PERGAMON Mathematical and Computer Modelling 38 (2003) 339–350

MATHEMATICAL AND COMPUTER MODELLING

www.elsevier.com/locate/mcm

The Evolution of Internal Representation

RAY TSAIH
Department of Management Information Systems, National Chengchi University
No 64, Chih-nan Road, Sect. 2, Wenshan 11623. Taipei, R.O.C.
tsaih@mis.nccu.edu.tw

(Received September 2002; revised and accepted February 2003)

Abstract—To develop an appropriate internal representation, a deterministic learning algorithm that can 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 are used to demonstrate the performance of the proposed algorithm. The experimental results are clearly positive.

Keywords—Internal representation, Recruiting mechanism, Pruning mechanism, Generalized delta rule.

1. INTRODUCTION

In modern finance, derivatives such as futures and options play increasingly prominent roles in risk management and price speculation. Owing to the high leverage 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. Generally, predicting the financial market's movements is considered more difficult than expected. 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, [1–3]). To make the forecasting of futures prices more reliable, the application of artificial neural networks (ANN), especially the layered feed-forward network [4], has received extensive attentions [3,5,6].

Instead of directly deriving the nonlinear equation, the layered feed-forward network tries to develop an appropriate internal representation for such forecasting problem. In general, a nonlinear forecasting problem is like the problem of finding a nonlinear equation to capture the general

Research supported by National Science Council of R.O.C. under Grants No. NSC 91-2416-H-004-009. Thanks are also due to W. Ke of National Chengchi University and two anonymous reviewers for useful comments on an earlier draft of this paper.

0895-7177/03/$ - see front matter Typeset by AMS-TEX
doi: 10.1016/S0895-7177(03)00225-5

