Winter 12-10-2002

The Evolution of Internal Representation

Rua-Huan Tsaih

Wen-Chyan Ke

Chia-Yu Liu

The Evolution of Internal Representation

Ray Tsaih, Wen-Chyan Ke, Chia-Yu Liu
Department of Management Information Systems
National Chengchi University
Taipei, Taiwan
tsaih@mis.nccu.edu.tw

Abstract

To develop an appropriate internal representation, a deterministic learning algorithm that has an ability to adjust not only weights but also the number of adopted hidden nodes is proposed. The key mechanisms are (1) the recruiting mechanism that recruits proper extra hidden nodes, and (2) the reasoning mechanism that prunes potentially irrelevant hidden nodes. This learning algorithm can make use of external environmental clues to develop an internal representation appropriate for the required mapping. The encoding problem and the parity problem is used to demonstrate the performance of the proposed algorithm. The experimental results are clearly positive.

Keywords: internal representation, Generalized Delta Rule, recruiting mechanism, pruning mechanism.

1. Introduction

In modern finance, derivatives such as futures and options play increasingly prominent roles in risk management and price speculative activities. Owing to the high-leverage characteristic involved in derivative trading, investors can gain enormous profits with a small amount of capital if they can accurately predict the market's direction. Financial markets, however, can be influenced by many factors, such as, political events, general economic conditions, and traders' expectations. Predicting the financial market's movements is considered to be rather difficult in general. Movements in market prices are not random. Rather, they behave in a highly nonlinear, dynamic manner. The standard random walk assumption of futures prices may merely be a veil of randomness that shrouds a messy nonlinear process (see, for example, [2][4][7]). To make the forecasting of futures prices more reliable, the application of Artificial Neural Networks (ANN), especially the multi-layered feed-forward network [10], have received extensive attention[7][8][15].

Instead of directly deriving the nonlinear equation, these ANN tries to develop an appropriate internal representation for such forecasting problem. In general, a nonlinear forecasting problem is like a problem of finding a nonlinear equation to capture the general pattern of a relationship between the independent variables x_j's and the dependent variables y_l's. The form of the equation is $y_l = F_l(\mathbf{x})$, where $\mathbf{x}$ is the vector of independent variables x_j and F_l is a nonlinear function derived from a given data set of samples $\{({}_1\mathbf{x}, {}_1t_l), \ldots, ({}_N\mathbf{x}, {}_Nt_l)\}$ with ${}_ct_l$ being the observed value of y_l corresponding to ${}_c\mathbf{x}$. In the context of multi-layered feed-forward network, as shown in Figure 1, the information $\mathbf{x}$ coming to the input nodes is recoded into an internal representation $\mathbf{h} \equiv (h_1, h_2, \ldots, h_p)^t$ and the output O_l, the estimated value of y_l, is generated by the internal representation $\mathbf{h}$ rather than by the original pattern $\mathbf{x}$. "Input patterns can always be encoded, if there are enough hidden units, in a form so that the appropriate output pattern can be generated from any input pattern," as mentioned in [10].

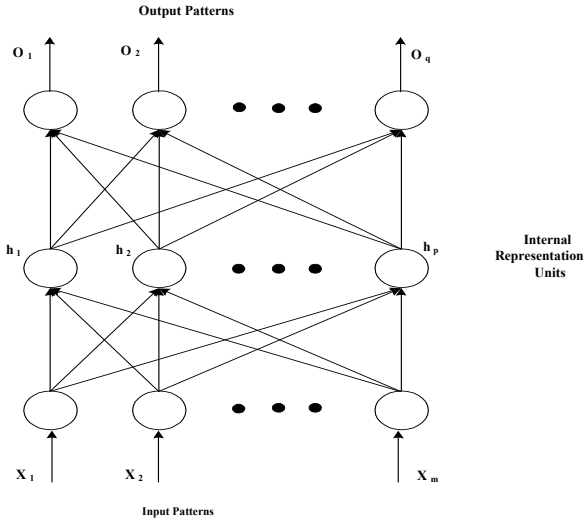

Figure 1. An ANN with a layer of hidden nodes.

Let's take the multi-layered feed-forward network with the hyperbolic tangent (tanh) function [14][10], where $\tanh(x) \equiv \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$, used in all output and hidden nodes as the illustration. Given the c^{th} stimulus ${}_c\mathbf{x}$, the activation value of the i^{th} hidden node $h({}_c\mathbf{x}, {}_2\mathbf{w}_i)$ and the activation values of the l^{th} output node $O({}_c\mathbf{x}, {}_3\mathbf{w}_l, {}_2\mathbf{w})$ are as follows:

$$h({}_c\mathbf{x}, {}_2\mathbf{w}_i) \equiv \tanh({}_2w_{i0} + \sum_{j=1}^{m} {}_2w_{ij}\, {}_cx_j) \quad (1)$$

$$O({}_c\mathbf{x}, {}_3\mathbf{w}_l, {}_2\mathbf{w}) \equiv \tanh({}_3w_{l0} + \sum_{i=1}^{p} {}_3w_{li}\, h({}_c\mathbf{x}, {}_2\mathbf{w}_i))$$

$$= \tanh({}_3w_{l0} + \sum_{i=1}^{p} {}_3w_{li} \tanh({}_2w_{i0} + \sum_{j=1}^{m} {}_2w_{ij}\, {}_cx_j)) \quad (2)$$

where m, p and q are the numbers of input, hidden and output nodes, respectively; ${}_2w_{i0}$ is the bias of the i^{th} hidden

node, ${}_2w_{ij}$ is the weight of connection between the j^{th} input node and the i^{th} hidden node, ${}_3w_{l0}$ is the bias of the l^{th} output node, and ${}_3w_{li}$ is the weight of the connection between i^{th} hidden nodes and the l^{th} output node. Denote the bold character a vector and the superscript t indicates the transposition: ${}_2\mathbf{w_i}^t \equiv ({}_2w_{i0}, {}_2w_{i1}, \ldots, {}_2w_{im})$, ${}_2\mathbf{w}^t \equiv ({}_2\mathbf{w_1}^t, {}_2\mathbf{w_2}^t, \ldots, {}_2\mathbf{w_p}^t)$, ${}_3\mathbf{w_l}^t \equiv ({}_3w_{l0}, {}_3w_{l1}, \ldots, {}_3w_{lp})$, ${}_3\mathbf{w}^t \equiv ({}_3\mathbf{w_1}^t, {}_3\mathbf{w_2}^t, \ldots, {}_3\mathbf{w_q}^t)$, and $\mathbf{w}^t \equiv ({}_2\mathbf{w}^t, {}_3\mathbf{w}^t)$.

