神經網路的敏感度分析

Sensitivity Analysis and Neural Networks

蔡瑞煌* 林修葳** 林義評*

Ray Tsaih, Hsiou-Wei Lin, Yi-ping Lin

摘要

本研究探討敏感度分析之技術是否能讀取神經網路所學得之知識，以及是否能用來評估神經網路之學習績效。本研究以選擇權定價公式（Black-Scholes formula）之模擬為研究對象。本研究之實驗結果顯示敏感度分析之技術能讀取神經網路所學得之知識，也能用來評估神經網路之學習績效。

關鍵詞：敏感度分析、神經網路、選擇權

Abstract

This study presents the methodology of sensitivity analysis and explores whether it can be an alternative evaluation criterion as well as a tool to "read" artificial neural networks' knowledge. The simulation of the Black-Scholes formula is employed for this object. Since, in the Black-Scholes formula, the mapping relationship between the call price and five relevant variables is a mathematically close form, it is feasible to verify the validity of the methodology of sensitivity analysis. The experiment results are promising; they show that both values of the sensitivity analysis and the partial derivative of the Black-Scholes formula are consistent. Furthermore, the sensitivity analysis can be an alternative criterion for comparing the effectiveness of ANNs.

Keywords: Sensitivity analysis, Neural Networks, Options

1. Introduction and Literature Review

This study adopts artificial neural networks (ANN) to simulating the Black-Scholes formula to price the call options. Similar researches had been done, but they focused mainly on the performance comparison with some statistic models, and had no further analysis of the ANNs. Here we present the

* 國立政治大學資訊管理學系
Dept. of Management Information Systems, National Chengchi University
** 國立台灣大學國際企業學系
Dept. of International Business, National Taiwan University, Taiwan

methodology of sensitivity analysis, which can explore the knowledge embedded in ANNs, to see whether the ANNs are actually well trained and valid. The object of this study is to investigate whether the sensitivity analysis can be as an alternative evaluation criterion as well as a tool to “read” ANN’s knowledge. This is feasible since the ANNs are trained to simulate the Black-Scholes formula in which the mapping between the call price and five relevant variables is a mathematically close form.

There are two ANNs: the multi-layered feed-forward (MLP) networks with the Back Propagation learning algorithm (BP) (Rumelhart et al., 1986) and the RNBP learning algorithm (RNBP) (Tsaih, Chen, & Lin, 1998). The performance of BP and RNBP are measured and compared based on two criteria, the learning efficiency and the forecasting error.

When the ANN is used as a modeling tool, it is interesting to check if the ANN is well trained and if it can display some useful information about the task. For the first question, we might simply test the ANN with a huge amount of data. If the performance of the ANN is acceptable within a predetermined tolerance, it is comparatively reliable to claim that the ANN has been well trained. As for the second question, it is necessary to have a deeper analysis of the network structure, analysis that in fact is rather complicated mathematically. For example, it is desirable to be capable of specifying the relationships between input and output variables.

The sensitivity analysis, which is similar to the factor analysis in statistics, is proposed to examine the impact of each input variable. When the model had been completely understood, the sensitivity analysis can be utilized to examine whether the characteristic of each (input) variable in the network system has been “well trained.” On the other hand, when we are not sure about how they interact within a system, it is capable of exploring the complexity of its sensitive curve, which corresponds to the sensitivity of the output value to the variation of each (input) variable.

The experiment results are promising. Both values of the sensitivity analysis and the partial derivative are consistent. Furthermore, in both sensitivity analysis of ANN and partial derivative of the formula, the stock price and the strike price are the most determinant factors to the call price, compared with the other variables, the risk-free interest rate, the time to expiration, and the volatility. Also, in both sensitivity analysis and partial derivative, the stock price positively affects the call price and the strike price negatively affects the call price.

In the following, we first review the

relevant work. In section 2, we describe our experiment design. The performances and analyses of the experiments are presented in section 3. In section 4, we summarize the lessons learned from this study, as well as the future work.

1.1. The Pricing of Option

Options on stocks were first traded on an organized exchange in 1973. Since then there has been a dramatic growth on options markets. Options are now traded on many exchanges throughout the world. Huge volumes of options are also traded over the counter by banks and other institutions. The underlying assets include stocks, stock indices, foreign currencies, debt instruments, commodities, and future contracts.

There are two basic types of options: a call option gives the holder the right to buy the underlying asset by a certain date for a certain price; whereas a put option gives the holder the right to sell the underlying asset by a certain date for a certain price. The price in the contact is known as the exercise price or strike price; the date is known as the expiration date, exercise date, or maturity. American options can be exercised at any time up to the expiration date. European options can be exercised only on the expiration date.

According to (Black & Scholes, 1973), derived based on the no-arbitrage condition and other assumptions, the pricing model for the European call option can be expressed as following:

$$C = SN(d_1) - Ke^{-RT} N(d_2) \tag{1}$$

With $d_1 = \dfrac{\ln(S/K) + (R + 0.5\sigma^2)T}{\sigma\sqrt{T}}$ and

$d_2 = d_1 - \sigma\sqrt{T}$;

where C is the price of a call option, S the stock price, K the strike price, R the interest rate, T the maturity, σ the volatility of stock return, and N(x) the cumulative normal distribution density function. This formula has provided a great contribution to the option market for a long time and many advanced analyses for this model have been introduced.

The first order partial derivatives of the Black-Scholes formula are well defined; for readers who are interested in the detailed explanations of them, please refer to (Hull, 1997). Those partial derivatives will be used a benchmark for the validation of the ANN through sensitivity analysis.

1.2. Applications of artificial neural networks in pricing of options

Applications of ANN to pricing the options have been explored a lot recently, for example, (Hutchinson et al., 1994; Lajbcygier et al., 1996, and Hanke, 1997). All of them applied BP to simulate the Black-Scholes formula.

Hutchinson and his colleagues adopted the Monte Carlo simulations to produce a two-year sample of daily stock prices and create a cross-section of options each date according to the rules by the Chicago Board Options Exchange with prices given by Black-Scholes formula. For comparison, they estimated models using four popular methods: the least square method, the radial basis function neural networks, BP, and the projection pursuit. Their results showed the performance of BP was not significantly better than statistical models. Note that, in their study, the values of the strike prices (K) and maturity (T) needed to be fixed in each Monte Carlo simulation. Lajbcygier and his colleagues set up a BP with 4 inputs (the ratio of the stock price over the strike price, maturity, interest rate, and volatility) and one output (the ratio of the call price over the strike price). The ranges of the input variables were as follows: the values of S/K were in the range [0.9, 1.1], the values of T were in the range [0.0, 0.2], R was the risk free interest rate, and σ was the volatility of the underlying future. They claimed approximately 54% of the real data falls within those ranges. They compared BP's results to statistical linear regressions, and argued that, in such ranges of the input variables, BP's performance was significantly better than those of statistical methods. Hanke incorporated GARCH(1,1) model and stochastic volatility into BP networks, which had 7 input nodes, 50 hidden nodes and 1 output node. Hanke adopted the GARCH(1,1) for additional information regarding the current volatility. He merely presented the deviations from the target values.