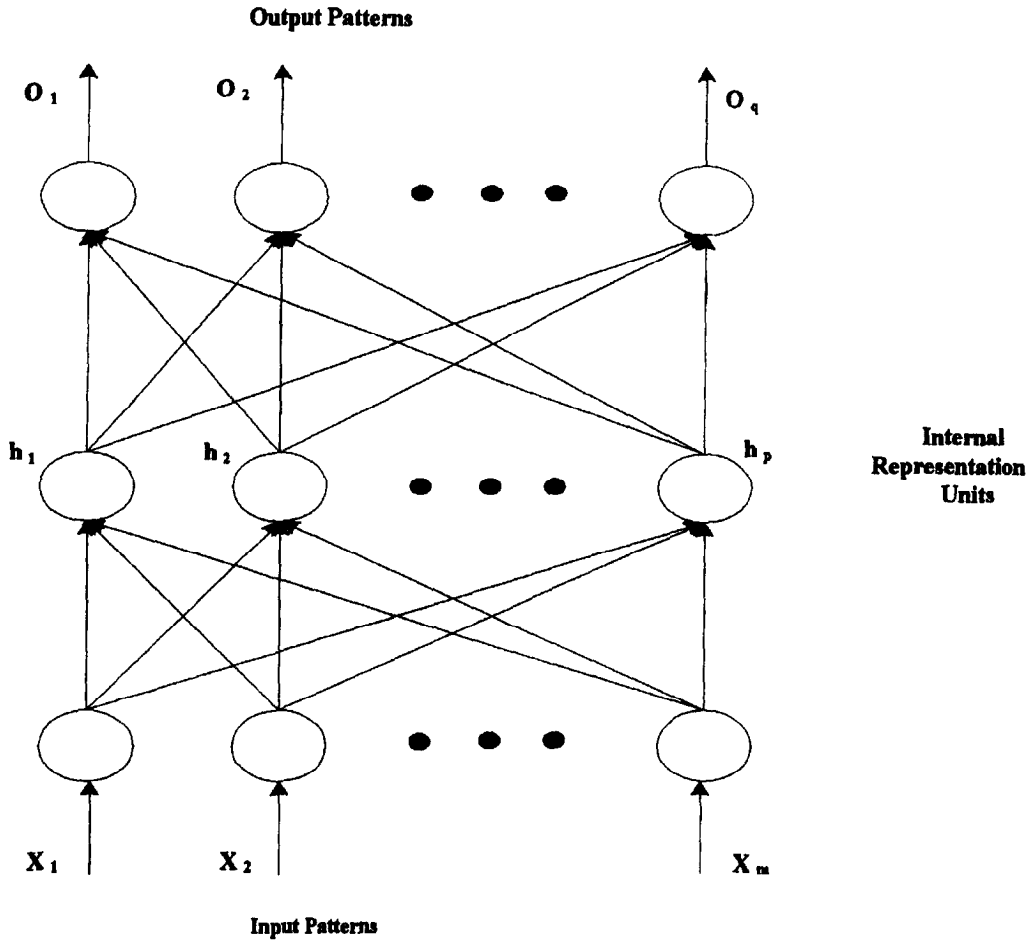

Figure 1. An ANN with a layer of hidden nodes.

pattern of a relationship between the independent variables x_js and the dependent variables y_ls. 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 ${}_ct_l$ the observed value of y_l corresponding to ${}_c\mathbf{x}$. In the context of a layered feed-forward network as shown in Figure 1, the information $\mathbf{x}$ coming to the input nodes is recoded into an internal representation $\mathbf{h} \equiv (h_1, h_2, \ldots, h_p)^\top$, 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 [4, p. 320].

Let us take as an illustration a layered feed-forward network with the hyperbolic tangent (tanh) function [4,7], where $\tanh(x) \equiv (e^x - e^{-x})/(e^x + e^{-x})$, used in all output and hidden nodes. 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:

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

$$\begin{aligned} O({}_c\mathbf{x}, {}_3\mathbf{w}_l, {}_2\mathbf{w}) &\equiv \tanh\left({}_3w_{l0} + \sum_{i=1}^{p} {}_3w_{li}\, h({}_c\mathbf{x}, {}_2\mathbf{w}_i)\right) \\ &= \tanh\left({}_3w_{l0} + \sum_{i=1}^{p} {}_3w_{li} \tanh\left({}_2w_{i0} + \sum_{j=1}^{m} {}_2w_{ij}\,{}_cx_j\right)\right), \end{aligned} \tag{2}$$

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. Character in bold represents a column vector and the superscript $\top$ indicates the transposition: ${}_2\mathbf{w}_i^\top \equiv ({}_2w_{i0}, {}_2w_{i1}, \ldots, {}_2w_{im})$, ${}_2\mathbf{w}^\top \equiv ({}_2\mathbf{w}_1^\top, {}_2\mathbf{w}_2^\top, \ldots, {}_2\mathbf{w}_p^\top)$, ${}_3\mathbf{w}_l^\top \equiv ({}_3w_{l0}, {}_3w_{l1}, \ldots, {}_3w_{lp})$, ${}_3\mathbf{w}^\top \equiv ({}_3\mathbf{w}_1^\top, {}_3\mathbf{w}_2^\top, \ldots, {}_3\mathbf{w}_q^\top)$, and $\mathbf{w}^\top \equiv ({}_2\mathbf{w}^\top, {}_3\mathbf{w}^\top)$.

Given a set of training samples $\{({}_1\mathbf{x}, {}_1\mathbf{t}), \ldots, ({}_N\mathbf{x}, {}_N\mathbf{t})\}$, the goal of learning 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, say 10^{-6}. In general, the learning can be recognized as a minimization of the sum of residual squares $E(\mathbf{w}, p)$, set thus,

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

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 a globally optimal solution $(\mathbf{w}*, p*)$ of problem (3), many researchers are motivated to develop a reasonable algorithm to explore 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 an algorithm is not easy due to the necessity of coping 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 nondifferentiable since, for example, changes in the value of p are discrete and can have discontinuous effects on the value of $E(\mathbf{w}, p)$;
(3) the surface is nonanalyzable 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. Due to these characteristics, there is hardly any solid theoretical support for developing a reasonable learning algorithm.

In the context of internal representation, the activation functions adopted for all hidden nodes are predefined and fixed. Thus, the internal presentation evolves when the values of p and ${}_2\mathbf{w}$ are altered.