Given a set of training samples $\{({}_1\mathbf{x}, {}_1\mathbf{t}), \ldots, ({}_N\mathbf{x}, {}_N\mathbf{t})\}$, the learning goal is to seek a $(\mathbf{w}, p)$ that renders $|O({}_c\mathbf{x}, {}_3\mathbf{w_l}, {}_2\mathbf{w}) - {}_ct_l| < \varepsilon$ for all $c \in \{1, \ldots, N\}$ and all l, where ε is a given acceptable tolerance, says 10^{-6}. In general, the learning can be recognized as a minimization of the sum of residual squares $E(\mathbf{w}, p)$ via setting as

$$\min_{\mathbf{w},p} E(\mathbf{w},p) \equiv \min_{\mathbf{w},p} \sum_{c=1}^{N} \sum_{l=1}^{q} (O({}_c\mathbf{x}, {}_3\mathbf{w_l}, {}_2\mathbf{w}) - {}_ct_l)^2 \quad (3)$$

This is a complicated unconstrained nonlinear programming problem.

The process of an optimization algorithm applied to problem (3) is similar to the process of searching along the surface defined by the sum of residual squares in the $\{\mathbf{w}, p\}$ space composed of all possible $(\mathbf{w}, p)$. Instead of desperately obtaining the globally optimal solution $(\mathbf{w}^*, p^*)$ in a complicated nonlinear environment, many research are motivated to develop a reasonable algorithm for searching an acceptable learning solution $(\mathbf{w}, p)$ that renders $|O({}_c\mathbf{x}, {}_3\mathbf{w_l}, {}_2\mathbf{w}) - {}_ct_l| < \varepsilon$ for all $c \in \{1, \ldots, N\}$ and all l.

Developing such algorithm is not easy due to managing with the following characteristics: (1) the $\{\mathbf{w}, p\}$ space is unbounded since the number of possible $(\mathbf{w}, p)$ is infinite; (2) the surface defined by values of $E(\mathbf{w}, p)$ over the $\{\mathbf{w}, p\}$ space is non-differentiable since, for example, changes in the value of p is discrete and can have discontinuous effects on the value of $E(\mathbf{w}, p)$; (3) the surface is non-analyzable since the mapping from $(\mathbf{w}, p)$ to the value of $E(\mathbf{w}, p)$ is not yet analyzable; (4) the surface is complex and deceptive since values of $E(\mathbf{w}, p)$ with similar $(\mathbf{w}, p)$ may be dramatically different, and with quite different $(\mathbf{w}, p)$ may be very similar. These characteristics lead to hardly having any solid theoretical support in developing a reasonable learning algorithm.

In the context of internal representation, equations (1) and (2) state that (1) the vector of activation values of all hidden nodes $\mathbf{h}(\mathbf{x}, {}_2\mathbf{w}) \equiv (h(\mathbf{x}, {}_2\mathbf{w_1}), \ldots, h(\mathbf{x}, {}_2\mathbf{w_p}))^t$ is the internal representation of the input pattern $\mathbf{x}$, and (2) the activation values of all output nodes are calculated based on this internal representation, i.e., $O({}_c\mathbf{x}, {}_3\mathbf{w_l}, {}_2\mathbf{w}) \equiv O(\mathbf{h}({}_c\mathbf{x}, {}_2\mathbf{w}), {}_3\mathbf{w_l})$ for all l. The activation functions adopted on all hidden nodes are predefined and fixed; usually, they are semi-linear [10]. Thus the internal presentation evolves when the values of p and ${}_2\mathbf{w}$ are altered.

The internal representation is a level-adjacent mapping from the $\{\mathbf{x}\}$ space to the $\{\mathbf{h}\}$ space. Level-adjacent mapping means that level-adjacent points in the previous-layer space are mapped to neighboring points in the latter-layer space [14]. While the nearness of two points in the latter-layer space is measures with their (direct) distance, the level-adjacency between two points in the previous-layer space is measured with the difference of their associated activation level. Owing to the linear characteristic of computing the net input value and the semi-linear characteristic of the activation function adopted in equations (1) and (2), there are level-adjacent mapping from the input layer $\{\mathbf{x}\}$ space to the hidden layer $\{\mathbf{h}\}$ space, and from the hidden layer $\{\mathbf{h}\}$ space to the output layer $\{\mathbf{O}\}$ space.

In the learning stage, the position of the training stimuli $\{{}_1\mathbf{x}, \ldots, {}_N\mathbf{x}\}$ in the $\{\mathbf{x}\}$ space are given and fixed; however, $\{\mathbf{h}({}_1\mathbf{x}, {}_2\mathbf{w}), \ldots, \mathbf{h}({}_N\mathbf{x}, {}_2\mathbf{w})\}$ are determined by p and ${}_2\mathbf{w}$, and each $O({}_c\mathbf{x}, {}_3\mathbf{w_l}, {}_2\mathbf{w})$ is determined with $\mathbf{h}({}_c\mathbf{x}, {}_2\mathbf{w})$ and ${}_3\mathbf{w_l}$. As stated in [14], (1) no matter what kind of learning algorithm is used, the resulting mapping between two consecutive layers is always a level-adjacent mapping; (2) the crux of learning is to adjust p and ${}_2\mathbf{w}$ to render the internal representation appropriate for the learning task. An internal representation is appropriate for the task if there is a ${}_3\mathbf{w}$ such that $|O(\mathbf{h}({}_c\mathbf{x}, {}_2\mathbf{w}), {}_3\mathbf{w_l}) - {}_ct_l| < \varepsilon$ for all $c \in \{1, \ldots, N\}$ and all l. Although the learning of ANN should derive appropriate p and ${}_2\mathbf{w}$ from the training samples to obtain an appropriate internal representation, the development of the internal representation is relevant with the current value of p and $\mathbf{w}$, not the current value of p and ${}_2\mathbf{w}$.

In literatures, there are two categories of learning algorithms for multi-layered feed-forward network: the evolutionary ANN (EANN) algorithms, which are stochastic, and the weight-and-structure-change learning algorithms, which are deterministic.