1.3. BP and RNBP

(Rumelhart et al., 1986) presented BP; since then, BP has been widely used in many fields. There are, however, notorious predicaments when using BP; for example, the unknown of the proper number of hidden nodes, the relatively optimal learning result, and the sluggish learning process (Tsaih, 1993). Many modifications of the original BP has been presented (Sarkar, 1995); for example, the momentum strategy, the adaptive learning rate (Takechi et al., 1995), the self-adaptive back propagation (Jacobs, 1988), the controlling oscillation of weights, the rescaling of weights, the expected source values, the adaptive learning rates, the conjugate gradient method (Battiti, 1992), and the different error function. But none of them provide a generalized solution for the undesirable predicaments of BP. (Wang, 1995) argued that the unpredictability was the biggest problem of BP, and more information or prior knowledge of the case can provide more meaningful classification boundaries for the network structures.

To address these notorious predicaments

of BP, Tsaih has developed Reasoning Neural Networks, which is an MLP network with the reasoning learning algorithm (RN) (Tsaih, 1993; Tsaih, 1997; Tsaih, 1998). In summary, RN guarantees an optimal solution for 2-classes categorization learning problems. At this point, however, RN is designed to deal only with binary output patterns. When working with non-binary outputs, real numbers can first be converted into binary digits. However, this increases the number of output nodes and the learning complexity, requiring a longer learning time. Thus, Tsaih has further developed RNBP (Tsaih, Chen & Lin, 1998), which can deal with the non-binary output patterns. As stated in (Tsaih, Chen & Lin, 1998), RNBP significantly outperforms BP in the effectiveness regarding the testing data set, while both of them perform similarly in the effectiveness regarding the training data set. Therefore, we also adopt RNBP in our research.

A brief summary about RNBP is presented in the following two paragraphs.[1]

The main idea of RNBP is to utilize the following credits of the learning algorithm of RN: the ability of autonomously recruiting as well as pruning hidden nodes during the learning process, and the guaranteeing of obtaining the desired solution for the 2-classes categorization learning problem. With those credits as well as the fact that both RN and BP can be applied to the MLP network, RN may acquire some useful information for BP, for example, a proper amount of used hidden nodes and the well-assigned (initial) weights.

With respect to the application problems of the output values being real values, we firstly classify the training data into two categories via a rule of thumb. For example,

$$\overline{\mathrm{d_c}} \equiv \begin{cases} 1 & \text{if } \mathrm{d_c} \geq \mu \\ -1 & \text{if } \mathrm{d_c} < \mu \end{cases} \tag{2}$$

where d_c is the cth (real) desired output value, μ is the mean value (or the median) of d_cs, and $\overline{d_c}$ is the corresponding desired output value for RN's learning algorithm. In other words, each output value will be replaced by the associated binary digit. Such data are used as the training patterns for RN's learning algorithm. Then we adopt the network obtained from RN's learning, and uses BP to learn the original training data.

1.4. the sensitivity analysis

(Yoon et al., 1994) argued that, after building the ANN, reading or understanding the knowledge in ANN was difficult because the knowledge was distributed over the entire network. However, the sensitivity analysis of the ANN is necessary not only for a better

[1] For the details of the RNBP, the readers can be referred to [Tsaih, Chen & Lin, 1998].

understanding of the mapping between input and output variables in the applying domains, but also for the further research of the ANN itself.

Table 1 List of papers relevant to the sensitivity analysis

reference	formula
Yoon, Guimaraes, & Swale (1994)	$RS_{lj} = \frac{\sum w_{ij} r_{li}}{\sum ABS\{\sum w_{ij} r_{li}\}}$
Naimimohasses, Barnett, Green, & Smith (1995)	$S_{jl}(\mathbf{B}_c) = r_{li}\sum_{i=1}^{p} w_{ij} h(\mathbf{B}_c, \mathbf{X}_i)(1 - h(\mathbf{B}_c, \mathbf{X}_i))$ & $S = \frac{1}{n}\sum_{c} ABS\{S_{jl}(\mathbf{B}_c)\}$
Steiger & Sharda (1996)	$RS_{lj} = \frac{ABS(\overline{\Delta g_{lj}})}{\sum ABS(\overline{\Delta g_{lj}})}$
Chiou, Liu, & Tsaih (1996)	$S_{lj} \equiv \sum_{i=1}^{p}\sum_{c} r_{li}(1 - h^2(\mathbf{B}_c, \mathbf{X}_i))w_{ij}$

From the literature review, there were some related studies. Table 1 shows some relevant literature review; the explanations of symbols please refer to Table 2 and the following paragraphs. Without presenting the derivation of the methodology, (Yoon et al., 1994) proposed a way of profiling the impact of each input variable. For a network with one hidden layer, it involved computing a test statistic of the form:

$$RS_{lj} = \frac{\sum w_{ij} r_{li}}{\sum ABS\{\sum w_{ij} r_{li}\}} \qquad (3)$$

RS_{lj} was the relative strength between the jth input and the lth output variable, and ABS meant the absolute value. This statistic measured the strength of the relationship of the jth input and the lth output variable to the total strength of all of the input and output variables. It was similar to the multivariate analysis.

(Naimimohasses et al., 1995) defined a sensitivity matrix for inputs and outputs vector arrays over the training patterns c:

$$S_{jl}(\mathbf{B}_c) = r_{li}\sum_{i=1}^{p} w_{ij} h(\mathbf{B}_c, \mathbf{X}_i)(1 - h(\mathbf{B}_c, \mathbf{X}_i)) \qquad (4)$$

$$S = \frac{1}{n}\sum_{c} ABS\{S_{jl}(\mathbf{B}_c)\} \qquad (5)$$

That is, the total sensitivity S was derived by calculating the statistical significance of the contribution due to each individual input, $S_{jl}(B_c)$, over all training patterns. Naimimohasses et al. were interested in plotting the sensitivity function of training epochs, which could give trends of relative input sensitivity.