Because of the linear nature of computing the net input value and the semilinear characteristic of the activation function adopted in equations (1) and (2), there is a level-adjacent mapping from the $\{\mathbf{x}\}$ space to the $\{\mathbf{h}\}$ space, and from the $\{\mathbf{h}\}$ space to the output layer $\{\mathbf{O}\}$ space. Level-adjacent mapping means that level-adjacent points in the previous-layer space are mapped to neighboring points in the latter-layer space [7]. While the proximity of two points in the latter-layer space is measured by their (direct) distance, the level-adjacency between two points in the previous-layer space is measured by the difference between their associated activation levels.

In the learning stage, the positions 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 [7], no matter what kind of learning algorithm is used, the resulting mapping between two consecutive layers is always a level-adjacent mapping. Furthermore, 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. Currently, however, the development of the internal representation is related to the current values of p and $\mathbf{w}$.

In the literature, there are two categories of learning algorithms for layered feed-forward networks: evolutionary ANN (EANN) algorithms, which are stochastic, and 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 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 distinguished styles of evolutionary algorithms:

- evolutionary strategies (ES),
- evolutionary programming (EP),
- genetic algorithms (GA), and
- genetic programming (GP).

All of them model the search process over the solution space by mimicking biological evolution. They differ mainly in the evolution operators involved and the representation of the solution space. Most researchers believe that 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 [8].

In contrast, all weight-and-structure-change learning algorithms adjust $\mathbf{w}$ and p in a deterministic way; for example, the tiling algorithm [9], the cascade-correlation (CC) algorithm [10], the upstart algorithm [11], the W&S algorithm [12], the CTN algorithm [13], and the softening algorithm [7,14,15]. 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 the ability to build up an appropriate internal representation via recruiting/pruning hidden nodes and altering the associated weights during the learning process.

Here, I introduce a deterministic learning algorithm that makes use of sequentially presented training samples to adjust the values of $\mathbf{w}$ and p to develop an internal representation appropriate for the required mapping. Moreover, this learning algorithm guarantees an acceptable learning result. 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.

The remainder of 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 presented in [16] will be used here 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

Without losing generality, 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 listed below.

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 $\lambda = 1$.
      (b) Restore the weights.
      (c) $p + 2 \rightarrow p$ and recruit two extra hidden nodes with ${}_2\mathbf{w}_{p-1}{}^{\top} = (\zeta - \lambda\boldsymbol{\alpha}^{\top}{}_k\mathbf{x}, \lambda\boldsymbol{\alpha}^{\top})$, ${}_2\mathbf{w}_p{}^{\top} = (\zeta + \lambda\boldsymbol{\alpha}^{\top}{}_k\mathbf{x}, -\lambda\boldsymbol{\alpha}^{\top})$, $\zeta = 10^{-6} \min_{c \in I(k-1)} |\boldsymbol{\alpha}^{\top}({}_k\mathbf{x} - {}_c\mathbf{x})|$, ${}_3w_{p-1} = {}_3w_p = \left(\tanh^{-1}({}_kt) - {}_3w_0 - \sum_{i=1}^{p-2} {}_3w_i h({}_k\mathbf{x}, {}_2\mathbf{w}_i)\right)$ $/2\tanh(\zeta)$, where the length of vector $\boldsymbol{\alpha}$ is one and $\boldsymbol{\alpha}^{\top}({}_k\mathbf{x} - {}_c\mathbf{x}) \neq 0$, $\forall c \in I(k-1)$.
      (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 $\lambda * 2 \rightarrow \lambda$ and $p - 2 \rightarrow p$, then go to (b).
Step 4: Prune all potentially irrelevant hidden nodes.
Step 5: $k + 1 \rightarrow k$; go to Step 1.

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

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

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

Note that accomplishing goal (4) is a sufficient, but not a necessary condition to obtain an appropriate internal representation. However, the effort of regularly checking whether of not the current internal representation is more complicated than the one of regularly checking the current internal representation is more complicated than the one of regularly checking if goal (4) is currently accomplished. This argument accounts for adopting goal (4) in the arrangement of this learning algorithm.