Over past years, researchers have applied evolutionary computation to problems whose solution space is so large and highly complex that it is difficult to employ conventional optimization procedures to search for a global optimum. Evolutionary computation refers to a collection of stochastic searching algorithms whose designs are based upon the ideas of genetic inheritance and the Darwinian principle of the survival of the fittest (natural selection). There are several different styles of evolutionary algorithms: evolutionary strategies (ES), evolutionary programming (EP), genetic algorithms (GA), and genetic programming (GP). All of them model the searching process over the solution space by mimicking a biological evolution process. They differ mainly in the evolution operators involved and the representation of the solution space. Most researchers believe that, basically, evolutionary computation should not be considered as a kind of optimization technique to compete with other alternative techniques, but an optimization principle to be incorporated into existing techniques. Thus, they propose to apply EP, GA and GP to the determination of the network structure of an ANN. This gives rise to three classes of ANN, EPNN, GANN, and GPNN. All of these classes are portions of EANN. The most promising EANNs involve a global search algorithm that is stochastic [17].

In contrast, all weight-and-structure-change learning algorithms alter $\mathbf{w}$ and p in a deterministic way; for example, the tiling algorithm [9], the cascade-correlation

(CC) algorithm [5], the upstart algorithm [6], the W&S algorithm [16], the CTN algorithm [3], and the softening algorithm [12][13][14]. Without following the ideas of genetic inheritance and natural selection, they adjust the network structure in one of the following ways:

(1) Destructively: using excess hidden nodes initially and pruning (removing) least effective hidden nodes during the learning process; e.g., W&S algorithm and CTN algorithm;

(2) Constructively: using less hidden nodes initially and recruiting (adding) more hidden nodes during the learning process; e.g., the tiling algorithm, CC algorithm, and the upstart algorithm;

(3) Aggregately: using only one hidden node initially, and recruiting as well as pruning hidden nodes during the learning process; e.g., the softening algorithm.

These deterministic learning algorithms have an ability to develop an appropriate internal representation via recruiting/pruning hidden nodes and altering the weights during the learning process.

Here we introduce a deterministic learning algorithm that makes use of sequentially presented training samples to adjust the values of **w** and p to develop an internal representation appropriate for the required mapping. More over, this learning algorithm guarantees an acceptable learning result, without the binary output restriction. Recall that a learning result is acceptable if $|O({}_c\mathbf{x}, {}_3\mathbf{w}_l, {}_2\mathbf{w}) - {}_ct_l| < \varepsilon$ for all $c \in \{1, \ldots, N\}$ and all l, where ε is a given acceptable tolerance.

This paper is organized as follows. The proposed learning algorithm and its theoretical justification are introduced in Section 2. Empirical justification of the proposed algorithm is given in Section 3. The encoding problem and the parity problem [11] will be used to demonstrate the performance of the proposed algorithm. Finally, conclusions and future work are presented in Section 4. For the simplicity of presentation, all theoretical proofs are given in the Appendix.

2. The Proposed Learning Algorithm

To simplify the presentation, without lose of the generality, we let q = 1 in the explanation of our design. Table 1 presents the general procedure and Figure 2 displays the flow chart of the proposed algorithm. The key mechanisms are (1) the recruiting mechanism that effectively recruits proper extra hidden nodes, and (2) the reasoning mechanism that effectively prunes potentially irrelevant hidden nodes. The details of the proposed algorithm at each step are given below.

The pairs of $({}_c\mathbf{x}, {}_ct)$ are presented sequentially. At the k^{th} stage, the stage when the k^{th} sample $({}_k\mathbf{x}, {}_kt)$ enters, the goal is to seek the values of (**w**, p) so that

$$|O({}_c\mathbf{x}, \mathbf{w}) - {}_ct| < \varepsilon \text{ for all } c \in I(k) \equiv \{1, \ldots, k\} \qquad (4)$$

The learning proceeds via evolving the internal representation to render it appropriate for accomplishing the goal (4). The internal representation that can accomplish the goal (4) is an appropriate one.

Table 1. The proposed deterministic algorithm.

Step 0: Set one hidden node with weights assigned randomly; set k = 1.
Step 1: If it is the end of the sample input sequence, STOP
Step 2: Present the k^{th} given sample $({}_k\mathbf{x}, {}_kt)$.
Step 3: If $|O({}_k\mathbf{x}, \mathbf{w}) - {}_kt| > \varepsilon$, then
  Step 3.1: Store the weights.
  Step 3.2: Apply the weight-tuning mechanism to adjust weights until one of the following cases occurs:
  (1)If $|O({}_c\mathbf{x}, \mathbf{w}) - {}_ct| < \varepsilon \ \forall\ c \in I(k)$, then go to Step 4.
  (2)If an unacceptable result is obtained, then
    (a) set λ=1.
    (b) Restore the weights.
    (c) p+2 → p and recruit two extra hidden nodes with ${}_2\mathbf{w}_{p-1}{}^t = (\zeta - \lambda\boldsymbol{\alpha}^t {}_k\mathbf{x}, \lambda\boldsymbol{\alpha}^t)$, ${}_2\mathbf{w}_p{}^t = (\zeta + \lambda\boldsymbol{\alpha}^t {}_k\mathbf{x}, -\lambda\boldsymbol{\alpha}^t)$, $\zeta = 10^{-6} \min_{c \in I(k-1)} |\boldsymbol{\alpha}^t({}_k\mathbf{x} - {}_c\mathbf{x})|$, ${}_3w_{p-1} = {}_3w_p = (\tanh^{-1}({}_kt) - {}_3w_0 - \sum_{i=1}^{p-2} {}_3w_i\, h({}_k\mathbf{x}, {}_2\mathbf{w}_i))/2\tanh(\zeta)$, where $\boldsymbol{\alpha}$ is a unit vector that $\boldsymbol{\alpha}^t({}_k\mathbf{x} - {}_c\mathbf{x}) \neq 0\ \forall\ c \in I(k-1)$.
    (d) Apply the weight-tuning mechanism to adjust weights until one of the following cases occurs:
      (i) If $|O({}_c\mathbf{x}, \mathbf{w}) - {}_ct| < \varepsilon \ \forall\ c \in I(k)$, then go to Step 4.
      (ii) If an unacceptable result is obtained, let λ*2 → λ and p-2 → p, then go to (b).
Step 4: Prune all potentially irrelevant hidden nodes.
Step 5: k+1 → k; go to Step 1.