Table 2 List of Symbols

m, p, and q : the amounts of input, hidden, and output nodes, respectively.
$\mathbf{B}_c$: the cth given stimulus input pattern, c=1, 2 , …, k.
b_{cj} : the input value received by the jth input node when $\mathbf{B}_c$ is presented to the network.
w_{ij} : the weight of connection between the jth input and the ith hidden node.
$\mathbf{w}_i$: the vector of weights of the connections between all input nodes and ith hidden node; $\mathbf{w}_i \equiv (w_{i1}, w_{i2}, \ldots w_{im})$.
θ_i : the negative of the threshold value of the ith hidden node.
$\mathbf{X}_i^t \equiv (\theta_i, \mathbf{w}_i^t)$ and $\mathbf{X}^t \equiv (\mathbf{X}_1^t, \mathbf{X}_2^t, \ldots, \mathbf{X}_p^t)$.
$h(\mathbf{B}_c, \mathbf{X}_i)$: the activation value of the ith hidden node given the stimulus $\mathbf{B}_c$, and

$$h(\mathbf{B}_c, \mathbf{X}_i) \equiv \tanh\left(\theta_i + \sum_{j=1}^{m} w_{ij} b_{cj}\right)$$

$\mathbf{h}(\mathbf{B}_c, \mathbf{X})$: the activation value vector of all hidden nodes given the stimulus $\mathbf{B}_c$, and
$\mathbf{h}(\mathbf{B}_c, \mathbf{X}) \equiv (h(\mathbf{B}_c, \mathbf{X}_1), h(\mathbf{B}_c, \mathbf{X}_2), \ldots, h(\mathbf{B}_c, \mathbf{X}_p))^t$
r_{li} : the weight of the connections between ith hidden nodes and lth output node.
$\mathbf{r}_l$: the vector of weights of the connections between all hidden nodes and lth output node; $\mathbf{r}_l \equiv (r_{l1}, r_{l2}, \ldots, r_{lp})$.
s_l : the negative of the threshold value of the lth output node.
$\mathbf{Y}_l^t \equiv (s_l, \mathbf{r}_l^t)$, $\mathbf{Y}^t \equiv (\mathbf{Y}_1^t, \mathbf{Y}_1^t, \ldots, \mathbf{Y}_q^t)$, and $\mathbf{Z}^t \equiv (\mathbf{Y}^t, \mathbf{X}^t)$.
$net(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X})$: the net input value of the lth output node given the stimulus $\mathbf{B}_c$, and

$$net(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X}) \equiv \tanh\left(s_l + \sum_{i=1}^{p} r_{li} h(B_c, X_i)\right)$$

$O(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X})$: the activation value of the lth output node given the stimulus B_c, and

$$O(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X}) \equiv \tanh(net(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X}))$$

d_{cl} : the desired output value of the lth output node when B_c is presented to the network.

(Jarvis & Stuart, 1996) adopted the sensitivity analysis to explore the effects of altering networks parameters on the training times and the classification accuracy. (Steiger & Sharda, 1996) calculated the relative sensitivity of inputs to the network by “wiggling” each input value. The effect of each wiggle on every training pattern was determined, and the overall average absolute difference between the modified outputs and original outputs was calculated. The input sensitivity was the relative ratio of the overall average absolute difference over all input variables. It seems their mathematical model was as following:

$$RS_{lj} = \frac{ABS\left(\overline{\Delta g_{lj}}\right)}{\sum ABS\left(\overline{\Delta g_{lj}}\right)} \qquad (6)$$

where RS_{lj} is the relative sensitivity between the jth input and the lth output variable, $\overline{\Delta g_{lj}}$ is the average value of the effect by wiggling the value of each input.

In (Chiou et al., 1996), the sensitivity factor of each input variable was defined as the sum of partial derivatives of all training patterns as:

$$S_{lj} \equiv \sum_{i=1}^{p} \sum_{c} r_{li} \left(1 - h^2(\mathbf{B}_c, \mathbf{X}_i)\right) w_{ij} \qquad (7)$$

S_{lj} was the sensitivity factor between the jth input and the lth output variable.

Some sensitivity analyses described in Table 1 provided a gross indicator of key factor via measuring the effect of altering an input variable on the output value by integrating over all input patterns. However, they ignored the potential interaction between two or more input patterns. To summate them probably neutralizes their interactions.

Here we modify the sensitivity factor derived in (Chiou et al., 1996). The modified sensitivity factor does not summate over all training patterns, and is defined as following:

$$S_{lj}^{i}(\mathbf{B}_c) \equiv \left(\frac{\partial O(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X})}{\partial b_j}\right)^{i}$$

$$= \frac{\partial O(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X})}{\partial net_l} \frac{\partial net_l}{\partial h_i} \frac{dh_i}{dnet_i} \frac{\partial net_i}{\partial b_j} \qquad (8)$$

$$S_{lj}(\mathbf{B}_c) \equiv \sum_{i=1}^{p} S_{lj}^{i}(\mathbf{B}_c)$$

$$= \left(1 - O^2(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X})\right) \sum_{i=1}^{p} r_{li} \left(1 - h_i^2(\mathbf{B}_c, \mathbf{X}_i)\right) w_{ij} \qquad (9)$$

where net_l corresponds to the net input variable of the lth output node, h_i corresponds to the activation variable of the ith hidden node, and net_i corresponds to the net input variable of the ith hidden node. The relative sensitivity factor is defined as follows:

$$R_{lj}(\mathbf{B}_c) \equiv \frac{S_{lj}(\mathbf{B}_c)}{\sum_{k=1}^{q} ABS\left(S_{lk}(\mathbf{B}_c)\right)}$$

$$= \frac{\sum_{i=1}^{p} r_{li} \left(1 - h_i^2(\mathbf{B}_c, \mathbf{X}_i)\right) w_{ij}}{\sum_{k=1}^{q} ABS\left(\sum_{i=1}^{p} r_{li} \left(1 - h_i^2(\mathbf{B}_c, \mathbf{X}_i)\right) w_{ik}\right)} \qquad (10)$$

It is feasible to plot together the $R_{lj}(B_c)$ values and the partial derivatives of a close form equation of all training patterns, and make the comparison. For example, suppose an equation is defined as:

$$f(w, x, y, z) = \frac{1}{3} w + x^2 + 4y^3 - 10z^4 \qquad (11)$$

where w, x, y and z are independent variables. We use the RNBP network with four input nodes and one output node, and there are totally 400 training patterns, which are generated randomly with the value of each variable being ranged from -0.5 to 0.5.

The mean values of w, x, y and z over those 400 training patterns are 0.034222, 0.005553, -0.00547, and -0.02562, respectively. After finishing the learning, the $R_{lj}(B_c)$ values of all training patterns can be calculated from the methodology described above, and then compared to the partial derivative values of f(w, x, y, z). Let's take the y as the illustration. Figure 1 shows the values of y, the corresponding values of $\frac{\partial f}{\partial y}$, and the corresponding $R_{lj}(B_c)$ values derived from RNBP. It seems that RNBP's generalizing ability is not good when the value of y is less than -0.4 or greater than 0.4.