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

Specifically, when the k^{th} sample $({}_k\mathbf{x}, {}_kt)$ is processed, we first check if goal (4) is accomplished. If so, there is only a reasoning effort involved. Then the next given sample is processed. If not, in our next step (Step 3), the weight-tuning mechanism implementing the momentum version of the generalized delta rule [4] with automatic adjustment of the learning rate [15] is applied to $\min_{\mathbf{w}} E_k(\mathbf{w})$ to adjust weights, where $E_k(\mathbf{w}) \equiv \sum_{c \in I(k)} (\tanh({}_3w_0 + \sum_{i=1}^{p} {}_3w_i \tanh({}_2w_{i0} +$

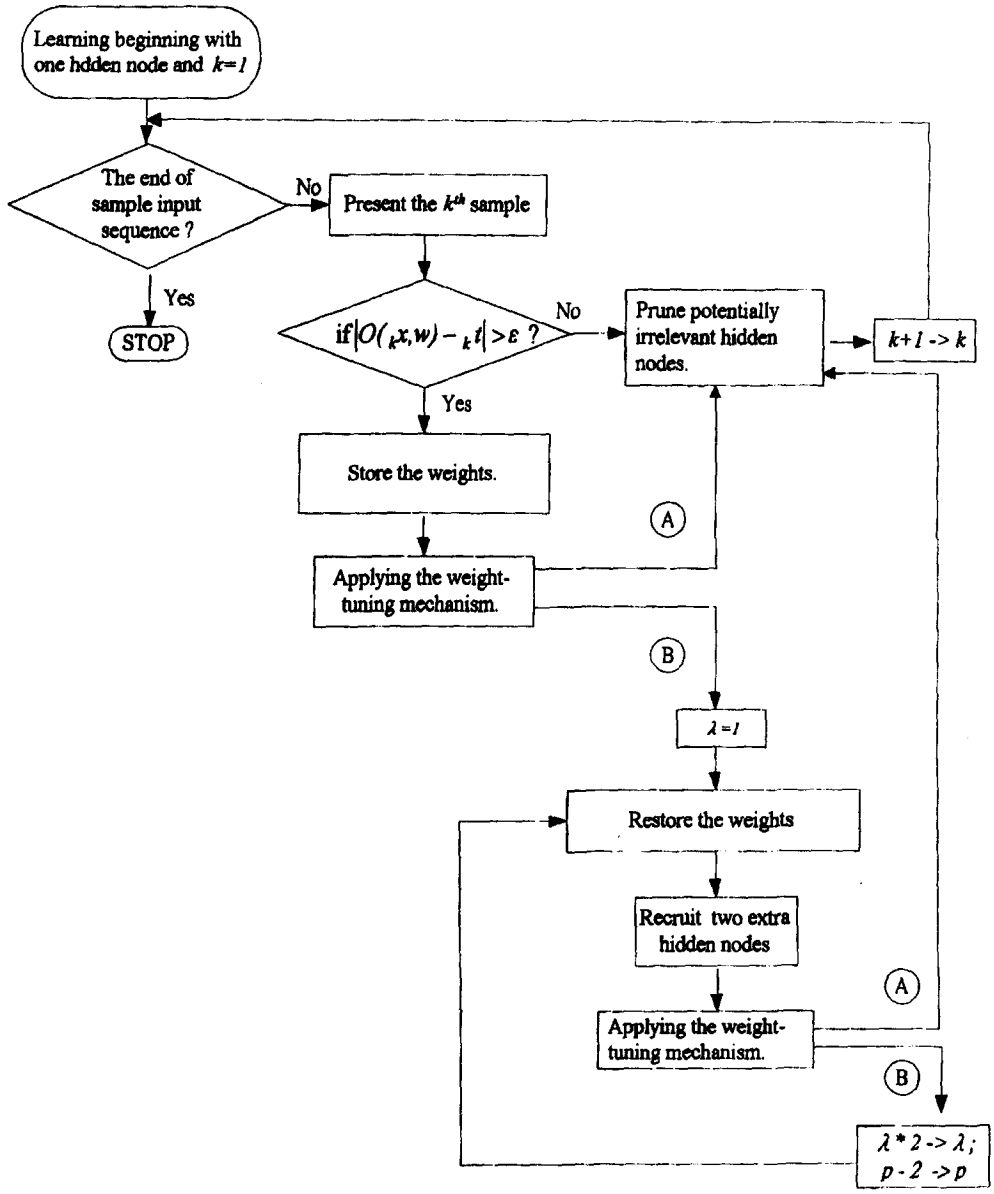

Figure 2. The flow chart of the proposed learning algorithm.

$\sum_{j=1}^{m} {}_2 w_{ijc} x_j)) - {}_c t)^2$. 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 parameter p remains the same. Such a weight-tuning mechanism attempts to achieve goal (4).

Rumelhart *et al.* in [4] have argued 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 or delete hidden nodes. Moreover, this mechanism may converge on the neighborhood of an undesired attractor of $\min_{\mathbf{w}} E_k(\mathbf{w})$ in which $\nabla_w E_k(\mathbf{w}) = \mathbf{0}$; 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 these situations lead to an unacceptable result. These are indicated in Figure 3. Path B indicates the situation when the result of implementing the weight-tuning mechanism is an unacceptable one.

A perfect weight-tuning mechanism that can avoid the predicament of converging on an undesired attractor is desirable, because the defective network structure will be the only cause of an unacceptable result. Unfortunately, there is currently no such perfect weight-tuning mechanism.