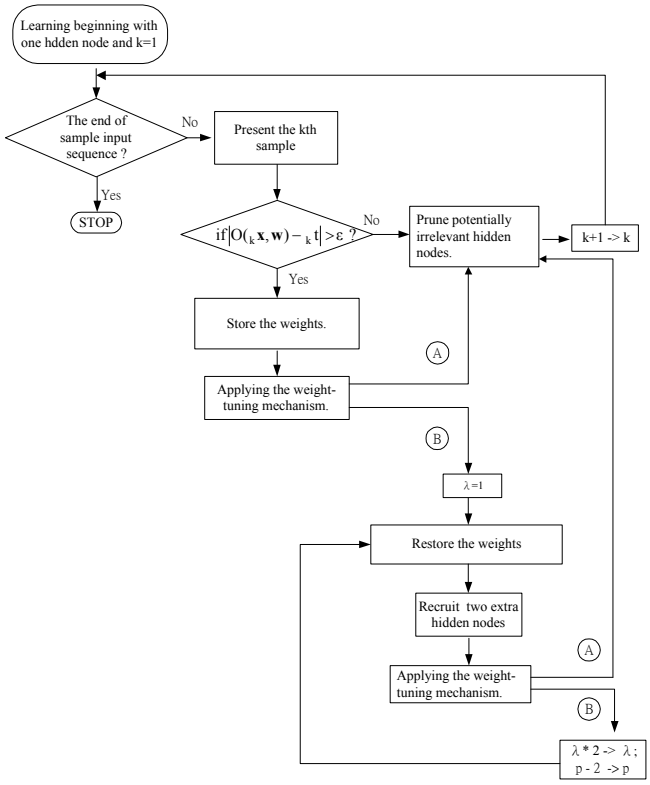

Figure 2. The flow chart of the proposed learning algorithm.

Note that, obtaining an appropriate internal representation is a necessary condition, but not a sufficient condition, of accomplishing the goal (4).

However, the effort of checking regularly if the current internal representation is appropriate is huge. Furthermore, it is useless when the current network structure is a defective one, i.e., the number of adopted hidden nodes is less than the necessary number of hidden nodes for accomplishing the goal (4). These arguments account for the reason of adopting accomplishing the goal (4), instead of obtaining an appropriate internal representation, as the stopping criterion.

The algorithm ensures that goal (4) is accomplished at the end of each stage. Consequently, it guarantees an acceptable learning result at the end.

Specifically, when the k^{th} sample $({}_k\mathbf{x}, {}_kt)$ enters, we first check if the goal (4) is accomplished. If so, the current internal representation is appropriate and there is only a reasoning effort involved. Then the next given sample is presented. If not, in our next step (Step 3), the weight-tuning mechanism implementing the momentum version of the generalized delta rule [10] with automatic adjustment of learning rate [13] is applied to $\min_{\mathbf{w}} E_k(\mathbf{w})$ to adjust weights, where $E_k(\mathbf{w}) \equiv \sum_{c\in I(k)} (\tanh({}_3w_0 + \sum_{i=1}^{p} {}_3w_i \tanh({}_2w_{i0} + \sum_{j=1}^{m} {}_2w_{ij}\ {}_cx_j)) - {}_ct)^2$. Namely, the objective function used in the optimization process at the k^{th} stage $E_k(\mathbf{w})$ is now defined as the current sum of residual squares and the parameter p remains the same. Such a weight-tuning mechanism attempts to achieve the goal (4).

(Rumelhart, Hinton & Williams, 1986a) has claimed that a mechanism that implements the generalized delta rule can "learn internal representations by error propagation." Unfortunately, this weight-tuning mechanism has the power to alter the weights, yet no power to add/delete hidden nodes. More over, this mechanism may converge to the neighborhood of an undesired attractor of $\min_{\mathbf{w}} E_k(\mathbf{w})$ in which $\nabla_{\mathbf{w}} E_k(\mathbf{w}) = \mathbf{0}$; for example, a relatively optimal solution or a saddle point solution. Another possible failure is the case that the current network structure is a defective one. All of the above lead to an unacceptable result. These were indicated in Figure 3: Path B indicates the situation when the result of implementing the weight-tuning mechanism is an unacceptable one; while Path A indicates the situation when the result is an acceptable one.

A perfect weight-tuning mechanism that can avoid the predicament of converging to an undesired attractor is desirable, because the defective network structure will be the only cause of obtaining an unacceptable result. Unfortunately, there is no such perfect weight-tuning mechanism currently. Under this confinement, together with the consideration of the computing complexity, the current weight-tuning mechanism is adopted. The consideration of the computing complexity is very important because the weight-tuning mechanism will be triggered very frequently during the learning process.

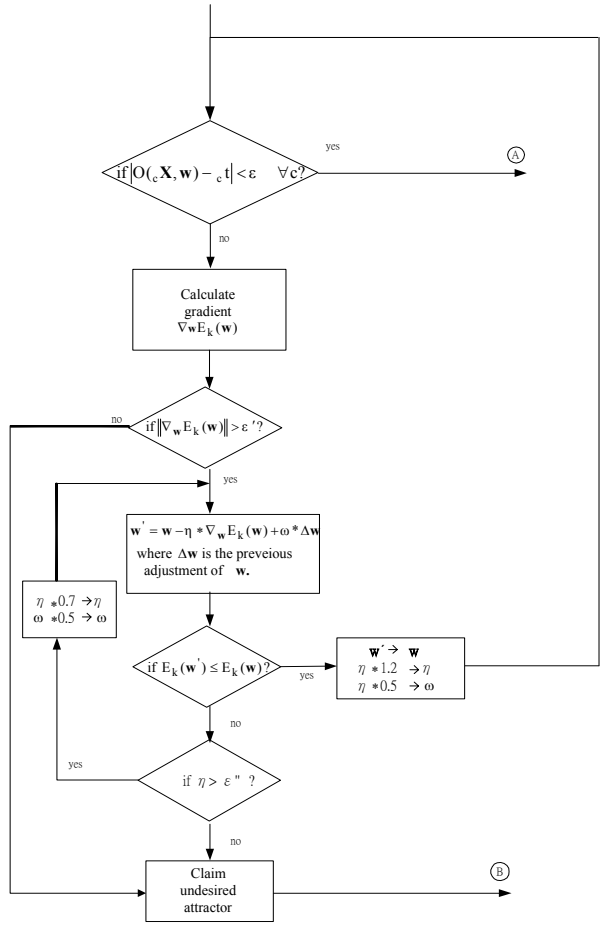

Figure 3. The flow chart of the weight-tuning mechanism implementing the momentum version of the generalized delta rule with automatic adjustment of learning rate. ε' and ε'' are given tiny numbers.