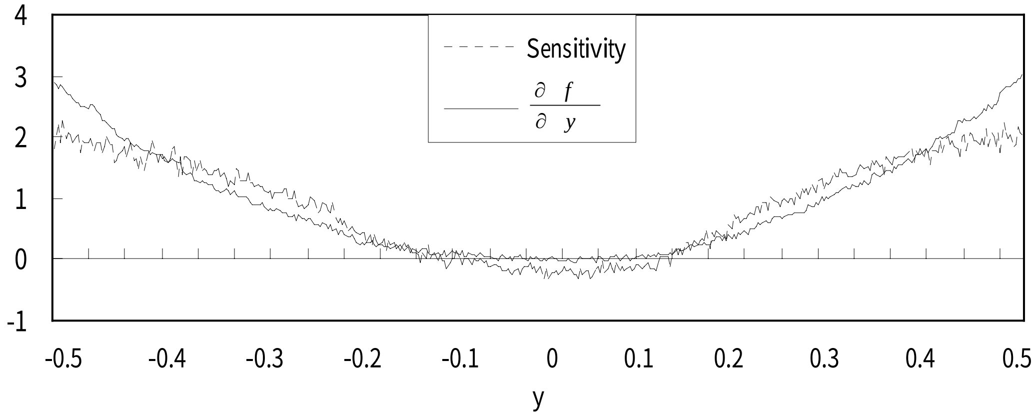


Figure 1 The curves of partial derivate and sensitivity of equation (11) concerning with y

Let $S_{lj}(b_j) \equiv$

$$\sum_{i=1}^{p} r_{li} w_{ij} \int\!\!\int_{b_k, k \neq j}\!\!..\int \left(1 - O^2(\mathbf{B}, \mathbf{Y}_l, \mathbf{X})\right)\left(1 - h^2(\mathbf{B}, \mathbf{X}_i)\right)$$

Since the value of

$$\int\!\!\int_{b_k, k \neq j}\!\!..\int \left(1 - O^2(\mathbf{B}, \mathbf{Y}_l, \mathbf{X})\right)\left(1 - h^2(\mathbf{B}, \mathbf{X}_i)\right) \text{ is}$$

independent with b_j,

$$R_{lj} \equiv \frac{S_{lj}(b_j)}{\sum_{k=1}^{q} ABS(S_{lk}(b_k))} = \frac{\sum_{i=1}^{p} r_{li} w_{ij}}{\sum_{k=1}^{q} ABS\left(\sum_{i=1}^{p} r_{li} w_{ik}\right)}. \tag{12}$$

Take the above case of f(w, x, y, z) as the illustration, the R_{lj} of w, x, y, z are 0.424315 -0.106258, 0.411498, and -0.057929, respectively; and the mean values of the $R_{lj}(B_c)$ are 0.213891, 0.002035, 0.351272, and 0.049241, respectively. Compared with the mean values of partial derivatives of f concerning w, x, y, and z, which are 0.33333, 0.011106, 0.596596, and 0.047527, the relative magnitudes of R_{lj} or $R_{lj}(B_c)$ are similar to the mean values of partial derivatives of f, though the numbers are not the same.

A larger $R_{lj}(B_c)$ indicates that, given a particular input pattern B_c, the lth output is more sensitive to the deviation of the jth input, while a larger R_{lj} indicates that, in general, the deviation of the jth input has a larger impact to the lth output. Thus, $R_{lj}(B_c)$ and R_{lj} are the relative impacts of the jth input to the lth output concerning some input pattern B_c and a general input pattern, respectively.

With the definition of the relative impact, $R_{lj}(B_c)$ and R_{lj}, it is capable of calculating the relative average sensitivity of the output to each input despite the (input) variables' interdependency. If the simulation model is unknown, the relative impact can be used to explore the characteristic of each input. In addition, the relative impact can be a tool for factors filtering. If the input variables of an ANN are imperfectly selected or have incomplete information, the relative impact is helpful for finding a less relevant factor whose relative impact is zero or tiny.

2. Experiment Designs and Methodology

The input data (S, K, R, T, σ) are generated randomly in the range defined in Table 3, and the desired output is the call prices derived based on the Black-Scholes formula. The patterns are separated into two categories: the in-the-money options and the out-of-the-money options. It is called the in-the-money option if the stock price is greater than its strike price; and the out-of-the money option if the stock price less than its strike price. In order to study whether the ANN behaves differently upon those two categories of options, there are two experiments: one for the in-the-money options and one for the out-of-the-money options.

Table 3 Ranges of input variables of training networks

Variable	In-the-money	Out-of-the-money	Distribution
S	20 - 60	40 - 60	Uniform
K	1.0 < S/K < 1.3	0.7 < S/K < 1.0	Uniform
R	0.04 - 0.09	0.04 - 0.09	Uniform
T	0.01 - 1.0	0.2 - 1.0	Uniform
σ	0.2 - 0.6	0.3 - 0.7	Uniform

Thus, there are five input nodes, each of which corresponds to each input variable, respectively, and one output node, which corresponds to the call prices. There exist 400 training patterns and 1000 testing patterns for both BP and RNBP. The networks are trained with a random sequence of patterns; Figures 2 and 3 display the sorted and unsorted desired values of call prices of the 400 training concerning the in-the-money options and the out-of-the-money options, respectively. The average call prices of the in-the-money and the out-of-the-money options are 7.96 and 6.17, respectively.

