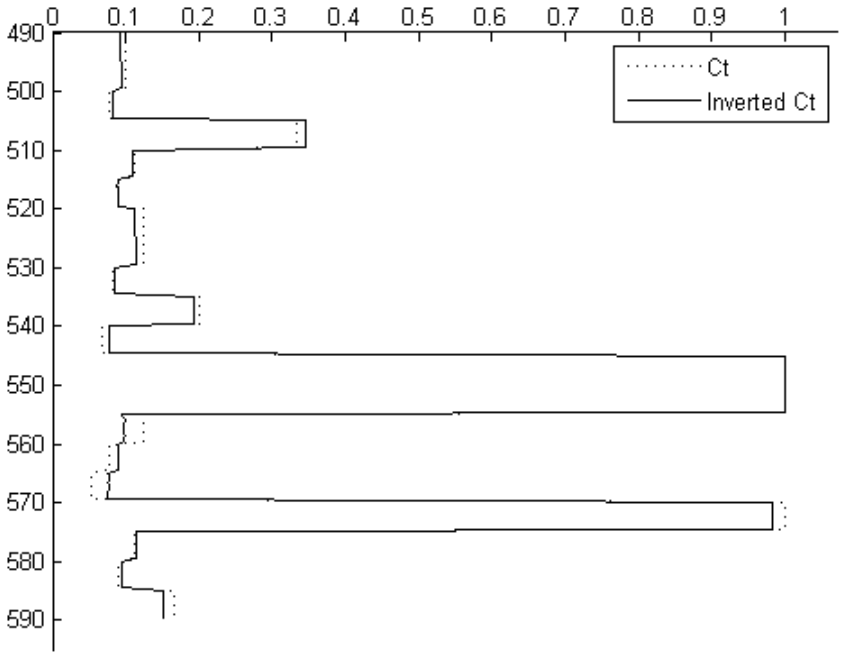


圖 3.8. 第三十一組井測資料利用 10-10 網路反推的 Ct 與期望的 Ct。

3.2 Results of Single Layer HONN

在這個實驗中，我們使用了單層的高階類神經網路 (HONN) 來執行模擬的井測資料反推。單層高階類神經網路的架構如圖 3.9 所示。網路的輸入節點個數 I 分別為 1、2、4、5、10、20、40、50、100、與 200。除此之外，網路的輸入分別包括輸入資料的二階特徵向量與三階特徵向量。利用二階特徵向量的實驗結果記錄在表 3.2 中，而三階特徵向量的實驗結果則記錄在表 3.3 中。圖 3.10 為網路大小為 20-10 時的訓練過程，圖 3.11 為網路大小為 30-10 時的訓練過程。

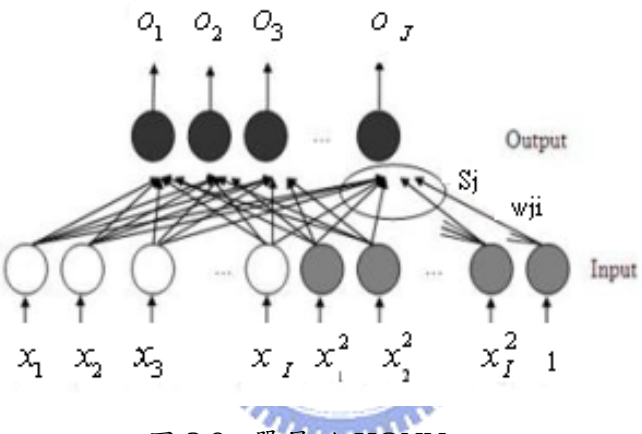


圖 3.9. 單層的 HONN。

表 3.2. 每一種單層二階高階類神經網路的實驗結果。

Network size	Number of training patterns	Error MAE at 20,000 iterations
2-1	5000	0.055296
4-2	2500	0.055375
8-4	1250	0.045936
10-5	1000	0.026701
20-10	500	0.013648
40-20	250	0.018926
80-40	125	0.020748
100-50	100	0.022741
200-100	50	0.023167
400-200	25	0.780082

表 3.3. 每一種單層三階高階類神經網路的實驗結果。

Network size	Number of training patterns	Error MAE at 20,000 iterations
3-1	5000	0.047405
6-2	2500	0.045860
12-4	1250	0.034748
15-5	1000	0.019202
30-10	500	0.008408
60-20	250	0.008766
120-40	125	0.010990
150-50	100	0.009827
300-100	50	0.011308
600-200	25	0.780082

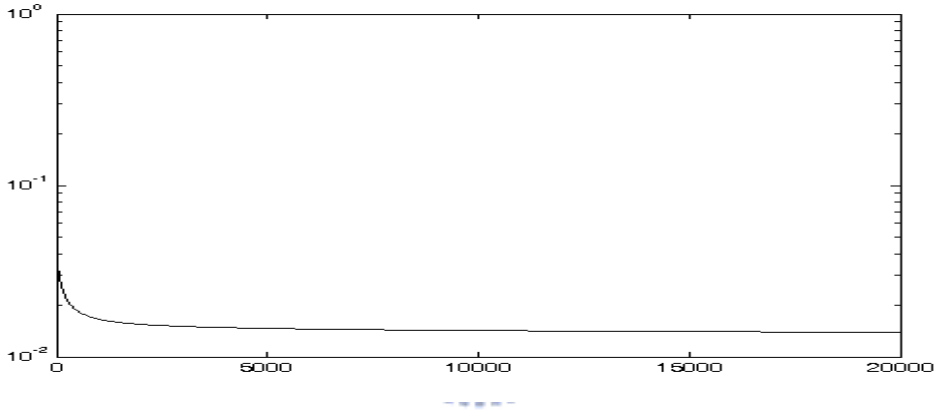


圖 3.10. 網路為 20-10 時的訓練過程。

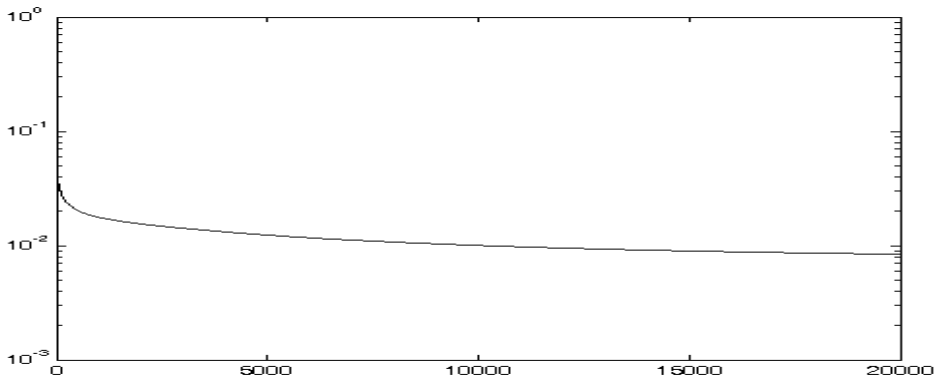


圖 3.11. 網路為 30-10 時的訓練過程。

當所有的網路訓練完之後，我們分別將第二十六組到第三十一組模擬資料當作網路的測試資料。由於高階單層感知器網路中以 30-10 訓練完可得到最小的絕對值誤差，我們利用 30-10 網路來進行模擬資料的測試。在這裡我們秀出第二十六組及第三十一組模擬的井測資料利用 30-10 網路做反推的結果。圖 3.12 為第二十六組井測資料利用 30-10 網路反推後的結果，其反推出來的 Ct 與預期的 Ct 之間的平均絕對值誤差為 0.00894。圖 3.13 為第三十一組井測資料利用 30-10 網路反推後的結果，其反推出來的 Ct 與預期的 Ct 之間的平均絕對值誤差為 0.008994。

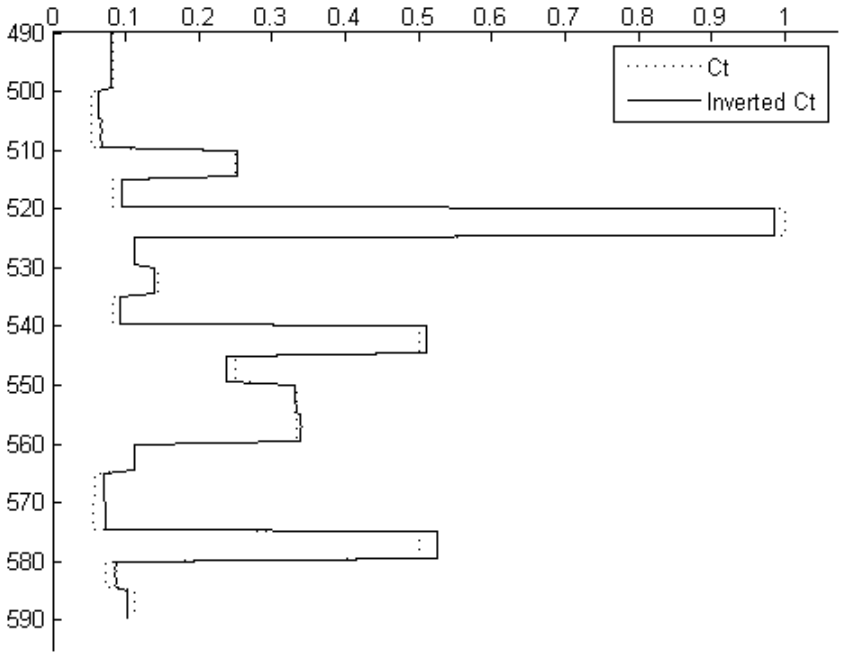


圖 3.12. 第二十六組井測資料利用 30-10 網路反推的 Ct 與期望的 Ct。

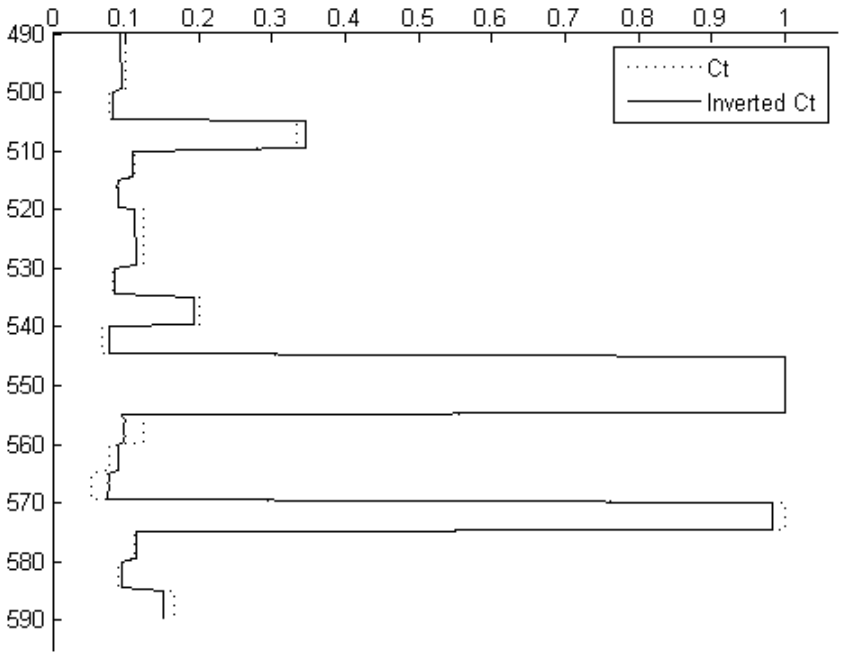


圖 3.13. 第三十一組井測資料利用 30-10 網路反推的 Ct 與期望的 Ct。

3.3 Results of Two Layer Neural Networks

3.3.1 Determination of the number of the hidden nodes

從單層感知器的實驗與單層的高階類神經網路的實驗得知，當網路的輸入節點數為 10 個的時候，可以得到最好的訓練效果。所以，我們將具有 10 個輸入節點與 10 個輸出節點的感知器網路另外再加上一層隱藏層，以期能達到更佳的訓練效果。關於隱藏層的節點個數，就採用定理 2.1 的方式來決定。