Recruiting extra hidden nodes is adopted to handle these problems in Path B of Figure 3 as follows. Action (b) in Step 3.2 restores the weights stored in Step 3.1. Assume the goal of the previous stage is accomplished at the end of the previous stage. Then, by restoring the weights stored in Step 3.1, we return to the internal representation that renders $|O({}_c\mathbf{x}, \mathbf{w}) - {}_ct| < \varepsilon$ for all $c \in I(k-1)$ and $|O({}_k\mathbf{x}, \mathbf{w}) - {}_kt| \geqq \varepsilon$. For Action (c) in Step 3.2, there is a recruiting mechanism arranged to recruit two extra hidden nodes with a gain parameter λ whose value is initially 1 set via Action (a) in Step 3.2. These two new-added hidden nodes, the $p\text{-}1^{th}$ and p^{th} ones, have weights ${}_2\mathbf{w}_{p-1}{}^t = (\zeta - \lambda\boldsymbol{\alpha}^t\ {}_k\mathbf{x}, \lambda\boldsymbol{\alpha}^t)$, ${}_2\mathbf{w}_p{}^t = (\zeta + \lambda\boldsymbol{\alpha}^t\ {}_k\mathbf{x}, -\lambda\boldsymbol{\alpha}^t)$, $\zeta = 10^{-6} \min_{c\in I(k-1)} |\boldsymbol{\alpha}^t({}_k\mathbf{x} - {}_c\mathbf{x})|$, ${}_3w_{p-1} = {}_3w_p = (\tanh^{-1}({}_kt) - {}_3w_0 - \sum_{i=1}^{p-2} {}_3w_i\ h({}_k\mathbf{x}, {}_2\mathbf{w}_i))/2\tanh(\zeta)$, where $\boldsymbol{\alpha}$ is a unit vector that $\boldsymbol{\alpha}^t({}_k\mathbf{x} - {}_c\mathbf{x}) \neq 0\ \forall\ c \in I(k-1)$. For Action (d) in Step 3.2, the λ value is fixed and the weight-tuning mechanism is applied to render the goal (4) accomplished. If an unacceptable result is obtained, we multiply the λ value by 2, and repeat Actions (b), (c), and (d). The Actions (b), (c), and (d) are repeated until the goal (4) is accomplished.

The arrangement of Actions (b), (c) and (d) is capable of solving (1) the defectiveness of the network structure via adding two hidden nodes, and (2) the predicament of converging to an undesired attractor via introducing extra dimensions in the weight space such that the trapped attractor could be no longer an attractor in the new weight space.

Recruiting two extra hidden nodes has introduced two extra dimensions in the {**h**} space, thus has introduced two extra dimensions in the internal representation. The arrangement of $_2\mathbf{w}_{p-1}$ and $_2\mathbf{w}_p$ has made the new corresponding point $\mathbf{h}(_k\mathbf{x}, _2\mathbf{w})$ placed at the positive side of each of these two new-added dimensions, and all other new corresponding points $\mathbf{h}(_c\mathbf{x}, _2\mathbf{w})$'s placed at the positive side of one new-added dimension and at the negative side of the other new-added dimension. A large λ value renders the behavior of the activation functions of the newly recruited p-1th and pth hidden nodes similar to the behavior of a threshold function, and results in a phenomenon that $h(_c\mathbf{x}, _2\mathbf{w}_{p-1})$ and $h(_c\mathbf{x}, _2\mathbf{w}_p)$ numerically equal 1 or –1 for almost all $c \in I(k-1)$. Thus, as stated in Lemma 1, the arrangement of such two new-added hidden nodes with a large λ value has made the new corresponding point $\mathbf{h}(_k\mathbf{x}, _2\mathbf{w})$ placed near the $\tanh(\zeta)$ position of each of these two new-added dimensions, and all other new corresponding points $\mathbf{h}(_c\mathbf{x}, _2\mathbf{w})$'s placed near the –1 corner of one new-added dimension and near the +1 corner of the other new-added dimension. Therefore, as stated in Lemma 2, if the internal representation is appropriate to render $|O(_c\mathbf{x}, \mathbf{w}) - _ct| < \varepsilon$ for all $c \in I(k-1)$ and $|O(_k\mathbf{x}, \mathbf{w}) - _kt| \geqq \varepsilon$, after recruiting such two hidden nodes, the new internal representation can be appropriate for all k training samples. In other words, the arrangement of such two new-added hidden nodes with a large λ value is capable of solving the defectiveness of the network structure via adding two hidden nodes.

It is difficult to identify the scenario (the sample input sequence and the value of λ) for which the weight-tuning mechanism will (or will not) achieve the goal (4), because the surface defined by $E_k(\mathbf{w})$ over the weight space is not possible to analyze. However, Lemma 2 shows that the goal (4) can be accomplished immediately via recruiting merely two extra hidden nodes with proper weights and λ. More over, Lemma 2 shows that there is no infinite loop in Step 3.2. In other words, the arrangement of Actions (b), (c) and (d) is capable of solving the predicament of converging to an undesired attractor via introducing extra dimensions in the weight space such that the trapped attractor could be no longer an attractor in the new weight space.

Therefore, Step 3.2 does ensure that the goal of each stage is accomplished at the end of each stage, thus guaranteeing an acceptable learning result obtained at the end.

The recruiting mechanism handles the situation of encountering an unacceptable result without involving the reason. A defective network structure triggers the recruiting mechanism; the situation of converging to an undesired attractor also triggers the recruiting mechanism. The recruiting mechanism triggered due to the situation of converging to an undesired attractor may recruit excess hidden nodes that later become irrelevant. At each stage, a hidden node is irrelevant if the goal (4) is still accomplished with this hidden node being deleted. The irrelevant hidden nodes are useless with respect to the goal (4); furthermore, they may contribute significant effort to the performance of network and result in bad generalization ability. More over, more samples typically lead to more concise information about the appropriate internal representation, and thus fewer hidden nodes are required. It is therefore necessary to prune irrelevant hidden nodes, and an internal representation is better if it accomplishes the goal (4) with smaller amount of adopted hidden nodes.

In Step 4, a reasoning mechanism is arranged to prune all potentially irrelevant hidden nodes. At the kth stage, a hidden node is potentially irrelevant if it is deleted and the goal (4) can be accomplished via applying the weight-tuning mechanism. In Step 4, every hidden node is checked whether it is potentially irrelevant. Each potentially irrelevant hidden node is deleted after being identified.

3. The Performance and Analysis of the Proposed Algorithm

Here we use two popular examples to examine how the current arrangement for the recruiting and reasoning mechanisms works: the encoding problem [1][11] and the parity problem [11].

[1] has posed the encoding problem where a set of N orthogonal input patterns are mapped to a set of N orthogonal output patterns through a small set of hidden nodes. Such problem requires a rather efficient way in encoding an N bit pattern into a small set of hidden nodes and then decoding this (internal) representation into the output pattern. [11] has proposed that a set of N orthogonal input patterns are mapped to a set of N orthogonal output patterns through a small set of $\log_2 N$ hidden nodes. The reason behind such design is that if the hidden nodes take on binary values, the hidden nodes must form a binary number to encode each of the input patterns. They present an encoding problem with an 8 input patterns, 8 output patterns, and 3 hidden nodes, and, as shown in Table 2, they find the learning system develop solutions that use the intermediate values.