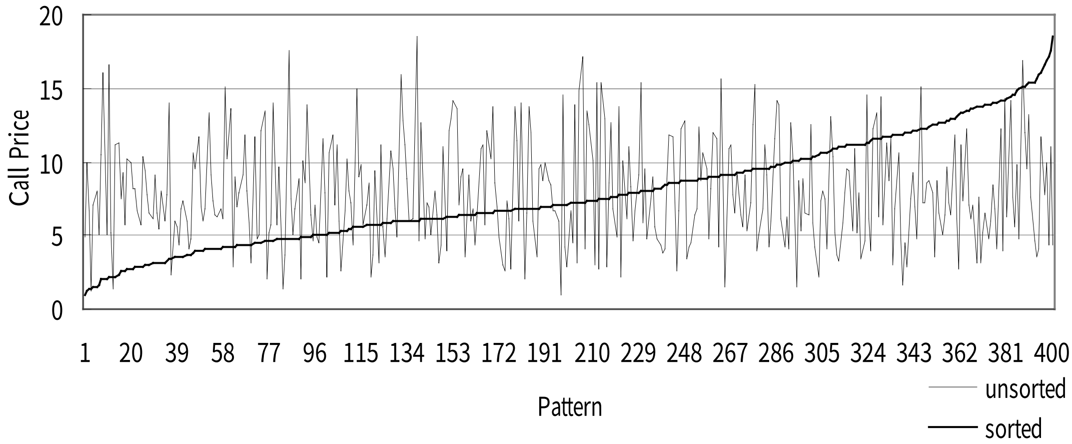


Figure 2 The desired call prices concerning the in-the-money options

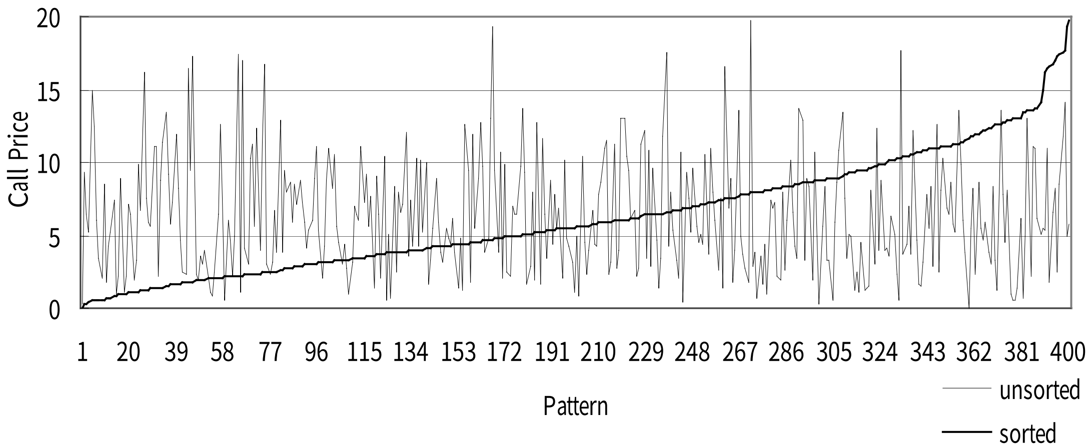


Figure 3 The desired call prices concerning the out-of-the-money options

RNBP is applied repeatedly to the same data set with different learning parameters and input sequence of the training patterns. As RNBP repeats, different amounts of recruited hidden nodes are obtained. Thus RNBP are applied several times till the variance of the amounts of recruited hidden nodes is acceptable. From the experimental results, twenty repeated simulations of RNBP has a reasonable volatility of the amounts of recruited hidden nodes. Therefore, for both experiments upon the in-the-money options and the out-of-money options, 20 repeated simulations of RNBP and BP have been performed.

There are two BPs, which have 4 and 12 hidden nodes, respectively, and are denoted BP(1) and BP(2), respectively. The initial weights and threshold value are given randomly from -1.0 to 1.0. The reason for we run BP(2) is that the average amounts of hidden nodes recruited over 20 RNBPs are near 12. It is desirable to make a fair comparison between BP and RNBP based on a similar network structure.

The evaluation criteria for the system performance include the efficiency and the effectiveness. The purpose of exploring the efficiency is to study which one has a faster learning process, and the exploring the effectiveness is to study which one has a better generalization ability.

For displaying the efficiency, the information of the average amount of learning iterations spent in each case is used. One of the stopping criteria of the learning is the tolerable error level, which is set as 0.01; however, the upper bound of learning iterations is 10000. That is, if the value of the total error can not converge below the tolerable error level within 10000, the learning stops. As for measuring the effectiveness, the mean relative error is used. The mean relative error is defined as:

$$\frac{\sum_c \sum_l \left| \mathrm{d}_{cl} - \mathrm{O}\left(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X}\right) \right| / a_{cl}}{n} \tag{13}$$

where a_{cl} is the actual option prices, the subscript c denotes the cth testing data, and n is the amount of testing data.

3. Performance and Analysis

3.1. Simulation performance of BP and RNBP

Table 4 displays the summary results of simulations. The averages and standard deviations of N_{RNBP} concerning the in-the-money options are very similar to those concerning the out-of-the-money options. Table 4 also displays following facts:

(1) In each experiment, the average and the standard deviation of $T_{BP(1)}$ and $A_{BP(1)}$ are quite similar to those of $T_{BP(2)}$ and $A_{BP(2)}$.

(2) From the experimental results of $T_{BP(1)}$, $T_{BP(2)}$ and T_{RNBP}, it seems that, in terms of the learning efficiency, it is more difficult for both ANNs to learn the out-of-the-money options than the in-the-money options. Furthermore, Both BP and RNBP perform better in the in-the-money options than in the out-of-the-money options.

Table 4 Simulation results of BP and RNBP. T denotes the amount of the learning iterations taken in the BP part, A denotes the mean relative error, and N is the amount of recruited hidden nodes.

The in-the-money options							
	$T_{BP(1)}$	$A_{BP(1)}$	$T_{BP(2)}$	$A_{BP(2)}$	T_{RNBP}	A_{RNBP}	N_{RNBP}
Average	6222	8.24%	6596	8.29%	3684.10	8.85%	12.35
Standard deviation	150.67	0.02%	284.57	0.019%	3929.73	0.80%	5.45
Average CPU time (seconds)	120.7		280.33		156.57		
The out-of-the-money options							
	$T_{BP(1)}$	$A_{BP(1)}$	$T_{BP(2)}$	$A_{BP(2)}$	T_{RNBP}	A_{RNBP}	N_{RNBP}
Average	10000	18.32%	10000	18.31%	5704.3	12.27%	12.1
Standard deviation	0	0.04%	0	0.01%	3342.06	4.06%	5.45
Average CPU time (seconds)	193.98		425		242.42		

The in-the-money options					
	$(T_{BP(1)}, A_{BP(1)})$	$(T_{BP(2)}, A_{BP(2)})$	(T_{RNBP}, A_{RNBP})	(T_{RNBP}, N_{RNBP})	(A_{RNBP}, N_{RNBP})
correlation coefficient	0.7937	0.8823	0.5247	-0.0554	-0.1043
The out-of-the-money options					
	$(T_{BP(1)}, A_{BP(1)})$	$(T_{BP(2)}, A_{BP(2)})$	(T_{RNBP}, A_{RNBP})	(T_{RNBP}, N_{RNBP})	(A_{RNBP}, N_{RNBP})
correlation coefficient	0	0	0.5504	-0.0803	0.1324

(3) Regarding the mean relative errors, RNBP outperforms both BP(1) and BP(2) (p-values are 1.12E-06 and 1.15E-06, respectively, by T-test) in the out-of-the-money options, although both BP(1) and BP(2) perform a little better than RNBP (p-values are 0.0108 and 0.0268, respectively, by T-test) in the in-the-money options.

(4) The standard deviation of A_{RNBP} is evidently greater than that of $A_{BP(1)}$ and $A_{BP(2)}$ (p-values are nearly 0.0, by F-test) in both in-the-money and out-of-the-money options.

(5) The mean of T_{RNBP} is significantly less, but the CPU time taken by RNBP is larger than the one taken by BP(1). This is because RNBP adopts much more hidden nodes (12.35 and 12.1 in average, respectively) than BP(1). More hidden nodes cause more time complexity.