因為我們的實驗中，訓練樣本數為 500 (T=500)，而網路的輸入節點數為 10 (d=10)，根據定理 2.1 及 (2.16) 式，我們可以得到最多分割區域數 M 為：

H= 7, $M(7,10) = 2^7 = 128$ regions

H = 8, $M(8,10) = 2^8 = 256$ regions

H = 9, $M(9,10) = 2^9 = 512$ regions

H = 10, $M(10,10) = 2^{10} = 1,024$ regions

H = 11, $M(11,10) = 1024 + 11 = 1,035$ regions

H = 12, $M(12,10) = 1035 + 66 = 1,101$ regions

因為在訓練的時候，每一個區域至少要有一個訓練樣本，區域才能被分割出來，所以訓練樣本至少要有 M 個。因此我們能推測出網路的隱藏層需要 8 個或 9 個節點才能符合需求。

3.3.2 Experimental results

在這個實驗中，我們採用了兩層的類神經網路來執行模擬的井測資料反推，而隱藏層的節點分別為 7 個、8 個、9 個、10 個、11 個與 12 個，網路的架構如圖 3.14。這些網路訓練的實驗結果記錄在表 3.4。當網路的執行步驟設定為 20,000 時，我們發現 10-8-10 網路可以得到一個比之前利用單層的感知器網路或是單層的高階類神經網路還要小的絕對值誤差，然而當我們增加隱藏層的節點為 24 個的時候，網路訓練的誤差並不會就因此而變得更小。圖 3.15 為 10-8-10 網路的訓練過程，而圖 3.16 是利用 10-8-10 網路對第二十六組資料反推出來的結果，其反推出來的 Ct 與期望的 Ct 之間的平均絕對值誤差為 0.002602。圖 3.17 是利用 10-8-10 網路來對第三十一組資料反推出來的結果，其反推出來的 Ct 與期望的 Ct 之間的平均絕對值誤差為 0.002698。

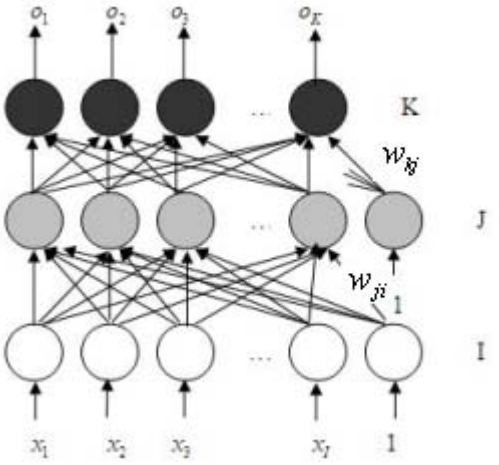


圖 3.14. 兩層的類神經網路。

表 3.4. 不同隱藏節點個數的兩層類神經網路的實驗結果

Network size	Number of training patterns	Error MAE at 20,000 iterations
10-7-10	500	0.002318
10-8-10	500	0.001921
10-9-10	500	0.002155
10-10-10	500	0.001996
10-11-10	500	0.002059
10-12-10	500	0.002141
10-24-10	500	0.002672

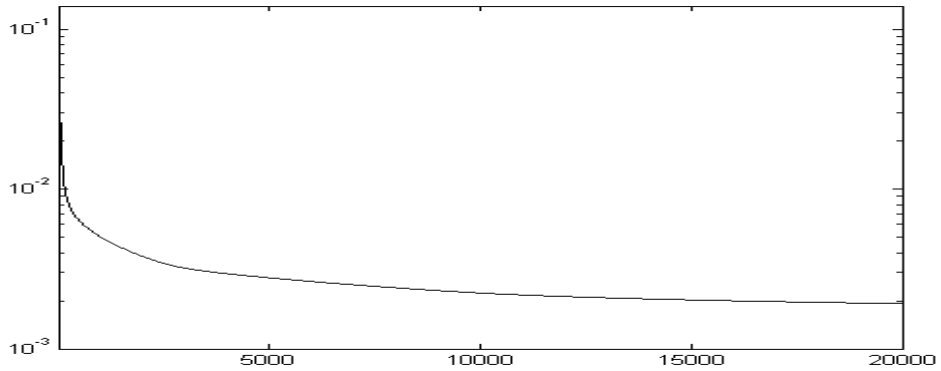


圖 3.15. 網路為 10-8-10 時的訓練過程。

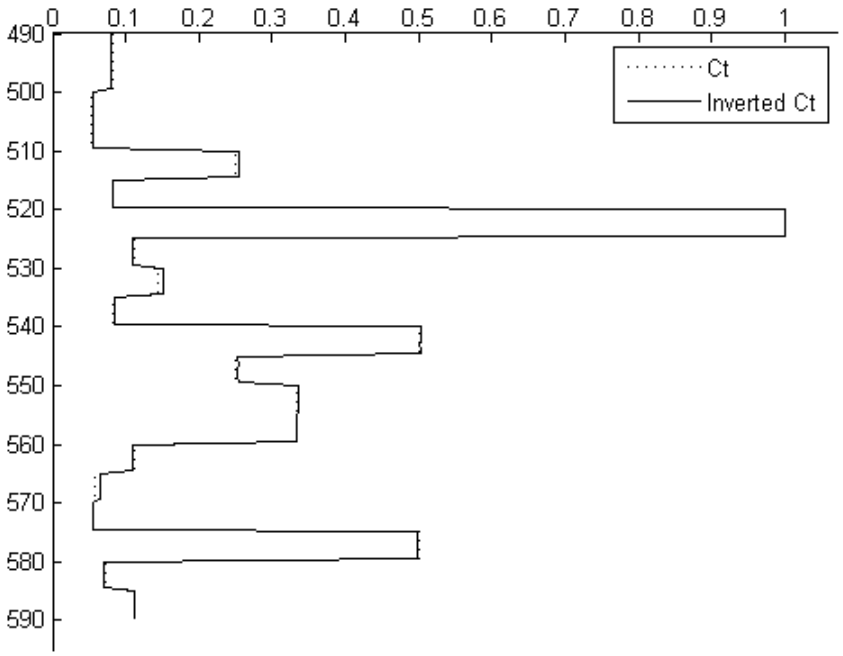


圖 3.16. 第二十六組井測資料利用 10-8-10 網路反推的 Ct 與期望的 Ct。

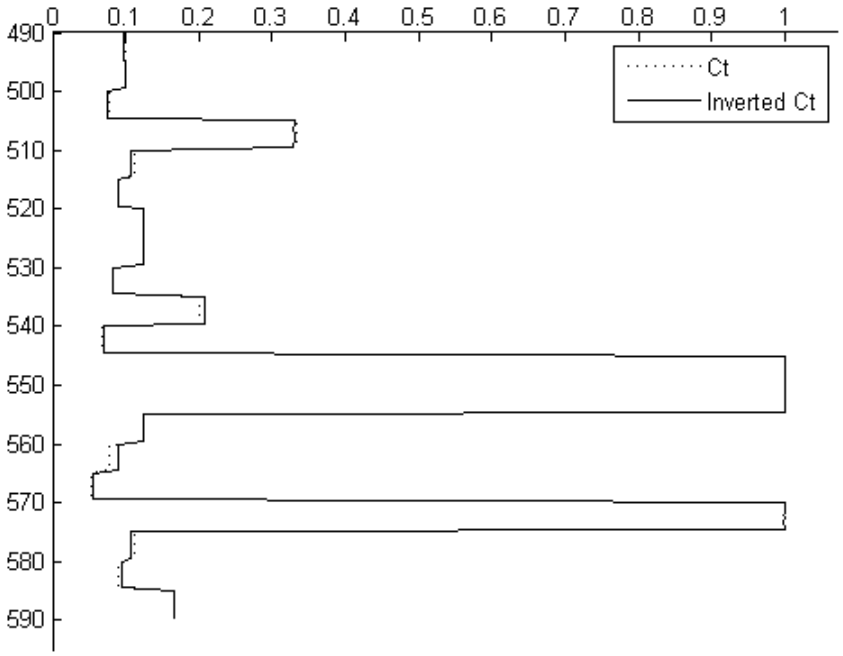


圖 3.17. 第三十一組井測資料利用 10-8-10 網路反推的 Ct 與期望的 Ct。

3.4 Results of Two Layer HONN

在這個實驗中，我們分別採用兩層的二階高階類神經網路與兩層的三階高階類神經網路來執行模擬的井測資料反推，而網路的隱藏層的節點個數分別為 7 個、8 個、9 個、10 個、11 個及 12 個。

3.4.1 Results of two layer with second order input features

兩層的二階高階類神經網路的實驗結果記錄在表 3.5。圖 3.18 為 20-8-10 網路的訓練過程，而圖 3.19 與圖 3.20 是利用 20-8-10 網路來分別對第二十六組資料與第三十一組資料做井測反推的結果。

表 3.5. 兩層的二階高階類神經網路實驗結果。

Network size	Number of training patterns	Error MAE at 20,000 iterations	Iterations of convergence (Threshold = 0.002)
20-7-10	500	0.002034	Over 20,000
20-8-10	500	0.001598	8,286
20-9-10	500	0.001649	8,880
20-10-10	500	0.001753	9,904
20-11-10	500	0.001994	13,740
20-12-10	500	0.002074	Over 20,000