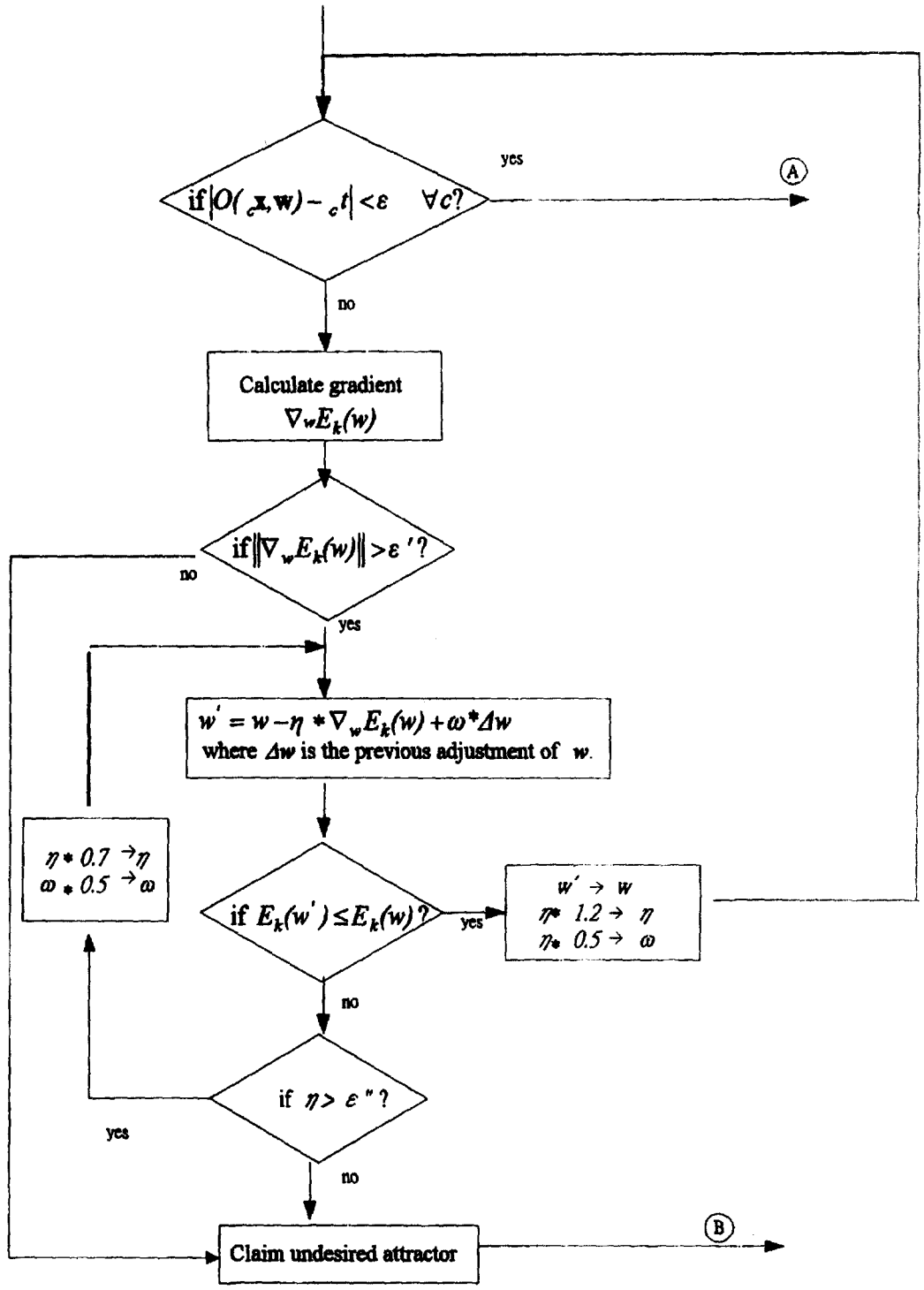

Figure 3. The flow chart of the weight-tuning mechanism implementing the momentum version of the generalized delta rule with automatic adjustment of the learning rate. ε' and ε'' are given tiny numbers.

Under this constraint as well as with the consideration of computing complexity, the current weight-tuning mechanism is adopted. The consideration of computing complexity is important because the weight-tuning mechanism will be triggered frequently during the learning process.

Extra hidden nodes are recruited to handle the predicament in Path B of Figure 3 in the following manner. 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 $|O({}_k\mathbf{x}, \mathbf{w}) - {}_kt| \geq \varepsilon$. For Action (c) in Step 3.2, there is a mechanism arranged to recruit two extra hidden nodes with a gain parameter λ whose value is initially one set from Action (a) in Step 3.2. These two newly-added hidden nodes, the $p-1^{\text{th}}$ and p^{th} ones, have weights ${}_2\mathbf{w}_{p-1}^\top = (\zeta - \lambda\boldsymbol{\alpha}^\top {}_k\mathbf{x}, \lambda\boldsymbol{\alpha}^\top)$, ${}_2\mathbf{w}_p^\top = (\zeta + \lambda\boldsymbol{\alpha}^\top {}_k\mathbf{x}, -\lambda\boldsymbol{\alpha}^\top)$, $\zeta = 10^{-6} \min_{c\in I(k-1)} |\boldsymbol{\alpha}^\top({}_k\mathbf{x} - {}_c\mathbf{x})|$, ${}_3w_{p-1} = {}_3w_p = (\tanh^{-1}({}_kt) - {}_3w_0 - \sum_{i=1}^{p-2} {}_3w_i h({}_k\mathbf{x}, {}_2\mathbf{w}_i))/(2\tanh(\zeta))$, where the length of vector $\boldsymbol{\alpha}$ is one and $\boldsymbol{\alpha}^\top({}_k\mathbf{x} - {}_c\mathbf{x}) \neq 0$, $\forall c \in I(k-1)$. For Action (d) in Step 3.2, the λ value is fixed and the weight-tuning mechanism is applied to render goal (4) accomplished. If an unacceptable result is obtained, we multiply the λ value by 2, and repeat Actions (b)–(d). Actions (b)–(d) are repeated until goal (4) is achieved.