(6) A_{RNBP} increases as T_{RNBP} increases. This is reasonable since some simulations do not reach the tolerant error level within 10000 times. If they do not learn successfully, their

forecasting errors should be larger.

(7) N_{RNBP} is less relevant to either A_{RNBP} or T_{RNBP} in both in-the-money and out-of-the-money experiments.

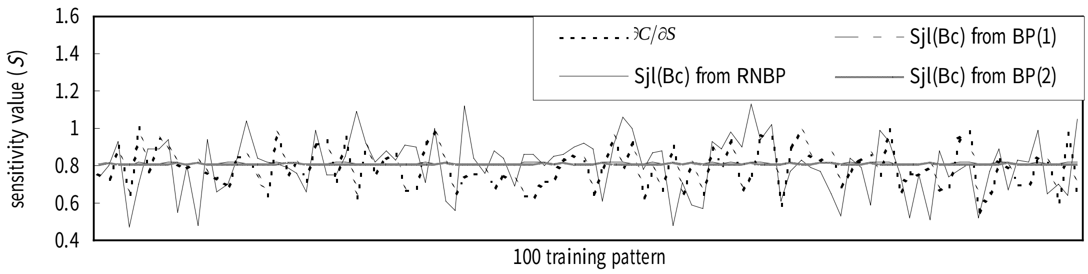
3.2. the sensitivity analysis

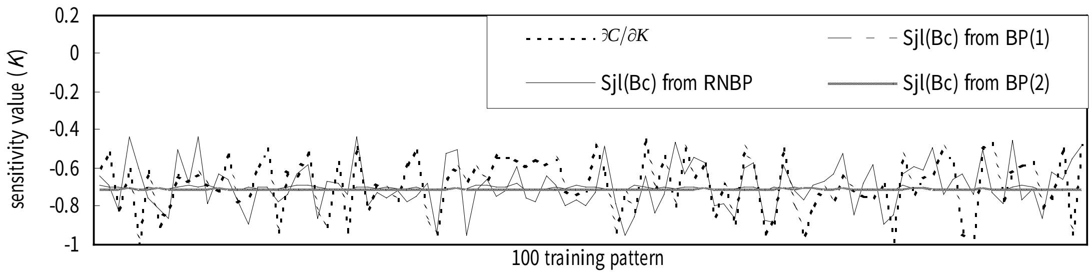
Table 5 The results of sensitivity analysis

			S	*K*	*R*	*T*	σ
In-the-money	B-S	mean (standard deviation) of the partial derivative values over all training patterns	0.4549 (0.0311)	-0.3915 (0.0546)	0.0803 (0.0339)	0.0283 (0.0278)	0.0393 (0.0372)
	BP(1)	mean (standard deviation) of the $R_{lj}(\mathbf{B}_c)$ values over all training patterns	0.4595 (1.05E-05)	-0.3917 (1.56E-05)	0.0755 (1.16E-05)	0.0297 (1.30E-05)	0.0436 (9.23E-06)
		R_{lj}	0.4595	-0.3917	0.0756	0.0297	0.0436
	BP(2)	mean (standard deviation) of the $R_{lj}(\mathbf{B}_c)$ values over all training patterns	0.4516 (1.81E-05)	-0.3868 (3.10E-05)	0.0946 (1.62E-05)	0.0292 (2.45E-05)	0.0423 (1.60E-05)
		R_{lj}	0.4518	-0.3816	0.0908	0.0293	0.0426
	RNBP	mean (standard deviation) of the $R_{lj}(\mathbf{B}_c)$ values over all training patterns	0.4521 (0.1186)	-0.3752 (0.1321)	0.0029 (0.1075)	0.0347 (0.0564)	0.0523 (0.0425)
		R_{lj}	0.4427	-0.3961	0.0469	0.0725	0.0418
Out-of-the-money	B-S	mean (standard deviation) of the partial derivative values over all training patterns	0.4289 (0.1368)	-0.2792 (0.0982)	0.1196 (0.0646)	0.0882 (0.0346)	0.1685 (0.0559)
	BP(1)	mean (standard deviation) of the $R_{lj}(\mathbf{B}_c)$ values over all training patterns	0.4201 (0.0029)	-0.2690 (0.0018)	0.0531 (0.0065)	0.0886 (0.0006)	0.1692 (0.0012)
		R_{lj}	0.4201	-0.2689	0.0532	0.0886	0.1692
	BP(2)	mean (standard deviation) of the $R_{lj}(\mathbf{B}_c)$ values over all training patterns	0.4145 (3.08E-05)	-0.2654 (6.22E-05)	0.0657 (2.64E-05)	0.0874 (3.75E-05)	0.1670 (3.14E-05)
		R_{lj}	0.4145	-0.2653	0.0657	0.0874	0.1671
	RNBP	mean (standard deviation) of the $R_{lj}(\mathbf{B}_c)$ values over all training patterns	0.4266 (0.0546)	-0.2862 (0.0466)	0.0299 (0.0603)	0.0961 (0.0211)	0.1886 (0.0371)
		R_{lj}	0.4123	-0.2719	0.0735	0.0531	0.1535

Table 5 displays the results of the sensitivity analysis. The results in both ANN and partial derivative values are consistent. It shows that, in both ANN and partial derivative values, S and K are the most determinant factors to the call price, compared with the other variables, R, T and σ. Furthermore, in both ANN and partial derivative values, S positively affects the call price and K negatively affects the call price. Thus, the results of the sensitivity analysis indicate that both ANNs have learned the characteristics of those five input variables.

However, it seems that RNBP has learned the characteristics of those five input variables more successfully than BP. Table 5 also displays that the standard deviations of $R_{lj}(B)$ in BP are much smaller, compared with those in RNBP. Such phenomenon can be demonstrated more clearly with Figure 4. It seems that, with BP, the standard deviation of the sensitivity of every input pattern is almost the same. This may be due to the saturation phenomenon happened in BP. BP adopts the tanh function as its activation function. Thus, if most weights between the hidden nodes and the output node are large, then, for any (input) patterns, the magnitude of the net input to the output node is liable to be too large such that its output value will saturate (near to 1.0 or –1.0). This is denoted as the saturation phenomenon. RNBP evidently does not have the saturation phenomenon. Furthermore, the distribution of its sensitivity values is more reasonable, comparing to BP.