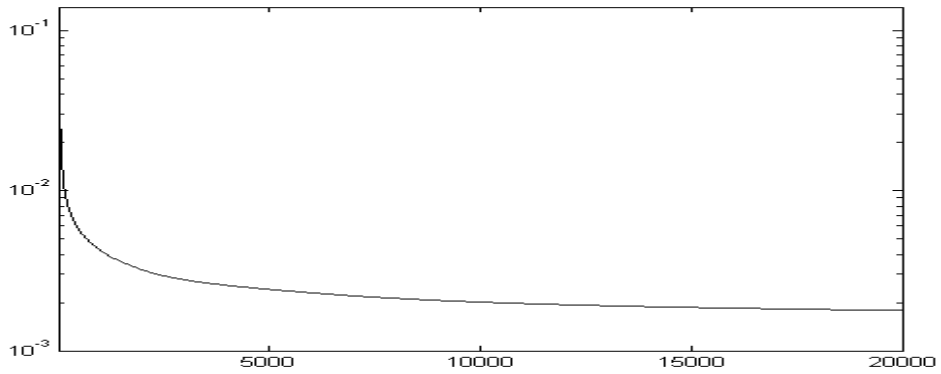


圖 3.18. 網路為 20-8-10 時的訓練過程。

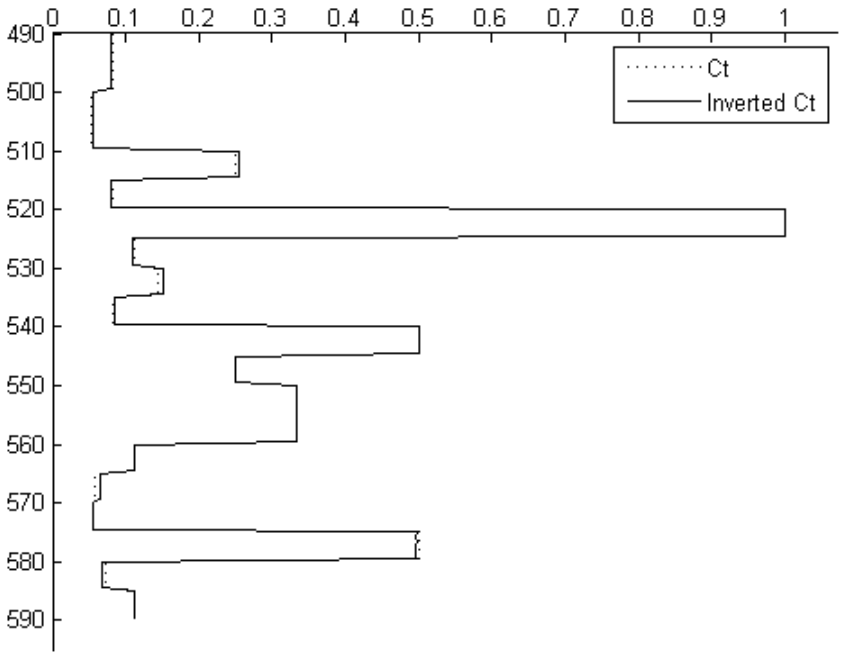


圖 3.19. 第二十六組井測資料利用 20-8-10 網路反推的 Ct 與期望的 Ct。

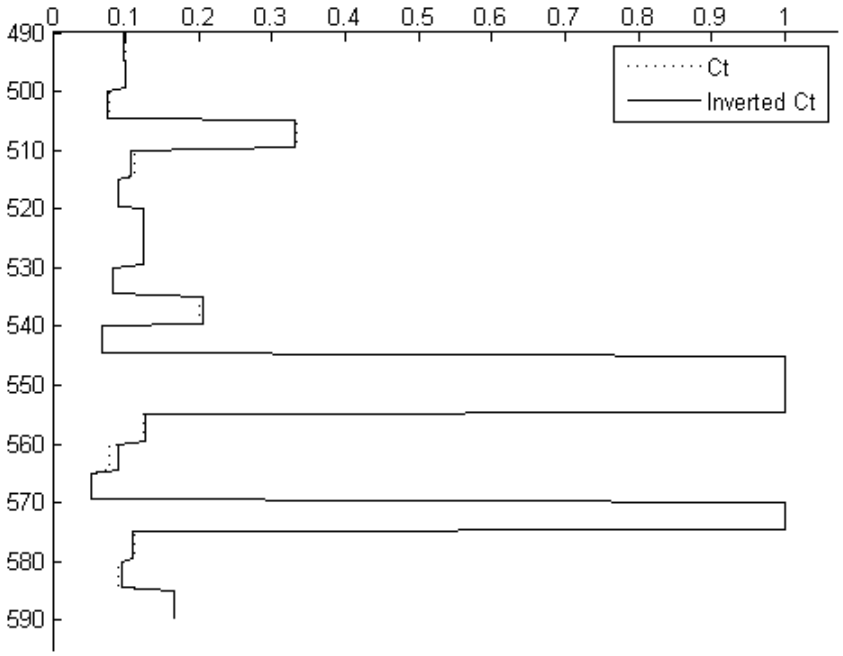


圖 3.20. 第三十一組井測資料利用 20-8-10 網路反推的 Ct 與期望的 Ct。

3.4.2 Results of two layer with third order input features

兩層的三階高階類神經網路的實驗結果紀錄在表 3.6。圖 3.21 為 30-8-10 網路的訓練過程，圖 3.22 與圖 3.23 是利用 30-8-10 網路來分別對第二十六組資料與第三十一組資料做井測反推的結果。因為利用 30-8-10 網路去訓練後所得到的平均絕對值誤差只有 0.001574，所以在圖 3.22 與圖 3.23 中，網路反推出來的 Ct 與期望的 Ct 是非常接近的。在比較表 3.5 及表 3.6 後，可以發現利用三階特徵向量來訓練網路，比利用二階特徵向量來訓練，可以在較少的執行步驟下便可以收斂。

表 3.6. 兩層的三階高階類神經網路實驗結果。

Network size	Number of training patterns	Error MAE at 20,000 iterations	Iterations of convergence (Threshold = 0.002)
30-7-10	500	0.002022	Over 20,000
30-8-10	500	0.001574	3,372
30-9-10	500	0.001719	4,386
30-10-10	500	0.001667	3,444
30-11-10	500	0.001808	11,427
30-12-10	500	0.001736	10,917

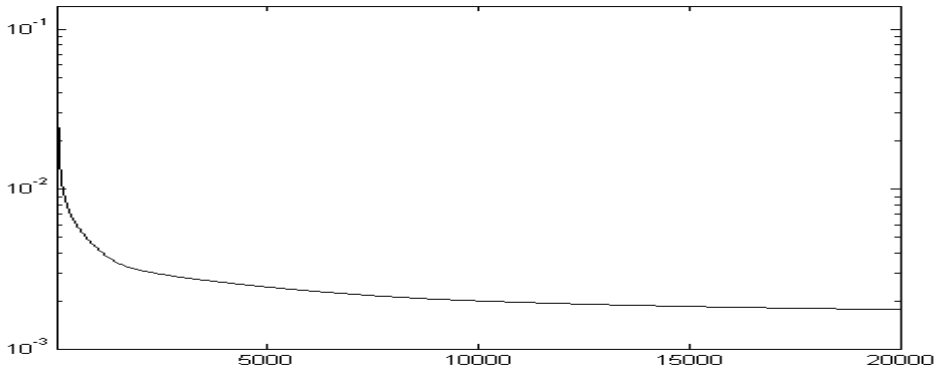


圖 3.21. 網路為 30-8-10 時的訓練過程。

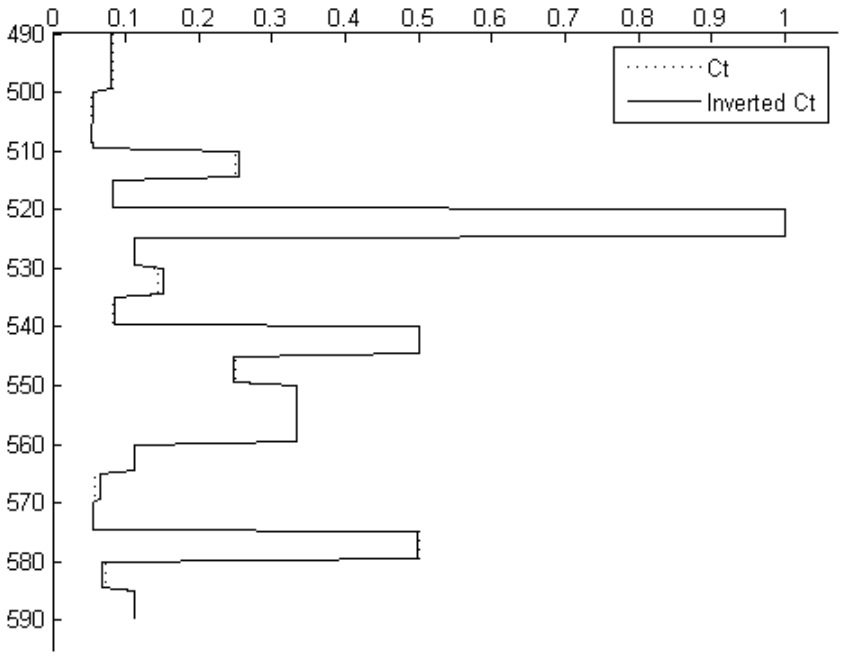


圖 3.22. 第二十六組井測資料利用 30-8-10 網路反推的 Ct 與期望的 Ct。

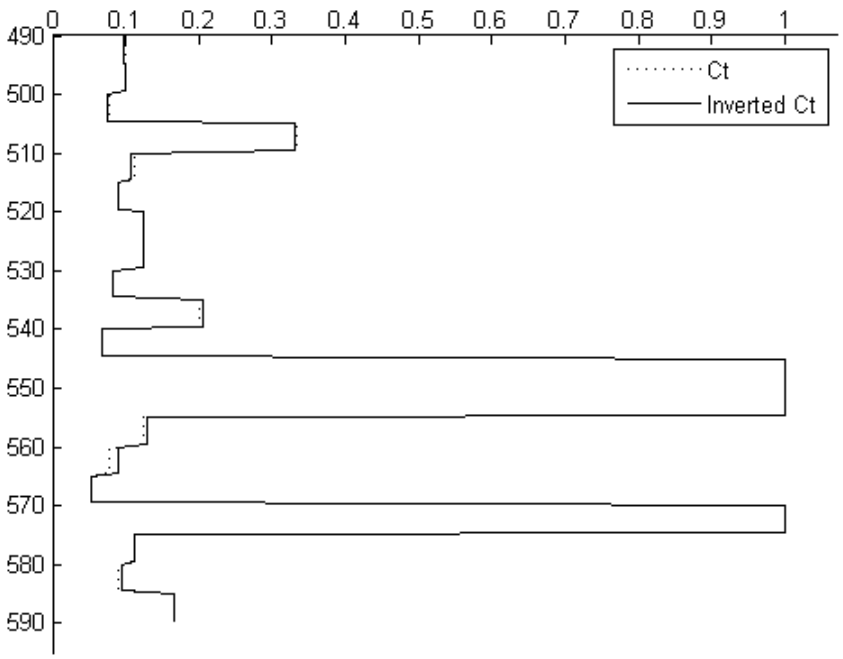


圖 3.23. 第三十一組井測資料利用 30-8-10 網路反推的 Ct 與期望的 Ct。

4. Differential Evolution for Well Log Data Inversion

差分進化法 (differential evolution, DE)中使用到符號在此先利用表 4.1 做整理：

表 4.1. 差分進化法中使用到的符號及其意義。

Symbol	Meaning
N_p	群組的大小
$\mathbf{x}_i^G$	群組中的一個個體 (i=1~ N_p)
L	個體的長度
G	群組的 index value，代表某一個世代
$\mathbf{x}_{BEST}^G$	第 G 代中的最佳個體
$\mathbf{v}_i^G$	由個體產生的突變體 (i=1~ N_p)
C_m	產生突變體時所用的參數
C_c	產生交配體時所用的參數
R_j	由 0~1 uniform random number 產生的一個數 (j=1~L)
$\mathbf{u}_i^G$	由個體與突變體做交配的交配體 (i=1~ N_p)
J	計算每一個個體適應值的函數
X	由個體所形成的群組，稱作群組矩陣
V	由突變體所形成的群組，稱作突變矩陣
U	由交配體所形成的群組，稱作交配矩陣

4.1 Introduction

差分進化法 (differential evolution, DE) 是由 R. Storn 與 K. Price [12] 在 1997 年所提出的一種全域最佳化的方法，能解決基因演算法 (genetic algorithm, GA)在 population size 較小的時候，會有效率不彰或是過早收斂而得到一個非最佳解的情況。同時 DE 還具有可以處理 non-differentiable、nonlinear 以及 multimodal cost function 的能力，此外還有可平行處理、簡單易於使用、良好的收斂效果, …, 等優點。

圖 4.1 為 DE 的執行過程的簡易流程圖。

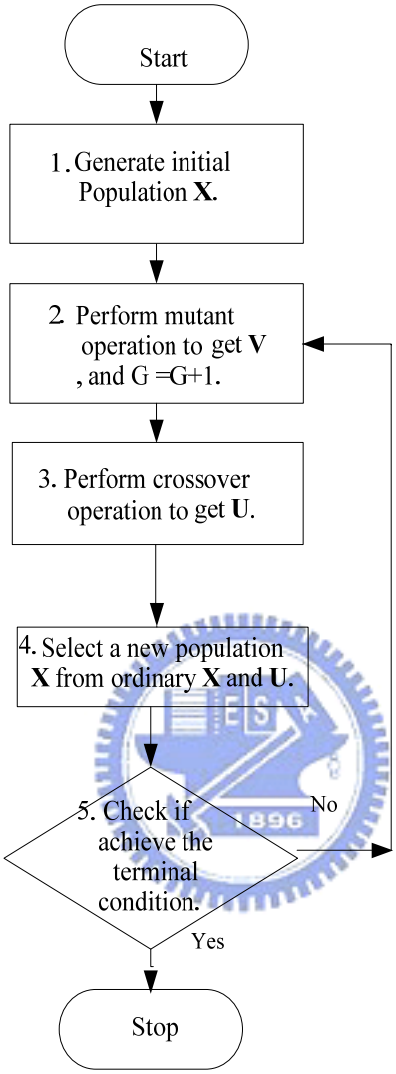


圖 4.1. 差分進化法的流程圖

假設我們有一個 fitness function J，若要以 DE 來對 fitness function J 做最佳化找尋其最大值，則以下為使用 DE 來做最佳化的步驟及每個步驟的詳細說明：