Recruiting two extra hidden nodes has introduced two extra dimensions in the $\{\mathbf{h}\}$ space, and has thus, introduced two extra dimensions in the internal representation. The arrangement of ${}_2\mathbf{w}_{p-1}$ and ${}_2\mathbf{w}_p$ has put the new corresponding point $\mathbf{h}({}_k\mathbf{x}, {}_2\mathbf{w})$ on the each positive side of these two newly-added dimensions, and all other new corresponding points $\mathbf{h}({}_c\mathbf{x}, {}_2\mathbf{w})$s on the positive side of one newly-added dimension and on the negative side of the other newly-added dimension. A large λ value makes the behavior of the activation functions of the newly recruited $p-1^{\text{th}}$ and p^{th} hidden nodes similar to the behavior of a threshold function, and results in the 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 two such newly-added hidden nodes with a large λ value has put the new corresponding point $\mathbf{h}({}_k\mathbf{x}, {}_2\mathbf{w})$ near the $(\tanh(\zeta), \tanh(\zeta))$ position of these two newly-added dimensions, and all other new corresponding points $\mathbf{h}({}_c\mathbf{x}, {}_2\mathbf{w})$s near the $(-1, 1)$ corner of these two newly-added dimensions. Therefore, as mentioned in Lemma 2, if the internal representation renders $|O({}_c\mathbf{x}, \mathbf{w}) - {}_ct| < \varepsilon$, for all $c \in I(k-1)$ and $|O({}_k\mathbf{x}, \mathbf{w}) - {}_kt| \geq \varepsilon$, the new internal representation can be appropriate for all k training samples after such two hidden nodes are recruited.

In fact, Lemma 2 reveals that goal (4) can be accomplished immediately merely by recruiting extra hidden nodes with proper weights and λ, and that only two of these extra hidden nodes are needed. Moreover, there is no infinite loop in Step 3.2. Therefore, Step 3.2 ensures that the goal of each stage is achieved at the end of each stage, thus, guaranteeing an acceptable learning result in the end.

The recruiting mechanism handles the occurrence of an unacceptable result without involving the reason. A defective network structure triggers the recruiting mechanism; convergence on an undesired attractor also triggers the recruiting mechanism. The triggering of the recruiting mechanism due to the convergence on an undesired attractor may recruit excess hidden nodes that become irrelevant later. At each stage, a hidden node is irrelevant if goal (4) is still accomplished with this hidden node deleted. The irrelevant hidden nodes are useless with respect to goal (4); furthermore, they may contribute significant effort to the performance of the network and result in poor generalization. In addition, more samples typically produce more concise information about the appropriate internal representation, and thus, fewer hidden nodes are required. It is accordingly necessary to prune irrelevant hidden nodes, and an internal representation is better if it reaches goal (4) with a smaller amount of adopted hidden nodes.

In Step 4, a reasoning mechanism is arranged to prune all potentially irrelevant hidden nodes. At the k^{th} stage, a hidden node is potentially irrelevant if it is deleted and goal (4) can be accomplished by 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, I use two popular examples to examine how the current arrangements for the recruiting

and reasoning mechanisms work: the encoding problem [16,17] and the parity problem [16].

Ackley *et al.* in [17] has posed the encoding problem where a set of N orthogonal input patterns is mapped to a set of N orthogonal output patterns through a small set of hidden nodes. Such a 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. Rumelhart *et al.* in [16] has proposed that a set of N orthogonal input patterns is mapped to a set of N orthogonal output patterns through a small set of $\log_2 N$ hidden nodes. The reason behind such a design is that if the hidden nodes take on binary values, the hidden nodes must form a binary number to encode each input pattern. The authors of [16] present an encoding problem with eight input patterns, eight output patterns, and three hidden nodes, and find the learning system develops solutions that use the intermediate values, as shown in Table 2.

The result of simulation is encouraging: the proposed algorithm employs the intermediate values in a more efficient way. Table 3 shows the mapping generated by the proposed algorithm. The proposed algorithm also utilizes intermediate values to gain an appropriate internal representation. The appropriate internal representations shown in Tables 2 and 3 are similar, except that the former uses three hidden nodes and the latter 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 2. The mapping of the encoding problem generated in [16].