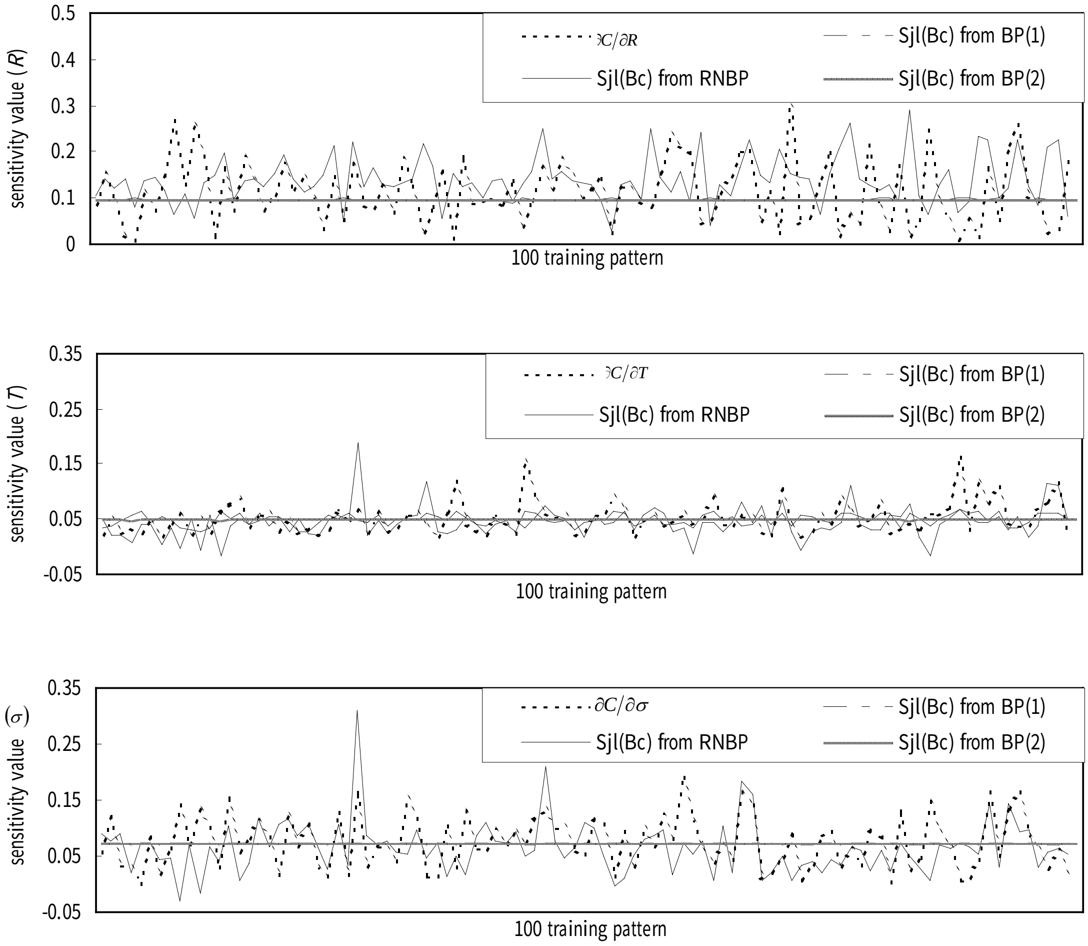


Figure 4　The sensitivity curves

Figures 5 and 6 display the frequency distributions of the desired call prices and the forecast results of BP and RNBP regarding those 400 training patterns. Those figures agree with the previous conclusion that BP performs better than RNBP in the in-the-money options, while worse in the out-of-the-money options.

The most interesting observation of Figure 5 is that, there is no occurrence of forecasting value on the range of call price either less than 2.0 or greater than 17. Likewise, in Figure 6, no occurrence of forecasting value on the range of call price greater than 13.8. It seems that both ANNs are not well trained since their generalization ability are defective in some range of call prices.

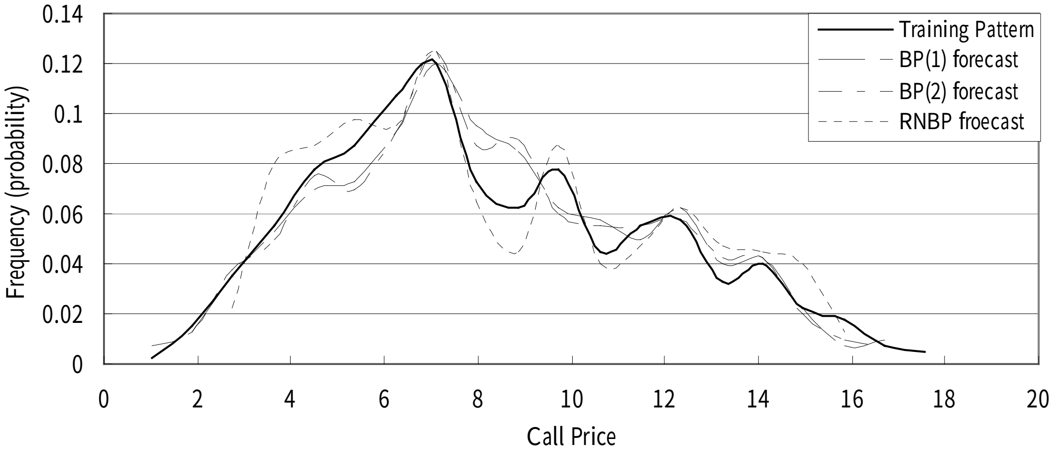


Figure 5 The frequency distribution in the in-the-money experiment

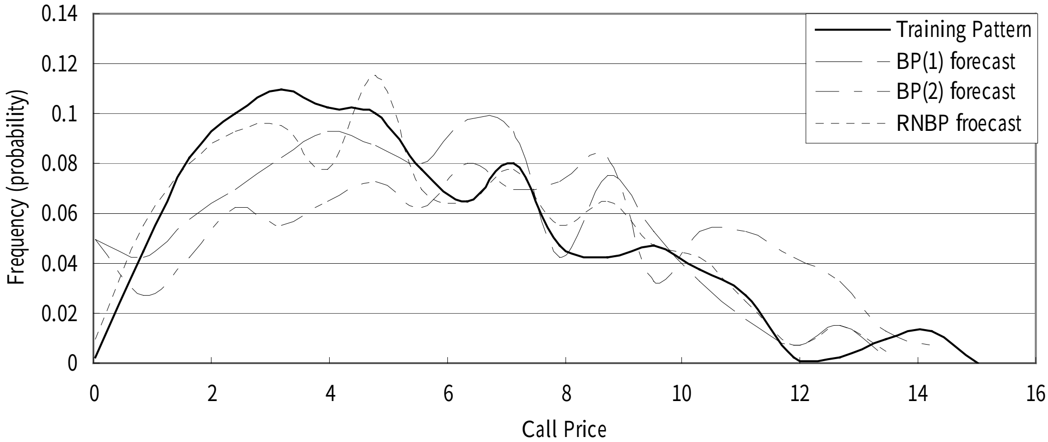


Figure 6 The frequency distribution in the out-of-the-money experiment

4. Summary and Future Work

The following lessons have been learned from this study:

(1) Table 5 displays the fact that the standard deviations of $R_{lj}(B)$ in BP are much smaller, compared with those in RNBP. This may be due to the saturation phenomenon happened in BP. On the other hand, RNBP evidently does not have the saturation phenomenon. Furthermore, the distribution of RNBP's sensitivity values is more reasonable, comparing to BP.