(1) 產生初始的 population

DE 的群組 (population) **X** 有多個個體 (individuals)，每一個個體包含了 L 個 parameters。表示法如下：

DE 的初始群組所形成的群組矩陣 **X**

$$
\begin{array}{c|cccccc}
\mathbf{x}_1^0 & x_{1,1}^0 & x_{1,2}^0 & x_{1,3}^0 & \cdots & x_{1,L-1}^0 & x_{1,L}^0 \\
\mathbf{x}_2^0 & x_{2,1}^0 & x_{2,2}^0 & x_{2,3}^0 & \cdots & x_{2,L-1}^0 & x_{2,L}^0 \\
\mathbf{x}_3^0 & x_{3,1}^0 & x_{3,2}^0 & x_{3,3}^0 & \cdots & x_{3,L-1}^0 & x_{3,L}^0 \\
\vdots & & & & & & \vdots \\
\mathbf{x}_{NP-2}^0 & x_{NP-2,1}^0 & x_{NP-2,2}^0 & \cdots & x_{NP-2,L-1}^0 & x_{NP-2,L}^0 \\
\mathbf{x}_{NP-1}^0 & x_{NP-1,1}^0 & x_{NP-1,2}^0 & \cdots & x_{NP-1,L-1}^0 & x_{NP-1,L}^0 \\
\mathbf{x}_{NP}^0 & x_{NP,1}^0 & x_{NP,2}^0 & \cdots & x_{NP,L-1}^0 & x_{NP,L}^0
\end{array}
$$

where $\mathbf{x}_i^G, i = 1, 2, 3, \ldots, N_p \; ; \; x_{i,j}^G, j = 1, 2, 3, \ldots, L$

其中，$\mathbf{x}_i^G$ 就是一個個體，G 為 DE 中群組的 index value (初始代為 0，接下來為第一代則 G=1，第二代則 G=2，依此類推)。N_p 為群組的大小, N 為 parameter 的維度。初始的群組可利用一個-1~1 的 uniform random number 來產生。

(2) mutant

一個突變體 $\mathbf{v}_i^G$ is generated from:

$$\mathbf{v}_i^G = \mathbf{x}_{BEST}^G + C_m(\mathbf{x}_m^G - \mathbf{x}_n^G); \quad i = 1, 2, 3, \ldots, N_p$$

$\mathbf{x}_{BEST}^G$指的是在第 G 代中的最佳個體，而$\mathbf{x}_m^G$、$\mathbf{x}_n^G$則是第 G 代中利用 uniform random number 從 N_p 中任意選擇的兩個個體。{BEST, m, n}三個數必須彼此不同 (mutually different)，其中參數C_m是使用者自訂的一個數。(C_m的意義類似權重，若C_m越大代表突變體 $\mathbf{v}_i^G$ 受到 $\mathbf{x}_m^G - \mathbf{x}_n^G$ 的影響越大；反之C_m越小，代表 $\mathbf{v}_i^G$受到$\mathbf{x}_m^G - \mathbf{x}_n^G$的影響越小)

以下為一個 DE 如何執行突變的例子：

假設群組 **X** 的大小為 100 (N_p=100)，每一個個體有 9 個參數 (N=9)，這裡要產生第一個突變體 $\mathbf{v}_1^0$ 而第 0 代的最佳個體 $\mathbf{x}_{BEST}^0$ 為：

$\mathbf{x}_{BEST}^0$，假設 BEST 為第 50 個個體

0.9	0.9	0.9	0.9	0.9	0.9	0	0.5	0.9

而利用 uniform random number 產生的兩個數 m 與 n 為 12 與 79，其個體 $\mathbf{x}_{12}^0$ 與 $\mathbf{x}_{79}^0$ 分別如下：

$\mathbf{x}_{12}^0$

0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9

$\mathbf{x}_{79}^0$

0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1

則產生一個突變體 $\mathbf{v}_i^G = \mathbf{x}_{BEST}^G + C_m(\mathbf{x}_m^G - \mathbf{x}_n^G)$，因為 $G = 0$，$BEST = 50$，m=12，n=79，在此假設使用者自訂數 C_m 為 1，所以產生的突變體 $\mathbf{v}_1^0$ 為：

0.1	0.3	0.5	0.7	0. 9	1.1	0.4	1.1	1.7

利用相同的方式，產生與族群大小 N_p 相同數量的突變體 $\mathbf{v}_1^G, \mathbf{v}_2^G, \mathbf{v}_3^G, \ldots, \mathbf{v}_{Np}^G$ 即完成第 G 代 (此處假設為初始代，所以 G=0) 的 mutant 步驟，產生一個與個體所形成的群組矩陣 **X** 類似的突變體矩陣 **V**。

第 0 代的突變體矩陣 **V**

$\mathbf{v}_1^0$	$v_{1,1}^0$	$v_{1,2}^0$	$v_{1,3}^0$	…	$v_{1,L-1}^0$	$v_{1,L}^0$
$\mathbf{v}_2^0$	$v_{2,1}^0$	$v_{2,2}^0$	$v_{2,3}^0$	…	$v_{2,L-1}^0$	$v_{2,L}^0$
$\mathbf{v}_3^0$	$v_{3,1}^0$	$v_{3,2}^0$	$v_{3,3}^0$	…	$v_{3,N-1}^0$	$v_{3,N}^0$
⋮						⋮
$\mathbf{v}_{NP-2}^0$	$v_{NP-2,1}^0$	$v_{NP-2,2}^0$	…	$v_{NP-2,L-1}^0$	$v_{NP-2,L}^0$	
$\mathbf{v}_{NP-1}^0$	$x_{NP-1,1}^0$	$v_{NP-1,2}^0$	…	$v_{NP-1,L-1}^0$	$v_{NP-1,L}^0$	
$\mathbf{v}_{NP}^0$	$v_{NP,1}^0$	$v_{NP,2}^0$	…	$v_{NP,L-1}^0$	$v_{NP,L}^0$	

(3) Crossover

一個交配體 $\mathbf{u}_i^G$ is generated from:

$$u_{i,j}^G = \begin{cases} v_{i,j}^G & \text{for } R_j \leq C_c \\ x_{i,j}^G & \text{otherwise} \end{cases} \qquad (4.1)$$

where $i = 1, 2, 3, \ldots, N_P$; $j = 1, 2, 3, \ldots, N$

其中 R_j 為一個 0~1 的 uniform random number，C_c 為一個 crossover 常數 (C_c 的意義為決定交配體 $\mathbf{u}_i^G$ 中的 element $u_{i,j}^G$ 該從 $\mathbf{x}_i^G$ 中的 $x_{i,j}^G$ 或是從 $\mathbf{v}_i^G$ 中的 $v_{i,j}^G$ 來，若 C_c 設定相當大，則根據 (4.1) 式，R_j 越容易小於 C_c，則交配體 $\mathbf{u}_i^G$ 的 element $u_{i,j}^G$ 便會由 $v_{i,j}^G$ 而來，反之則從 $x_{i,j}^G$ 而來)，j 為參數的 index。

以下 DE 執行如何 crossover 的一個例子：

假設仍然沿用突變例子的結果，在初始代 (G = 0) 且已經完成突變之後，對第 1 個個體 $\mathbf{x}_1^0$ 來說，可找到與其對應的突變體 $\mathbf{v}_1^0$，則 $\mathbf{x}_1^0$ 與 $\mathbf{v}_1^0$ 如下：

$\mathbf{x}_1^0$

0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

$\mathbf{v}_1^0$

0.1	0.3	0.5	0.7	0. 9	1.1	0.4	1.1	1.7

若 C_c 設定為 0.6，而我們利用 0~1 的 uniform random number 來產生 $R_1, R_2, R_3, \cdots, R_9$，假設產生的 R_1 = 0.5, R_2= 0.9, R_3 = 0.6, R_4 = 0.3, R_5 = 0.5, R_6= 0.9, R_7 = 0.7, R_8 = 0.8, R_9= 0.1。根據 (4.1) 的關係式，我們可以得到由 $\mathbf{x}_1^0$與 $\mathbf{v}_1^0$去執行 crossover 後的 $\mathbf{u}_1^0$，過程如下：

∵ $R_1 \leq C_c$ ∴ $u_{1,1}^0 = v_{1,1}^0 = 0.1$

∵ $R_2 > C_c$ ∴ $u_{1,2}^0 = x_{1,2}^0 = 0.1$,

同理，可求得剩下的 $u_{1,3}^0$, $u_{1,4}^0$, $\cdots$, $u_{1,9}^0$

$\mathbf{u}_1^0$

0.1	0.1	0.5	0.7	0.9	0.1	0.1	0.1	1.7

利用相同的方式，產生與族群大小 N_p 相同數量的交配體 $\mathbf{u}_1^G, \mathbf{u}_2^G, \mathbf{u}_3^G, \ldots, \mathbf{u}_{Np}^G$即完成第 G 代 (此處假設為初始代，所以 G=0) 的 crossover 步驟，產生一個與個體所形成的群組矩陣 **X** 類似的交配矩陣 **U**。

第 0 代的交配體矩陣 **U**

$\mathbf{u}_1^0$	$u_{1,1}^0$	$u_{1,2}^0$	$u_{1,3}^0$	⋯	$u_{1,L-1}^0$	$u_{1,L}^0$
$\mathbf{u}_2^0$	$u_{2,1}^0$	$u_{2,2}^0$	$u_{2,3}^0$	⋯	$u_{2,L-1}^0$	$u_{2,L}^0$
$\mathbf{u}_3^0$	$u_{3,1}^0$	$u_{3,2}^0$	$u_{3,3}^0$	⋯	$u_{3,L-1}^0$	$u_{3,L}^0$
⋮						⋮
$\mathbf{u}_{NP-2}^0$	$u_{NP-2,1}^0$	$u_{NP-2,2}^0$	⋯	$u_{NP-2,L-1}^0$	$u_{NP-2,L}^0$	
$\mathbf{u}_{NP-1}^0$	$u_{NP-1,1}^0$	$u_{NP-1,2}^0$	⋯	$u_{NP-1,L-1}^0$	$u_{NP-1,L}^0$	
$\mathbf{u}_{NP}^0$	$u_{NP,1}^0$	$u_{NP,2}^0$	⋯	$u_{NP,L-1}^0$	$u_{NP,L}^0$	

(4) Selection

根據群組的大小 N_p，選擇 N_p 個合適的個體，讓它們能夠進入下一個世代 (從第 G 代至 G+1 代)。對第 G 代的一個個體 $\mathbf{x}_i^G$，選擇的方式如下：

$$\mathbf{x}_i^{G+1} = \begin{cases} \mathbf{u}_i^G & \text{for } J(\mathbf{u}_i^G) \geq J(\mathbf{x}_i^G) \\ \mathbf{x}_i^G & \text{otherwise} \end{cases} \tag{4.2}$$

(4.2) 式中，J為 fitness function，$J(\mathbf{u}_i^G)$ 代表將 $\mathbf{u}_i^G$代入 fitness function 所得到的 fitness value，而 $J(\mathbf{x}_i^G)$ 是將 $\mathbf{x}_i^G$代入 fitness function 所得到的 fitness value，比較兩個個體 $\mathbf{u}_i^G$、$\mathbf{x}_i^G$的 fitness value 的大小。

因為目標是將 fitness function 最大化，則選擇有較大的 fitness value 的個體 (若目標是將 fitness function 最小化，則將 (4.2) 式的大於改成小於即可)。以下是一個 DE 如何做個體選擇的例子：

假設要從初始代的個體 $\mathbf{x}_i^0$中選擇能成為第一代的個體 $\mathbf{x}_i^1$，在這裡沿用之前已經產生過的 $\mathbf{x}_1^0$ 、$\mathbf{u}_1^0$ 以及假設一個個體 $\mathbf{x}_2^0$與同樣以 (4.1) 的方式所產生的 u_2^0 如下：