Table 2. The mapping of the encoding problem generated in [11].

Input Patterns		Hidden Node Patterns				Output Patterns
1 0 0 0 0 0 0 0	→	0.5	0	0	→	1 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0	→	0	0	0.5	→	0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0	→	0.5	0	1	→	0 0 1 0 0 0 0 0
0 0 0 1 0 0 0 0	→	1	0	0.5	→	0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0	→	1	1	1	→	0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0	→	0	1	0	→	0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0	→	1	1	0	→	0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1	→	0	1	1	→	0 0 0 0 0 0 0 1

Table 3. The mapping of the encoding problem generated by the proposed algorithm.

Input Patterns		Hidden Node Patterns			Output Patterns
1 0 0 0 0 0 0 0	→	0.1452	-0.9939	→	1 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0	→	-0.1451	0.9926	→	0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0	→	0.8070	-0.5973	→	0 0 1 0 0 0 0 0
0 0 0 1 0 0 0 0	→	-0.8416	0.6074	→	0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0	→	0.9931	0.1794	→	0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0	→	-0.9920	-0.2055	→	0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0	→	0.6247	0.8334	→	0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1	→	-0.6150	-0.8390	→	0 0 0 0 0 0 0 1

Our simulation result is encouraging: the proposed algorithm employs the ability of the intermediate values in a more efficient way. Table 3 shows the mapping of the encoding problem generated by the proposed algorithm. The proposed algorithm also uses intermediate values to result in an appropriate internal representation. The appropriate internal representations shown in Table 2 and Table 3 are similar, except that the former uses three hidden nodes and the latter uses two hidden nodes. It seems that the reasoning mechanism effectively prunes a potentially irrelevant hidden node, which is likely to be the middle one in Table 2.

Table 4. The mapping of the 4-bit parity problem generated in [11] and by the proposed algorithm.

Input Patterns		Hidden Node Patterns generated in [11]	Hidden Node Patterns generated by the proposed algorithm				Output Patterns
0 0 0 0	→	1 1 1 1	0.8934	-0.9999	-0.9998	→	0
1 0 0 0	→	1 0 1 1	0.5254	-0.9371	-0.9972	→	1
0 1 0 0	→	1 0 1 1	0.5252	-0.9369	-0.9972	→	1
0 0 1 0	→	1 0 1 1	0.5252	-0.9367	-0.9972	→	1
0 0 0 1	→	1 0 1 1	0.5251	-0.9366	-0.9972	→	1
1 1 0 0	→	1 0 1 0	-0.2646	0.9338	-0.9630	→	0
1 0 1 0	→	1 0 1 0	-0.2647	0.9340	-0.9630	→	0
1 0 0 1	→	1 0 1 0	-0.2648	0.9341	-0.9630	→	0
0 1 1 0	→	1 0 1 0	-0.2649	0.9343	-0.9630	→	0
0 1 0 1	→	1 0 1 0	-0.2650	0.9344	-0.9630	→	0
0 0 1 1	→	1 0 1 0	-0.2651	0.9345	-0.9630	→	0
1 1 1 0	→	0 0 1 0	-0.8097	0.9999	-0.5882	→	1
1 1 0 1	→	0 0 1 0	-0.8097	0.9999	-0.5881	→	1
1 0 1 1	→	0 0 1 0	-0.8097	0.9999	-0.5881	→	1
0 1 1 1	→	0 0 1 0	-0.8098	0.9999	-0.5879	→	1
1 1 1 1	→	0 0 0 0	-0.9626	1.0000	0.5627	→	0

Table 4 shows the mapping of the 4-bit parity problem generated in [11] and by the proposed algorithm. [11] has proposed m hidden nodes required for the m-bit parity problem, and noted that "the internal representation created by the learning rule is to arrange that the number of hidden units that come on is equal to the number of zeros in the input and that the particular hidden units that come on depend only on the number, not on which input units are on." By contrast, as shown in Table 4, the appropriate internal representations developed in [11] and by the proposed algorithm are similar, except that the former uses four hidden nodes and the latter uses three hidden nodes. The appropriate internal representations developed by the proposed algorithm is to arrange that these three hidden nodes also activates also via through detecting the number of zeros in the input, not on which input units are on.

Table 5 shows the summary of the simulation results of m-bit parity problem, with the value of m from 2 to 6. For each value of m, there are 100 cases, each with a different randomly-generated input sequence. As shown in Table 5, the average final number of adopted hidden nodes for the m-bit parity problems is smaller than m and the final number of adopted hidden nodes is less than m in most cases.

Table 5. The minimum, maximum, mean and standard deviation of number of adopted hidden nodes for the m-bit parity problem, with the value of m from 2 to 6. For each value of m, there are 100 cases, each with a different randomly-generated input sequence.

m	2	3	4	5	6
Minimum	2	2	3	3	4
Mean	2	2.28	3.38	3.47	5.65
Maximum	2	4	5	5	17
Standard Deviation	0	0.5924	0.5464	0.6428	2.3154

In summary, it seems that the current arrangement of the recruiting and reasoning mechanisms in the proposed algorithm works well that the internal representations evolves in a better way than the one developed by the generalized delta rule. Recall that an appropriate internal representation with smaller amount of adopted hidden nodes is better.

4. Discussions and Future Work

In this paper, a deterministic learning algorithm that guarantees an acceptable learning result is proposed. The proposed algorithm does not follow the ideas of genetic inheritance and natural selection. During the process of the proposed algorithm, the internal representation evolves in a deterministic way to an appropriate one. The key mechanisms of the proposed algorithm are (1) the recruiting mechanism that can recruit proper extra hidden nodes, and (2) the reasoning mechanism that can prune potentially irrelevant hidden nodes. We provide theoretical justification to explain why the proposed algorithm guarantees an acceptable learning result.

The experimental results show that the proposed algorithm also employs a similar ability of developing the internal representation with the one shown in [11]. This is not surprised since the proposed algorithm also adopts in its weight-tuning mechanism the generalized delta rule proposed in [10]. However, the experimental results show that the current arrangement of the recruiting and reasoning mechanisms in the proposed algorithm works

well that the internal representations evolves in a better way than the internal representation developed by the generalized delta rule.

There is a criticism when we apply the ANN to practical problems: a black box obtained from the learning. A further study is to explore the appropriate internal representation obtained from the learning to provide the knowledge behind the application and to get rid of that criticism.