(2) The results of the sensitivity analysis of RNBP in this study are consistent with our

prior expectations. The sensitivity analysis can indicate the key factors which contribute the most impacts to the outputs.

In summary, the sensitivity analysis can be an alternative criterion for comparing the effectiveness of ANNs. Moreover, the sensitivity analysis can discover the knowledge embedded in ANN. Thus it is an efficient tool for information filtering and mining in an unknown environment.

Although we have obtained some robust results, the following topics need to be further studied in the future:

(1) The sensitivity analysis can discover the knowledge embedded in ANN. This is useful for artificial intelligent agent in applications, especially in this overmuch information society. One future work is to explore further the ability of the sensitivity analysis in reading the knowledge embedded in ANN via applying it to a real practical problem.

(2) We have observed that there is no occurrence of forecasting value on the range of call price either less than 2.0 or greater than 17 in Figure 5; similarly in Figure 6, there is no occurrence of forecasting value on the range of call price greater than 13.8. It seems that both ANNs are not well trained since its generalization ability is defect in those ranges of call prices. One future work is to explore the reason behind this defect.

Reference

1. Battiti, R. “First- and second-order methods for learning between steepest descent and Newton’s method,” Neural Networks, Vol. 5, 1992, pp. 507-529.
2. Black, F. and Scholes, M. “The pricing of options and corporate liabilities,” Journal of Political Economy, Vol. 81, 1973, pp. 637-659.
3. Chiou, Y., Liu, S. and Tsaih, R. “Applying reasoning neural networks to the analysis and forecast of Taiwan’s stock index variation,” Taipei Economics Inquiry, Taipei, Vol. 34, No. 2, 1996, pp.171-200.
4. Hanke, M. “Neural network approximation of option-pricing formulas for analytical intractable option-pricing models,” Journal of Computational Intelligence in Finance, Sep./Oct., 1997, pp. 20-27.
5. Hull, J. Options, Futures, and other Derivatives, 3rd edition, Prentice-Hall, Inc., 1997
6. Hutchinson, J., Lo, A. and Poggio, T. “A nonparametric approach to pricing and hedging derivative securities via learning networks,” The Journal of Finance, Vol. XLXI, No. 3, Jul., 1994, pp. 851-859.
7. Jacobs, R.A. “Increased rate of convergence through learning rate adaptation,” Neural Networks, Vol. 1,

1988, pp. 295-307.

8. Lajbcygier, P., Boek, C., Flitman, A., and Palaniswami, M. "Comparing conventional and artificial neural network models for the pricing of options on futures," NeuralVest Journal, Sep., 1996, pp. 16-24.
9. Naimimohasses, R., Barnett, D., Green, D., and Smith, P. "Sensor optimization using neural network sensitivity measures," Measurement Science & Technology, Sep., 1995, pp. 1291-1300.
10. Rumelhart, D., Hinton, G., and Williams, R. "Learning internal representation by error propagation," Parallel Distributed Processing, Cambridge, MA: MIT Press, Vol. 1, 1986, pp. 318-362.
11. Sarkar, D. "Methods to speed up error back-propagation learning algorithm," ACM Computer Surveys, Vol. 27, No. 4, Dec., 1995,pp. 519-542.
12. Steiger, D. and Sharda, R. "Analyzing mathematical models with inductive learning networks," European Journal of Operational Research, 93, 1995, pp.387-401.
13. Takechi, H., Murakami, K. and Izumida, M. "Back propagation learning algorithm with different learning coefficients for each layer," Systems and Computers in Japan, Vol. 26-(7), 1995, pp. 47-56.
14. Tsaih, R. "The softening learning procedure," Mathematical and Computer Modelling, Vol. 18, No. 8, 1993, pp. 61-64.
15. Tsaih, R. "Learning procedure that guarantees obtaining the desired solution of the 2-classes categorization learning problem," Proceedings of The First Asia-Pacific Conference on Simulated Evolution and Learning, Taejon, Korea, 1996, pp. 446-453.
16. Tsaih, R. "Reasoning network networks," In Ellacott, S., J. Mason & I. Anderson (Eds.), Mathematics of Neural Networks: Models, Algorithms and Applications, Kluwer Academic Publishers, London, 1997, pp. 366-371.
17. Tsaih, R. "An Explanation of Reasoning Neural Networks," Mathematical and Computer Modelling, Vol. 28, No. 2, 1998, pp. 37-44.
18. Tsaih, R, Chen, W. and Lin, Y. "Application of reasoning neural networks to financial swaps," Journal of Computational Intelligence in Finance, Vol. 6, No. 3, 1998, pp. 27-37.
19. Tsaih, R., Hsu, Y. and Lai, C. "Forecasting S&P 500 stock index futures with the hybrid AI system," Decision Support Systems, Vol. 23, No. 2, 1998, pp. 161-174.

20. Wang, S. "The unpredictability of standard back propagation neural networks in classification applications," Management Science, Vol.41, No. 3, 1995, pp. 555-559.
21. Yoon, Y., Guimaraes, T. and Swales, G. "Integration artificial neural networks with rule-based expert system," Decision Support Systems, 11, 1994, pp. 497-507.

作者簡介

Ray Tsaih, also known as Rua-Huan Tsaih, is a professor at National Chengchi University, Taipei, Taiwan. He received his Ph.D. in Operations Research in 1991 from University of California, Berkeley. His research interests are Developing new Neural Networks and Applying Neural Networks to Finance. His most recent work has been published in *Computer & Operations Research, Decision Support Systems, Mathematical and Computer Modelling, Advances in Pacific Basin Business, Economics and Finance, Journal of Computational Intelligence in Finance, Mathematics of Neural Networks: Models, Algorithms and Applications, 證券市場發展季刊, 資管評論, 經濟研究,* and *中山管理評論, 住宅學報.*

Hsiou-wei William Lin is a professor at Department of International Business, National Taiwan University. He received his Ph.D. in Business in 1994 from Stanford University. His research interests are Financial Innovation, Financial Statement Analysis, and Financial Forecasts. His most recent work has been published in *Journal of Accounting and Economics, Journal of Financial Studies*, *The Chinese Accounting Review, NTU Management Review, International Journal of Accounting Studies, Review of Securities and Futures Markets*, *Review of Quantitative Finance and Accounting, Asia Pacific Management Review, Journal of Management*, and *Sun Yat-Sen Management Review*.

Yi-ping Lin receive a M.S. degree in 1998, majoring in Management Information System in National Chengchi University, Taipei, Taiwa