$\mathbf{u}_1^0$

0.1	0.1	0.5	0.7	0.9	0.1	0.1	0.1	1.7

$\mathbf{u}_2^0$

0.4	-0.1	0.7	-0.2	0.1	0.5	-0.4	0.3	-1.2

$\mathbf{x}_1^0$

0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

$\mathbf{x}_2^0$

0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2

若 fitness function J 在此為：

$$J(\mathbf{x}) = \sum_{i=1}^{L} x_i$$

所以 $J(\mathbf{u}_1^0) = 4.3$，$J(\mathbf{u}_2^0) = 0.1$

$J(\mathbf{x}_1^0) = 0.9$，$J(\mathbf{x}_2^0) = 1.8$

根據 (4.2) 的關係式，可以得到

$J(\mathbf{u}_1^0) > J(\mathbf{x}_1^0) \therefore \mathbf{x}_1^1 = \mathbf{u}_1^0$

0.1	0.1	0.5	0.7	0.9	0.1	0.1	0.1	1.7

$J(\mathbf{u}_2^0) < J(\mathbf{x}_2^0) \therefore \mathbf{x}_2^1 = \mathbf{x}_2^0$

0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2

利用相同的方式，產生與族群大小 N_p 相同數量的下一世代個體 $\mathbf{x}_1^{G+1}, \mathbf{x}_2^{G+1}, \mathbf{x}_3^{G+1}, \ldots, \mathbf{x}_{N_p}^{G+1}$ 即完成第 G 代 (此處假設為初始代，所以 G=0) 的 selection 步驟，產生第 G+1 代的群組矩陣 **X**。

DE 的第一代所形成的群組矩陣 **X**

$\mathbf{x}_1^1$	0.1	0.1	0.5	…	0.1	1.7
$\mathbf{x}_2^1$	0.2	0.2	0.2	…	0.2	0.2
$\mathbf{x}_3^1$	$x_{3,1}^1$	$x_{3,2}^1$	$x_{3,3}^1$	…	$x_{3,L-1}^1$	$x_{3,L}^1$
⋮						⋮
$\mathbf{x}_{NP-2}^1$	$x_{NP-2,1}^1$	$x_{NP-2,2}^1$	…	$x_{NP-2,L-1}^1$	$x_{NP-2,L}^1$	
$\mathbf{x}_{NP-1}^1$	$x_{NP-1,1}^1$	$x_{NP-1,2}^1$	…	$x_{NP-1,L-1}^1$	$x_{NP-1,L}^1$	
$\mathbf{x}_{NP}^1$	$x_{NP,1}^1$	$x_{NP,2}^1$	…	$x_{NP,L-1}^1$	$x_{NP,L}^1$	

(5) 檢查終止條件

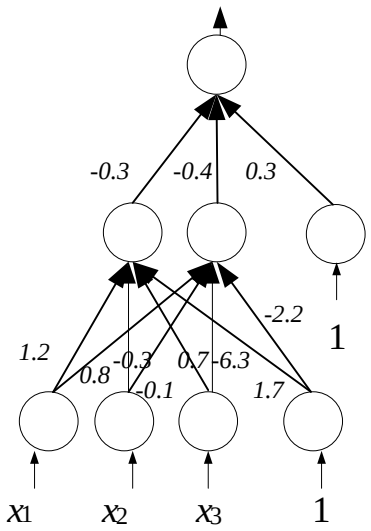
若終止條件是利用 DE 的演化代數 G 做為終止條件，則檢查是否已經達到設定的最大世代，亦即 DE 的執行次數。若沒有達到，則再重複步驟 (1) 至 (4)，直到終止條件滿足為止。

4.2 Encode Weights of Neural Networks

當我們使用差分進化法 (differential evolution, DE) 來對網路做訓練之前，首先要先將網路的所有權重 (weight) 編碼成 DE 的個體。由於網路中的權重為實數，所以我們採用由 Montana 與 Davis [10] 所提出的一種實數編碼方式來對網路的權重做編碼。DE 群組中每一個個體 $\mathbf{x}_i^G$ 中的參數 $x_{i,j}^G$ 都是一個權重值，而被編碼的網路的大小就等於每一個個體的長度 N，假如網路的輸入節點個數為 I，隱藏層的節點個數為 J，而輸出節點數的個數為 K，則對應的個體長度 L 為 ：

$$L = ((I+1)*J+(J+1)*K) \tag{4.3}$$

圖 4.2 為一個編碼例子：



1.2	0.8	-0.3	-0.1	0.7	-6.3	1.7	-2.2	-0.3	-0.4	0.3

圖 4.2. 將一個類神經網路編碼成一個個體。

圖 4.2 中網路的大小為 3-2-1，所以個體的長度 L 為 ((3+1)*2+(2+1)*1) = 11，即 DE 中個體 $\mathbf{x}_i^G$ 的長度 L 為 11。

4.3 Combination of Differential Evolution and Neural Networks

4.3.1. Training process

在這個實驗中，我們將利用差分進化法來對兩層的二階、三階的高階類神經網路作前饋式 (feed forward) 的訓練，而利用這種訓練方式的網路我們在此稱做 DENN (combining differential evolution and neural networks, DENN)。網路的輸入與第2.3節相同，是視導電率 (apparent conductivity, Ca)，而網路的期望輸出值則是地層真實導電率 (true formation conductivity, Ct)。

我們一共有三十一組模擬的井測資料，拿第一組到第二十五組的模擬資料來做網路的訓練，第二十六組到第三十一組來做測試。每一組模擬的資料的油井深度是從 490 英呎到 589.5 英呎，而取樣區間 (sample interval) 是 0.5 英呎。

所以每一組模擬的井測資料一共有兩百個模擬的視導電率與其對應的地層真實導電率。

我們先將模擬的訓練資料正規化 (normalized) 在 0.1 到 0.9 之間，接著再分別輸入網路的輸入特徵向量擴展到二階的高階類神經網路，及輸入特徵向量擴展到三階的高階類神經網路。而隱藏層的節點則分別為 7 個、8 個、9 個、10 個、11 個、12 個、13 個與 14 個，所以利用差分進化法的來訓練網路時，其個體長度 L 與網路的大小不同而有所不同，例如：

若網路為將輸入特徵向量擴展到二階的 20-8-10 網路，則個體長度 L 就為：

$$((20+1)*8+(8+1)*10) = 258$$

若網路為將輸入特徵向量擴展到三階的 30-8-10 網路，則個體長度 L 就為：

$$((30+1)*8+(8+1)*10) = 338$$

實驗中，差分進化法的群組數目 N_p 設定為 100，所以每一個世代 G 中一共有 100 個與其個體長度 L 大小相對應的類神經網路。初始群組 (G = 0) 則利用 -1~1 的 uniform random number 來產生，演化的停止世代設定為 2,000 (G=2,000) 代，而 fitness value 依然使用平均絕對值誤差 (mean absolute error, MAE) 來做為評估此個體的合適度，這裡的平均絕對值誤差定義與 (2.17) 式相同。差分進化法中的使用者自定參數 P 與 crossover 常數 C 都設定為 0.6，而誤差值的門檻則設定為 0.002。訓練完之後，取出最後一個世代中具有最佳適合度的個體，即一個利用差分進化法訓練網路時，在最後一個世代中，具有最小 MAE 的網路。再利用此最佳的網路對第二十六組至第三十一組的模擬資料去做測試得到網路的輸出值，最後再將這些輸出值經過還原數值 (re-scaled) 的步驟，即可以得到由差分進化法訓練出的類神經網路反推出來的一組真實地層導電率 (true formation conductivity)。

4.3.2. Experimental results

在這個實驗中，我們分別採用兩層的二階高階 DENN 與兩層的三階高階 DENN 來執行模擬的井測資料反推，而網路的隱藏層的節點個數分別為 7 個、8 個、9 個、10 個、11 個、12 個、13 個及 14 個，而每一種網路都執行十次，一次的意思為利用第一組到第二十五組井測資料訓練完之後得到一個訓練的結果，所以執行十次每一種網路就會有十個訓練結果。每一種網路都可以得到由差分進化法訓練後的 MAE 十次結果中的 best case、worst case 及平均這十次結果後所得到的 average case。兩層的二階高階 DENN 與兩層的三階高階 DENN 的實驗結果分別記錄在表 4.2 及表 4.3 中，表 4.4 為兩個實驗的比較。圖 4.3 為利用差分進化法對 20-8-10 的網路進行訓練所得到的一個訓練過程。圖 4.4 為 30-8-10 的網路進行訓練所得到的一個訓練過程，圖 4.5 中我們額外加入了利用基因演算法來訓練 30-8-10 的網路所得到的訓練過程來與圖 4.3 及圖 4.4 的結果做比較，而使用基因演算法來訓練的 GANN (combination of Genetic Algorithm and Neural Network) 其群組大小、個體初始值跟終止世代與本實驗中 DENN 的設定相同，而 crossover rate 為 1，mutation rate 為 0.001，選擇的方式採用輪盤式的選擇法則。

表 4.2. 兩層的二階高階 DENN 的實驗結果。

Network Size	AVG. MAE (Training)	Best MAE (Training)	Worst MAE (Training)
20-7-10	0.0323	0.023525	0.037846
20-8-10	0.0306	0.023145	0.042793
20-9-10	0.0243	0.023782	0.024748
20-10-10	0.0265	0.020317	0.034665
20-11-10	0.0245	0.017785	0.032921
20-12-10	0.0257	0.016757	0.040358
20-13-10	0.0248	0.019336	0.031763
20-14-10	0.0253	0.021870	0.028537

表 4.3. 兩層的三階高階 DENN 的實驗結果。

	AVG. MAE (Training)	Best MAE (Training)	Worst MAE (Training)
30-7-10	0.0301	0.022500	0.042608
30-8-10	0.0279	0.021167	0.038790
30-9-10	0.0233	0.019851	0.025672
30-10-10	0.0245	0.020484	0.032948
30-11-10	0.0228	0.020710	0.024176
30-12-10	0.0250	0.018239	0.044519
30-13-10	0.0229	0.016875	0.031450
30-14-10	0.0221	0.016688	0.027457

表 4.4. 兩層的二階高階 DENN 與三階高階 DENN 的 AVG. MAE 比較

Number of hidden node	AVG. MAE of second order DENN	AVG. MAE of third c DENN
7	0.0323	0.0301
8	0.0306	0.0279
9	0.0243	0.0233
10	0.0265	0.0245
11	0.0245	0.0228
12	0.0257	0.0250
13	0.0248	0.0229
14	0.0253	0.0221