The simulation results show that the number of adopted hidden nodes is a function of the sample input sequence. As expected, some sample input sequences cause difficulties in the associated processes, due to encountering a defective network structure or the predicament of converging to an undesired attractor. A further investigation (theoretically or numerically) on the surface defined by $E_k(\mathbf{w})$ over the weight space is currently under study to identify the scenario (the sample input sequence and the value of λ) for which the weight-tuning mechanism will (or will not) achieve the goal (4). Another study involves exploring (1) a better recruiting mechanism to recruit hidden nodes and (2) a better reasoning mechanism to prune potentially irrelevant hidden nodes in a much more efficient way.

5. Appendix: Theoretical Supports and their Proofs.

Lemma 1: Let $\{{}_c\mathbf{x} | \forall\ c \in I(k)\}$ be given, and assume ${}_2\mathbf{w}_{p-1}{}^t = (\zeta - \lambda\boldsymbol{\alpha}^t{}_k\mathbf{x},\ \lambda\boldsymbol{\alpha}^t)$, ${}_2\mathbf{w}_p{}^t = (\zeta + \lambda\boldsymbol{\alpha}^t{}_k\mathbf{x},\ -\lambda\boldsymbol{\alpha}^t)$, where λ is a given large number. Then, there exists a unit vector $\boldsymbol{\alpha}$ and a tiny positive number ζ that render $h({}_c\mathbf{x}, {}_2\mathbf{w}_{p-1}) + h({}_c\mathbf{x}, {}_2\mathbf{w}_p) \cong 0.0$ (Hereafter, $F(x) \cong y$ means that, numerically, the value of $F(x)$ is y.) $\forall\ c \in I(k-1)$ and $h({}_k\mathbf{x}, {}_2\mathbf{w}_{p-1}) + h({}_k\mathbf{x}, {}_2\mathbf{w}_p) = 2\tanh(\zeta)$.

Proof:

$\because\ {}_2\mathbf{w}_{p-1}{}^t = (\zeta - \lambda\boldsymbol{\alpha}^t{}_k\mathbf{x},\ \lambda\boldsymbol{\alpha}^t)$,

$\therefore\ {}_2w_{p-10} + \sum_{j=1}^{m} {}_2w_{p-1j}\ {}_cx_j = \zeta + \lambda(\boldsymbol{\alpha}^t{}_c\mathbf{x} - \boldsymbol{\alpha}^t{}_k\mathbf{x})$.

Similarly, ${}_2w_{p0} + \sum_{j=1}^{m} {}_2w_{pj}\ {}_cx_j = \zeta + \lambda(\boldsymbol{\alpha}^t{}_k\mathbf{x} - \boldsymbol{\alpha}^t{}_c\mathbf{x})$.

$\because\ \{{}_c\mathbf{x} | \forall\ c \in I(k)\}$ is given,

$\therefore\ {}_k\mathbf{x} - {}_c\mathbf{x}$ is known for every $c \in I(k-1)$.

With a reasonable assumption that the amount of samples is finite, i.e., I(k) is a finite set, there exists a unit vector $\boldsymbol{\alpha}$ that $\boldsymbol{\alpha}^t({}_k\mathbf{x} - {}_c\mathbf{x}) \neq 0\ \forall\ c \in I(k-1)$. Then, ζ is assigned as $10^{-6} \min_{c \in I(k-1)} |\boldsymbol{\alpha}^t({}_k\mathbf{x} - {}_c\mathbf{x})|$.

Since λ is a large value and $\zeta = 10^{-6} \min_{c \in I(k-1)} |\boldsymbol{\alpha}^t({}_k\mathbf{x} - {}_c\mathbf{x})|$,

$h({}_c\mathbf{x}, {}_2\mathbf{w}_{p-1}) = \tanh(\zeta + \lambda(\boldsymbol{\alpha}^t{}_c\mathbf{x} - \boldsymbol{\alpha}^t{}_k\mathbf{x})) \cong \tanh(\lambda(\boldsymbol{\alpha}^t{}_c\mathbf{x} - \boldsymbol{\alpha}^t{}_k\mathbf{x}))$ and $h({}_c\mathbf{x}, {}_2\mathbf{w}_p) = \tanh(\zeta + \lambda(\boldsymbol{\alpha}^t{}_k\mathbf{x} - \boldsymbol{\alpha}^t{}_c\mathbf{x})) \cong -\tanh(\lambda(\boldsymbol{\alpha}^t{}_c\mathbf{x} - \boldsymbol{\alpha}^t{}_k\mathbf{x}))$. Thus, $h({}_c\mathbf{x}, {}_2\mathbf{w}_{p-1}) + h({}_c\mathbf{x}, {}_2\mathbf{w}_p) \cong 0.0\ \forall c \in I(k-1)$.

$\because\ h({}_k\mathbf{x}, {}_2\mathbf{w}_{p-1}) = h({}_k\mathbf{x}, {}_2\mathbf{w}_p) = \tanh(\zeta)$,

$\therefore\ h({}_k\mathbf{x}, {}_2\mathbf{w}_{p-1}) + h({}_k\mathbf{x}, {}_2\mathbf{w}_p) = 2\tanh(\zeta)$.

Q.E.D.

Lemma 2: Assume $O({}_c\mathbf{x}, \mathbf{w}) = \tanh({}_3w_0 + \sum_{i=1}^{p-2} {}_3w_i\ h({}_c\mathbf{x}, {}_2\mathbf{w}_i))$, $|O({}_c\mathbf{x}, \mathbf{w}) - {}_ct| < \varepsilon\ \forall\ c \in I(k-1)$, and $|O({}_k\mathbf{x}, \mathbf{w}) - {}_kt| \geqq \varepsilon$. With recruiting two hidden nodes, $h({}_c\mathbf{x}, {}_2\mathbf{w}_{p-1}) \equiv \tanh({}_2w_{p-10} + \sum_{j=1}^{m} {}_2w_{p-1j}\ {}_cx_j)$ and $h({}_c\mathbf{x}, {}_2\mathbf{w}_p) \equiv \tanh({}_2w_{p0} + \sum_{j=1}^{m} {}_2w_{pj}\ {}_cx_j)$, the new value of $O({}_c\mathbf{x}, \mathbf{w})$ equals $\tanh({}_3w_0 + \sum_{i=1}^{p-2} {}_3w_i\ h({}_c\mathbf{x}, {}_2\mathbf{w}_i) + {}_3w_{p-1}\ h({}_c\mathbf{x}, {}_2\mathbf{w}_{p-1}) + {}_3w_p\ h({}_c\mathbf{x}, {}_2\mathbf{w}_p))$. Then, there exist ${}_2\mathbf{w}_{p-1}$, ${}_2\mathbf{w}_p$, ${}_3w_{p-1}$ and ${}_3w_p$ that render the new value of $|O({}_c\mathbf{x}, \mathbf{w}) - {}_ct| < \varepsilon\ \forall\ c \in I(k)$.

Proof:

Let ${}_cy'$ and ${}_cy$ be the values of $O({}_c\mathbf{x}, \mathbf{w})$ before and after introducing two hidden nodes, respectively. Also let ${}_cnet$ be the value of ${}_3w_0 + \sum_{i=1}^{p-2} {}_3w_i\ h({}_c\mathbf{x}, {}_2\mathbf{w}_i)$ before introducing two hidden nodes. Thus, ${}_cy' = \tanh({}_cnet)$ and ${}_cy = \tanh({}_cnet + {}_3w_{p-1}\ h({}_c\mathbf{x}, {}_2\mathbf{w}_{p-1}) + {}_3w_p\ h({}_c\mathbf{x}, {}_2\mathbf{w}_p))$.

$\because\ |O({}_c\mathbf{x}, \mathbf{w}) - {}_ct| < \varepsilon\ \forall\ c \in I(k-1)$ and $|O({}_k\mathbf{x}, \mathbf{w}) - {}_kt| \geqq \varepsilon$ before introducing two hidden nodes,

$\therefore\ |{}_cy' - {}_ct| < \varepsilon\ \forall\ c \in I(k-1)$ and $|{}_ky' - {}_kt| \geqq \varepsilon$.

Let λ, ${}_2\mathbf{w}_{p-1}$ and ${}_2\mathbf{w}_p$ be assigned as in Lemma 1. Thus, from Lemma 1, $h({}_c\mathbf{x}, {}_2\mathbf{w}_{p-1}) + h({}_c\mathbf{x}, {}_2\mathbf{w}_p) \cong 0.0\ \forall\ c \in I(k-1)$ and $h({}_k\mathbf{x}, {}_2\mathbf{w}_{p-1}) + h({}_k\mathbf{x}, {}_2\mathbf{w}_p) = 2\tanh(\zeta)$. Let ${}_3w_{p-1} = {}_3w_p = \gamma$, where $\gamma = (\tanh^{-1}({}_kt) - {}_knet)/2\tanh(\zeta)$.

$\because\ {}_3w_{p-1} = {}_3w_p$ and $h({}_c\mathbf{x}, {}_2\mathbf{w}_{p-1}) + h({}_c\mathbf{x}, {}_2\mathbf{w}_p) \cong 0.0\ \forall\ c \in I(k-1)$,

$\therefore\ {}_cy \cong {}_cy'\ \forall\ c \in I(k-1)$.

Therefore, $|{}_cy - {}_ct| < \varepsilon\ \forall\ c \in I(k-1)$ since $|{}_cy' - {}_ct| < \varepsilon\ \forall\ c \in I(k-1)$.

$\because\ h({}_k\mathbf{x}, {}_2\mathbf{w}_{p-1}) + h({}_k\mathbf{x}, {}_2\mathbf{w}_p) = 2\tanh(\zeta)$,

$\therefore\ {}_ky = \tanh({}_knet + \gamma\ 2\tanh(\zeta)) = {}_kt$. Thus $|{}_ky - {}_kt| < \varepsilon$.

Q.E.D.

References

[1] Ackley, D., Hinton, G., & Sejnowski, T. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, 9, 147—169.

[2] Blank, S. (1991). Chaos in futures market? A nonlinear dynamical analysis. J. *Futures Markets*, 11, 711—728.

[3] Chen, **Y**., Thomas, D., & Nixon, M. (1994). Generating-Shrinking Algorithm for Learning Arbitrary Classification. *Neural Networks*, 7, 1477—1489.

[4] DeCoster, G., Labys, W., & Mitchell, D. (1992). Evidence of chaos in commodity futures prices. *J. Futures Markets*, 12, 291—305.

[5] Fahlman, S., & Lebiere, C. (1990) *The Cascade-Correlation Learning Architecture. In Touretzky, D. (Eds.), Advances in*

Neural Information Processing Systems II (Denver, 1989), San Mateo: Morgan Kaufmann.

[6] Frean, M. (1990). The Upstart Algorithm: A Method for Constructing and Training Feedforward Neural Networks. *Neural Computation*, 2, 198－209.

[7] Grudnitski, G., & Osburn, L. (1993). Forecasting S & P and gold futures prices: an application of neural networks. *J. Futures Markets*, 13, 631－643.

[8] Hutchinson, J., Lo, A., & Poggio, T. (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *J. Finance*, 49 (3), 851－889.

[9] Me'zard, M., & Nadal, J. (1989). Learning in Feedforward Layered Networks: The Tiling Algorithm. *Journal of Physics A*, 22, 2191－2204.

[10] Rumelhart, D., Hinton, G. & Williams, R. (1986a). Learning Internal Representations By Error Propagation. In Rumelhart, D. & McClelland J. (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1 (pp. 318-362). Cambridge, MA: MIT Press.

[11] Rumelhart, D., Hinton, G., & Williams, R. (1986b). Learning Internal Representations By Error Propagation. *In Rumelhart, D. & McClelland J. (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1 (pp. 335-337). Cambridge, MA: MIT Press.

[12] Tsaih, R. (1993). The Softening Learning Procedure. *Mathematical and Computer Modelling*, Vol. 18, No. 8, 61－64.

[13] Tsaih, R. (1997). The Reasoning Neural Networks. In Ellacott S., J. Mason & I. Anderson (Eds.), *Mathematics of Neural Networks: Models, Algorithms and Applications*, (pp. 366-371). Kluwer Academic publishers, London.

[14] Tsaih, R. (1998). An Explanation of Reasoning Neural Networks. *Mathematical and Computer Modelling*, Vol. 28, No. 2, 37－44.

[15] Tsaih, R., Hsu, **Y**., & Lai, C. (1998). Forecasting S&P 500 Stock Index Futures with the Hybrid AI system. *Decision Support Systems*, Vol. 23, No. 2, 161－174.

[16] Watanabe, E., & Shimizu, H. (1993). Algorithm for Pruning Hidden nodes in Multi-Layered Neural Network for Binary Pattern Classification Problem, in Proc. *International Joint Conference on Neural Networks*, I, 1993, 327－330.

[17] Yao, X. (1993). A Review of Evolutionary Artificial Neural Networks. *International Journal of Intelligent Systems*, 8 (4), 539－567.