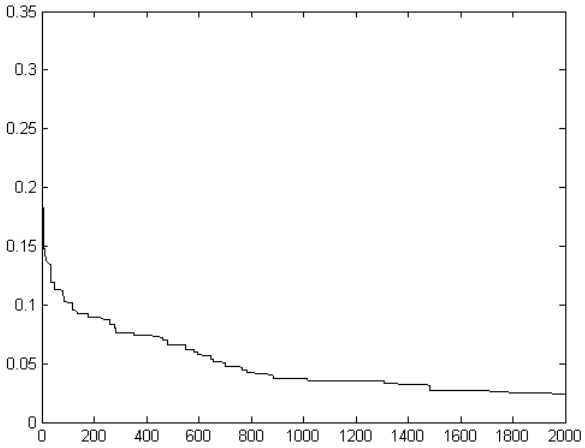


圖 4.3. 利用差分進化法對 20-8-10 的網路進行訓練所得到的訓練過程。

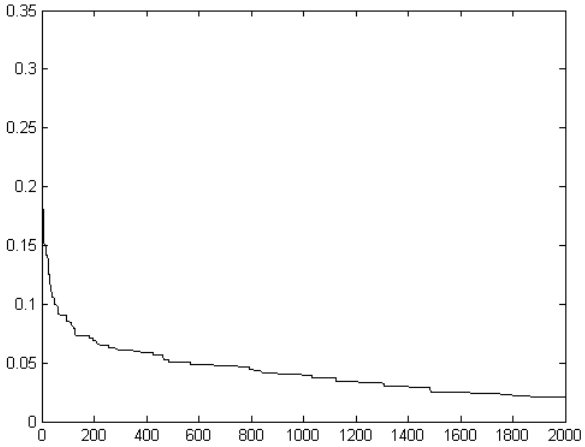


圖 4.4. 利用差分進化法對 30-8-10 的網路進行訓練所得到的訓練過程。

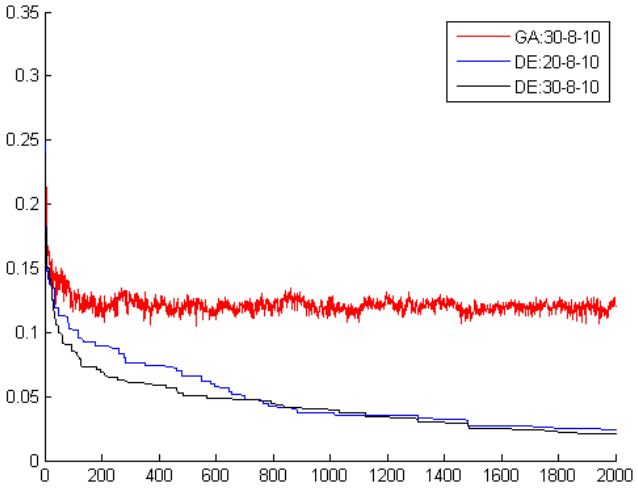


圖 4.5. 比較圖 4.3、圖 4.4 及利用 GA 對 30-8-10 的網路進行訓練所得到的訓練過程。

當所有的網路訓練完之後，我們利用具有最小平均誤差的 30-14-10 DENN 網路分別對第二十六組資料與第三十一組資料做井測反推。圖 4.6 與圖 4.7 是利用 best case 的 30-14-10 DENN 來分別對第二十六組資料與第三十一組資料做井測反推的結果。

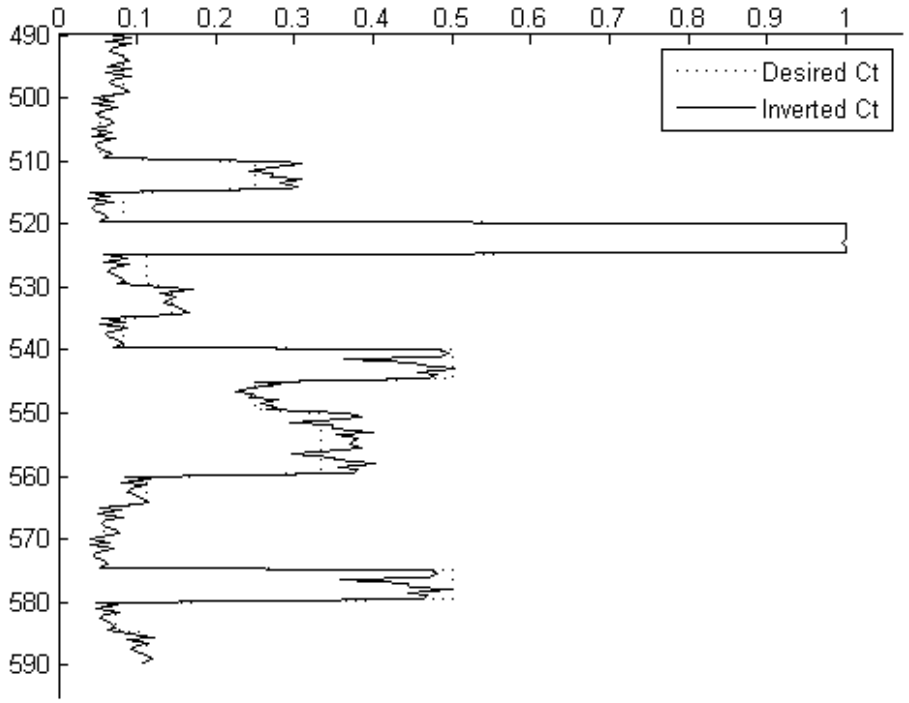


圖 4.6. 第二十六組井測資料利用 best case 30-14-10 DENN 反推的 Ct 與期望的 Ct。其反推出來 Ct 與期望的 Ct 之間的平均絕對值誤差為 0.019948。

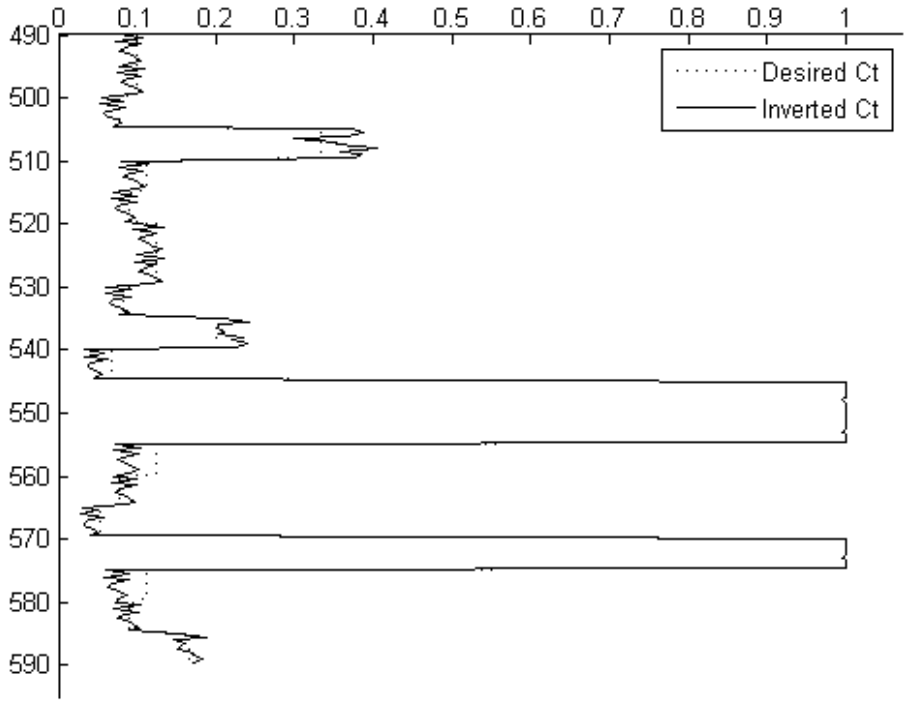


圖 4.7. 第三十一組井測資料利用 best case 30-14-10 DENN 反推的 Ct 與期望的 Ct。其反推出來 Ct 與期望的 Ct 之間的平均絕對值誤差為 0.015278。

由表 4.2、表 4.3、表 4.4、圖 4.3、圖 4.4 及圖 4.5 可以發現，雖然隨著網路輸入向量的擴增，越高階的 DENN 可以得到越好的訓練結果，但實驗的結果與先前使用梯度坡降法來訓練的結果比較起來，仍然有訓練速度較慢及誤差過大的問題。由於差分進化法先天上為一種全域的最佳化方式，當遇到網路大小為 20-8-10 時，根據 (4.3) 式，差分進化法必須搜尋 258 度空間才能找到最佳解，因此有訓練速度較慢的問題。至於誤差過大的問題，則與群組大小 N_p 與演化的停止世代 G 有關。群組大小 N_p 越大，代表能突變的個體數越多，越有機會找尋到全域的最佳解，然後隱含的則是執行的時間必須提高。而停止世代 G，則是用來避免演化進入不可預期的計算時間。當 G 設定的太低，則演化過程可能過早停止，而得到非最佳解；若 G 設定的太高，則必須增加更多時間來進行演化。因此 N_p 與 G 的設定，跟著問題不同而有所不同。

4.4 Combination of DE and Gradient Descent

因為只利用差分進化法並不能夠得到一個令人滿意的結果，所以在這個實驗中，我們希望能結合差分進化法的優點與梯度坡降法的優點來做兩階段的訓練 (DENN with gradient descent)。第一階段的訓練即是利用差分進化法進行全域搜尋，得到一組網路權重值。第二階段的訓練就是對第一階段找出的網路利用梯度坡降法進行區域搜尋，使用倒傳遞演算法得到最終的訓練結果。

我們預期第一階段的訓練能夠得到一組全域上的較佳解，避免一開始就陷入區域最小值的問題。第二階段的訓練則是能讓我們從較佳解去得到一組更精確的最佳解。

在這個實驗中，我們分別對實驗 4.3 中兩層的二階高階 DENN 與兩層的三階高階 DENN 所得到的十次執行結果利用梯度坡降法來進行第二階段的訓練，而網路的執行步驟設定為 20,000，學習速率為 0.6，動量係數為 0.4。訓練完之後，找出訓練結果中具有最小 MAE 的網路，再拿此網路分別對第二十六組至第三十一組模擬資料做測試，每一個測試的資料都可以得到一組由此網路得出來的輸出值。最後再將這些輸出值經過還原數值 (re-scaled) 的步驟，即可以得到由此網路反推出來的一組真實地層導電率 (true formation conductivity, Ct)。

表 4.5 為訓練的結果，圖 4.8 為在第一階段訓練後具有最小 MAE 的 20-8-10 DENN with gradient descent 的訓練過程。圖 4.9 為在第一階段訓練後具有最小 MAE 的 30-8-10 DENN with gradient descent 的訓練過程。比較表 3.5 與表 4.5，我們可以發現利用此種兩階段訓練的網路，可以得到較佳的訓練結果，雖然必須進行兩階段較耗時的訓練過程。

表 4.5. 兩階段的訓練結果。

Network size	Number of training patterns	AVG. MAE of 1st step	AVG. MAE of 2nd step
20-7-10	500	0.0323	0.001675
20-8-10	500	0.0306	0.001566
20-9-10	500	0.0243	0.001596
20-10-10	500	0.0265	0.001587
20-11-10	500	0.0245	0.001599
20-12-10	500	0.0257	0.001650
20-13-10	500	0.0248	0.001586
20-14-10	500	0.0253	0.001630
30-7-10	500	0.0301	0.001600
30-8-10	500	0.0279	0.001566
30-9-10	500	0.0233	0.001594
30-10-10	500	0.0245	0.001583
30-11-10	500	0.0228	0.001584
30-12-10	500	0.0250	0.001602
30-13-10	500	0.0229	0.001586
30-14-10	500	0.0221	0.001622

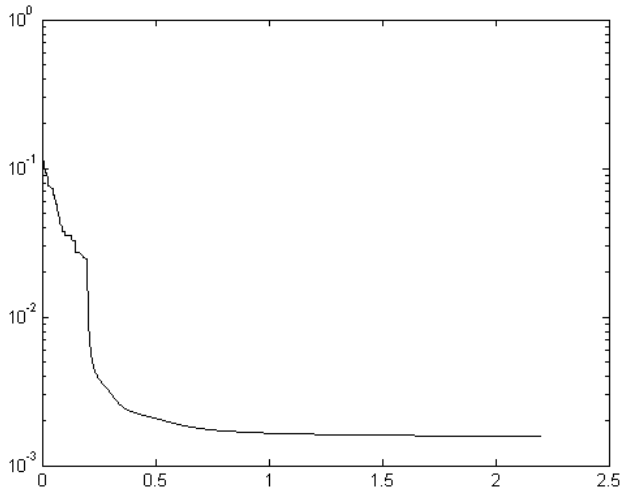


圖 4.8 在第一階段訓練後具有最小 MAE 的 20-8-10 DENN with gradient descent 的訓練過程。

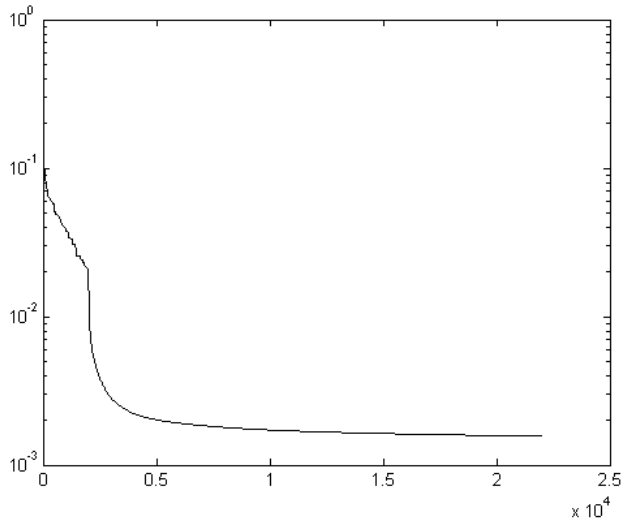


圖 4.9 在第一階段訓練後具有最小 MAE 的 30-8-10 DENN with gradient descent 的訓練過程。

圖 4.10 與圖 4.11 是利用具有最小 MAE 的 20-8-10 DENN with gradient descent 來分別對第二十六組資料與第三十一組資料做井測反推的結果。圖 4.12 與圖 4.13 是利用具有最小 MAE 的 30-8-10 DENN with gradient descent 來分別對第二十六組資料與第三十一組資料做井測反推的結果。

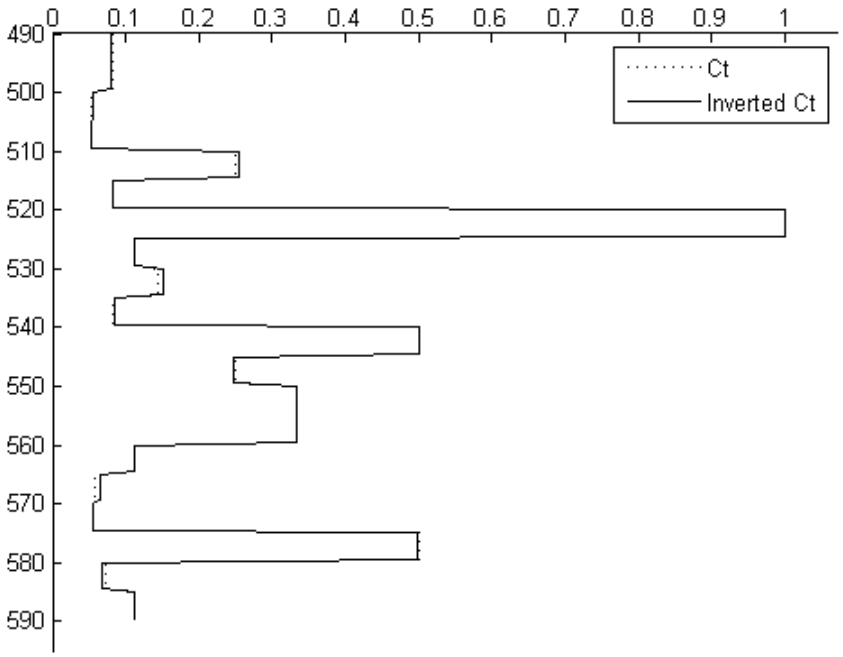


圖 4.10. 第二十六組井測資料利用有最小 MAE 的 20-8-10 DENN with gradient descent 反推的 Ct 與期望的 Ct。

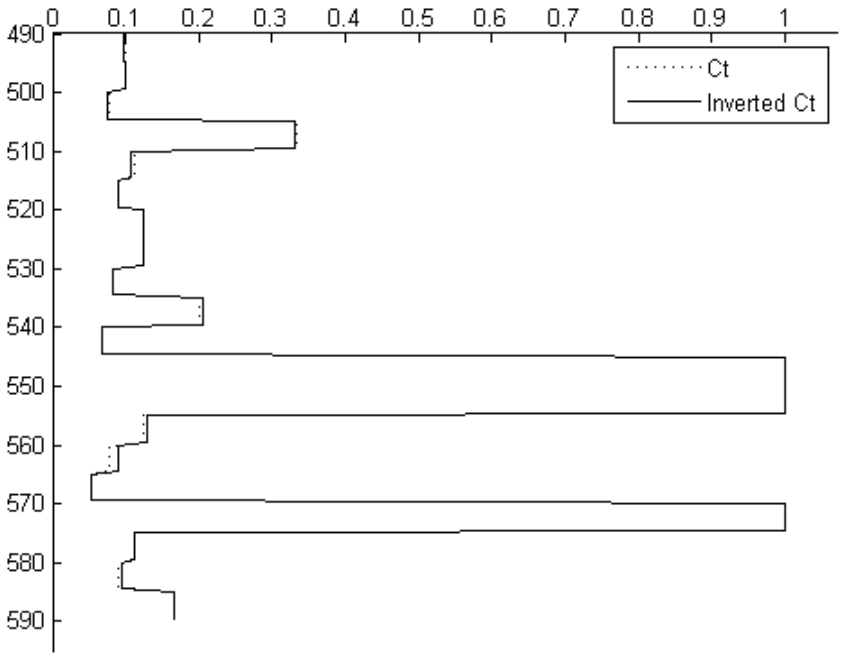


圖 4.11. 第三十一組井測資料利用有最小 MAE 的 20-8-10 DENN with gradient descent 反推的 Ct 與期望的 Ct。

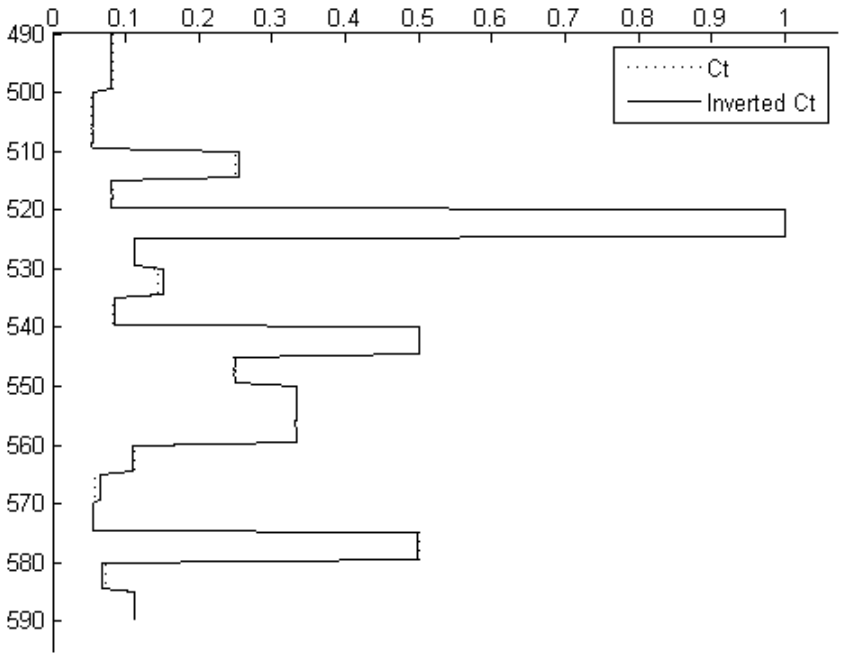


圖 4.12. 第二十六組井測資料利用有最小 MAE 的 30-8-10 DENN with gradient descent 反推的 Ct 與期望的 Ct。

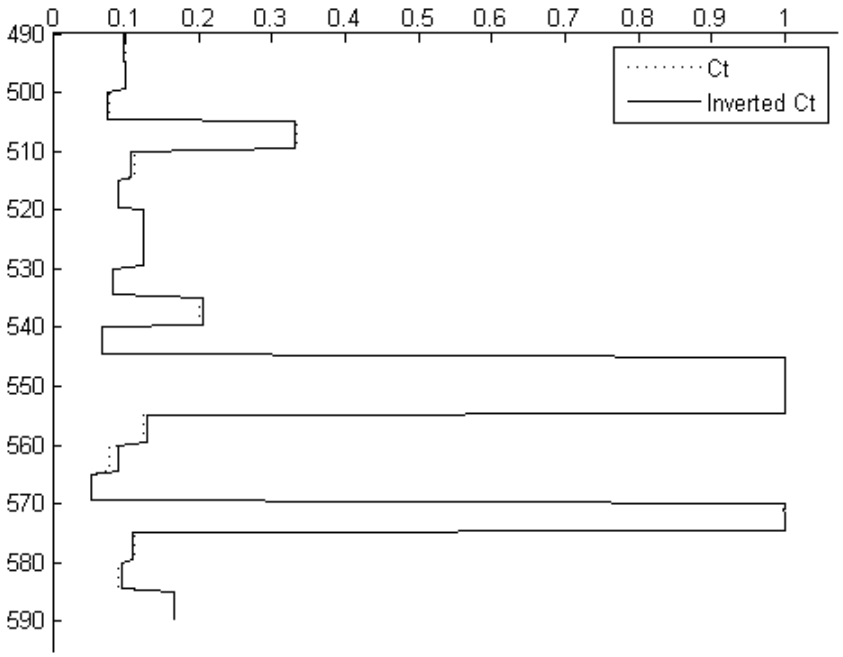


圖 4.13. 第三十一組井測資料利用有最小 MAE 的 30-8-10 DENN with gradient descent 反推的 Ct 與期望的 Ct。

5. Experimental Results on Real Field Logs

我們採用 30-8-10 網路來做真實井測資料的反推。在實驗中的真實井測資料，深度為 5,577.5 英呎到 6,722 英呎，取樣的間隔為 0.5 英呎，所以此真實井測資料總共有 2,290 個輸入樣本。

網路的輸入是 10 個輸入樣本與其輸入樣本的二階跟三階項。當實驗 4.4 利用模擬的資料訓練完 30-8-10 網路後，真實的資料便輸入到網路中，利用這個網路來反推出真實井測資料的導電率。圖 5.1 為視導電率與反推真實地層導電率的結果，圖 5.2 為反推真實地層倒電率的結果。

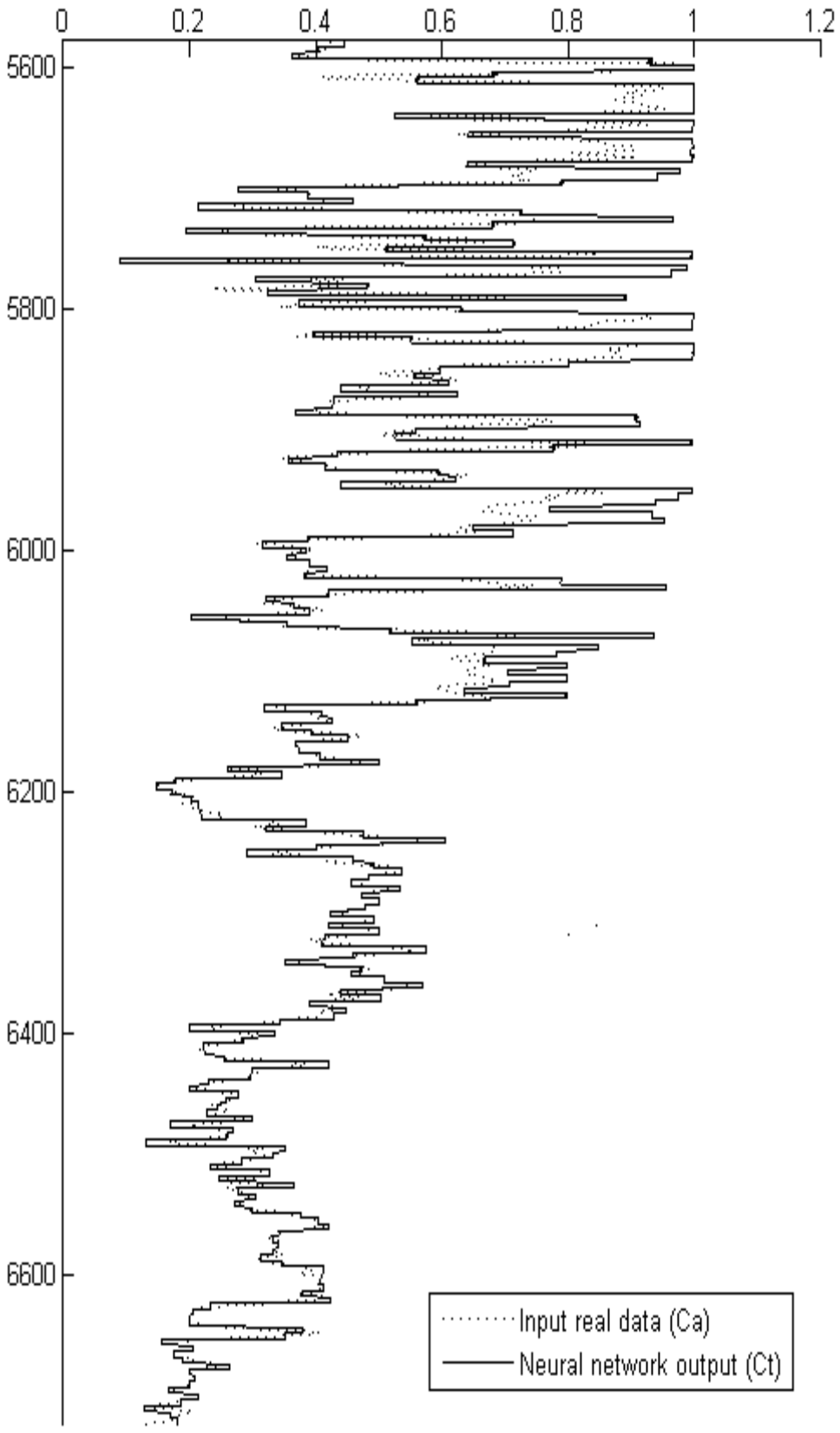


圖 5.1. 視導電率與反推真實地層導電率的結果。

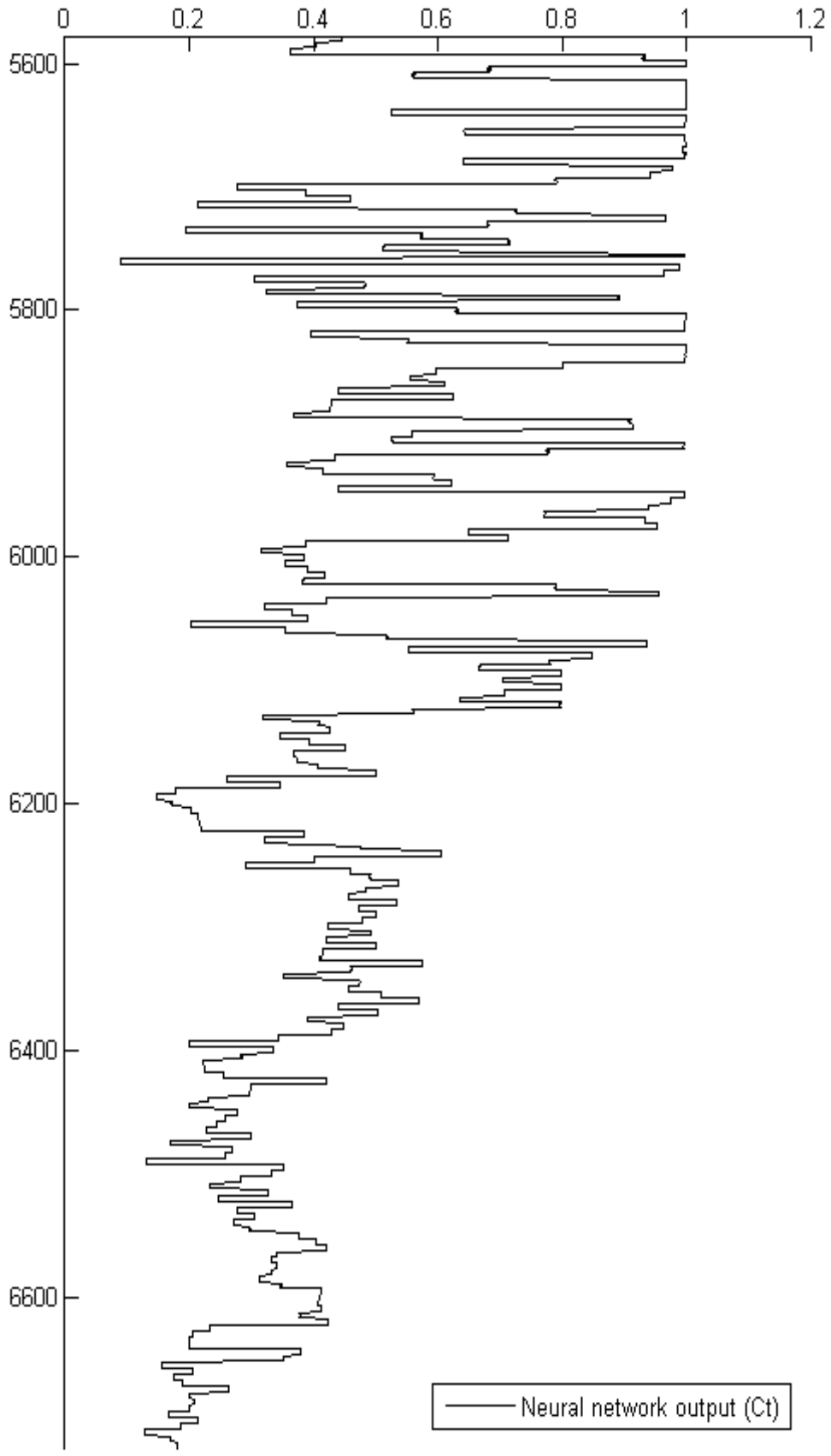


圖 5.2. 反推真實地層倒電率的結果。

6. Conclusions

本篇論文中，我們採用四種架構的感知器網路來執行井測資料的反推。第一種是使用原始特徵值當作輸入的單層感知器網路、第二種是將特徵值擴展到高階項來當作輸入的單層感知器網路、第三種是使用原始特徵值當作輸入的兩層感知器網路，而最後一種是將特徵值擴展到高階項來當作輸入的兩層感知器網路。網路訓練的方式分別採用了以梯度坡降法為基礎的倒傳遞學習法以及混合差分進化法與梯度坡降法來訓練網路。

在利用梯度坡降法來訓練網路時，我們發現使用原始特徵值當作輸入的單層感知器網路並無法有效的降低學習誤差，所以我們將特徵值擴展到二階項與三階項，再將這種高階項當作網路的輸入而採用梯度坡降法的訓練方式來訓練單層的感知器網路。我們發現上述方式可以讓網路的學習誤差顯著地降低。同時，我們也將網路擴展到兩層以期能達到更佳的學習效果。我們發現將特徵值擴展到高階項來當作輸入的兩層感知器網路，在學習的過程中可以加快收斂，同時也可以降低網路的實際輸出值與期望輸出值的平均絕對值誤差。原因是因為多層的感知器網路與擴展的高階項能夠提供更加非線性的對映。

在混合差分進化法與梯度坡降法時，我們直接訓練兩層的感知器網路與高階的兩層感知器網路。雖然這種混合式的訓練方式需要更長的訓練時間，我們發現網路實際輸出值與期望輸出值的平均絕對值誤差比只用梯度坡降法來訓練網路時更為降低。原因是我們先使用差分進化法的訓練方式進行全域搜尋來避免一開始就陷於區域最小值的問題，再利用梯度坡降法來進行區域搜尋以期能得到一個更精確的解。實驗的結果顯示，當網路利用差分進化法混合梯度坡降法來訓練時，在輸入為十個特徵值且擴展到三階項，隱藏節點為八個，以及輸出節點為十個的時候，可以得到最小的平均絕對值誤差。

而最後一個階段，我們將這個利用混合式訓練法訓練過後的 30-8-10 的網路用在真實井測資料的反推上，可有效的反推出真實井測的導電率。

References

[1] J. C. Goswami, R. Mydur, P. Wu, and D. Hwliot, “A robust technique for well-log data inversion,” IEEE Transactions on Antennas and Propagation, Volume 52, Issue 3, pp. 717-724, March 2004.

[2] Y. Y. Lin, S. Gianzero, R. Strickland, “Inversion of induction logging data using the least squares technique,” 25th Annual Logging Symposium Transactions, Paper AA1-14, 1984.

[3] C. J. Dyos, “Inversion of induction log data by the method of maximum entropy,” 28th Annual Logging Symposium, Paper T1-13, Jun 1987.

[4] L. S. Martin, D. Chen, T. Hagiwara, R. Strickland, G. Gianzero, and M. Hagan, “Neural network inversion of array induction logging data for dipping beds,” Society of Professional Well Log Analysts, 42nd Annual Logging Symposium, Paper U1-11, Jun 17-20, 2001.

[5] Y. H. Pao, Adaptive Pattern Recognition and Neural Networks, Reading, Mass., Addison-Wesley Publishing Co., 1989.

[6] Y. H. Pao and Y. Takefuji, “Functional-link net computing: theory, system architecture and functionalities,” Computer, Volume 25, Issue 5, pp. 76-79, May 1992.

[7] M. T. Hagan, H. B. Demuth, and M. Beale, Neural Network Design, PWS Publishing Co., Boston, 1996.

[8] J. H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, Mich., 1975.

[9] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, Mass., 1989.

[10] D. Montana and L. Davis, “Training feedforward neural networks using genetic algorithm,” Proceedings of the International Joint Conference on Artificial Intelligence, 1989.

[11] W. Kinnebrock, “Accelerating the standard back propagation method using a genetic approach,” Neurocomputing, Volume 6, pp. 583-588, 1994.

[12] R. Storn and K. Price, “Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces,” Global Optimization, Volume 11, pp. 341-359, 1997.

[13] V. Schnecke, O. Vornberger, “An adaptive parallel genetic algorithm for VLSI-layout optimization,” 4th International Conference on Parallel Problem Solving from Nature, Sep 22-27, 1996.

[14] Y. P. Chen and S. C. Jen, “Rank-based varying population size genetic algorithm with sizing strategy, ” Control Systems and Technology, Volume 6,

pp.243-251, 1998.

[15] B. Filipic and D. Juricic, “An interactive genetic algorithm for controller parameter optimization,” The International Conference on Artificial Neural Nets and Genetic Algorithms, pp.458-462, 1993.

[16] K. A. Michalski, Electromagnetic imaging of circular-cylindrical conductors and tunnels using a differential evolution algorithm,” Microwave and Optical Technology Letters, Volume 28, Issue 5, pp. 303-306, 2000.

[17] S. Doyle, D. Corcoran, and J. Connell, “Automated mirror design using an evolution strategy,” Optical Engineering, Volume 38, Issue 2, pp.323-333, 1999.

[18] T. Rogalsky, S. Kocabiyik, and R. Derksen, “Differential evolution in aerodynamic optimization,” Canadian Aeronautics and Space Journal, Volume 46, Issue 4, pp.183-190, 2000.

[19] G. Stumberger, D. Dolinar, U. Pahner, and K. Hameyer, “Optimization of radial active magnetic bearings using the finite element technique and differential evolution algorithm,” IEEE Transactions on Magnetics, Volume 36, Issue 4, pp.1009-1013, 2000.

[20] Kou-Yuan Huang, Neural Networks and Pattern Recognition, Weikeg Publishing Co., Taipei, Taiwan, March 2003, 406 pages.

[21] D. E. Rumelhart, G. E. Hinton, and R.J. Williams, 1986, “Learning internal representations by error propagation.” In D.E. Rumelhart and J. L. McClelland, eds., Parallel Distributed Processing: Explorations in the Microstructure of Cognition, pp. 318-362, MIT Press, Cambridge, MA.

[22] N. Qian, “On the momentum term in gradient descent learning algorithms,” Neural Networks, Volume 12, Number 1, pp. 145-151, Jan 1999.

[23] G. Mirchandani and W. Cao, “On hidden nodes for neural nets,” IEEE Transactions on Circuits and Systems, Volume 36, Number 5, pp. 661-664, May 1989.