Input Pattern		Hidden Node Pattern				Output Pattern
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 Pattern		Hidden Node Pattern			Output Pattern
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

Table 4 shows the mapping of the four-bit parity problem obtained in [16] and generated by the proposed algorithm. Rumelhart *et al.* in [16] 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", see [16, p. 335] By contrast, as shown in Table 4, the appropriate internal representations developed in [16] and by the proposed algorithm look similar, except that the former uses four hidden nodes and the latter uses three hidden nodes.

Table 4. The mapping of the four-bit parity problem generated in [16] and by the proposed algorithm.

Input Pattern		Hidden Node Pattern Generated in [16]	Hidden Node Pattern Generated by the Proposed Algorithm				Output Pattern
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

In sum, it seems that the current arrangement of the recruiting and reasoning mechanisms in the proposed algorithm works so that the internal representation evolves in a better way than the representation developed by the generalized delta rule, a result that is based upon the reasoning that an appropriate internal representation with a 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 into an appropriate one. The key mechanisms of the proposed algorithm are the recruiting mechanism and the reasoning mechanism. I provide theoretical justification to explain why the proposed algorithm guarantees an acceptable learning result.

The experimental findings demonstrate that the proposed algorithm also employs an ability to develop an internal representation similar with the one shown in [16]. This is not surprising since the proposed algorithm also adopts in its weight-tuning mechanism the generalized delta rule proposed in [4]. However, the experimental results indicate that the current arrangements of the recruiting and reasoning mechanisms in the proposed algorithm work quite well that it enables the internal representation to evolve in a superior way than the representation developed by the generalized delta rule.

There is a one caveat: when we apply the ANN to practical problems, a black box is obtained from the learning. Further study should attempt to explore the appropriate internal representation obtained from the learning in order to provide the knowledge behind the application, and thus, remove this caveat.

The results of simulation 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 converging on an undesired attractor. Further investigation (theoretical and numerical) on the surface defined by $E_k(\mathbf{w})$ over

the weight space is currently under way to identify the scenario (the sample input sequence and the value of λ for which the weight-tuning mechanism will (or will not) achieve goal (4). Another study involves exploring a more efficient recruiting mechanism to recruit hidden nodes and a more efficient reasoning mechanism to prune potentially irrelevant hidden nodes.

APPENDIX

THEORETICAL SUPPORTS AND THEIR PROOFS

LEMMA 1. *Let* $\{{}_c\mathbf{x} \mid \forall c \in I(k)\}$ *be given, and assume* ${}_2\mathbf{w}_{p-1}^\top = (\zeta - \lambda\boldsymbol{\alpha}^\top {}_k\mathbf{x}, \lambda\boldsymbol{\alpha}^\top), {}_2\mathbf{w}_p^\top = (\zeta + \lambda\boldsymbol{\alpha}^\top {}_k\mathbf{x}, -\lambda\boldsymbol{\alpha}^\top)$, *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 ${}_2\mathbf{w}_{p-1}{}^\top = (\zeta - \lambda\boldsymbol{\alpha}^\top {}_k\mathbf{x}, \lambda\boldsymbol{\alpha}^\top)$, ${}_2w_{p-10} + \sum_{j=1}^m {}_2w_{p-1j}\, {}_cx_j = \zeta + \lambda(\boldsymbol{\alpha}^\top {}_c\mathbf{x} - \boldsymbol{\alpha}^\top {}_k\mathbf{x})$. Similarly, ${}_2w_{p0} + \sum_{j=1}^m {}_2w_{pj}\, {}_cx_j = \zeta + \lambda(\boldsymbol{\alpha}^\top {}_k\mathbf{x} - \boldsymbol{\alpha}^\top {}_c\mathbf{x})$. $\{{}_c\mathbf{x} \mid \forall c \in I(k)\}$ is given, so ${}_k\mathbf{x} - {}_c\mathbf{x}$ 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 vector $\boldsymbol{\alpha}$ that the length of $\boldsymbol{\alpha}$ is one and $\boldsymbol{\alpha}^\top({}_k\mathbf{x} - {}_c\mathbf{x}) \neq 0$, $\forall c \in I(k-1)$. Then, ζ is assigned as $10^{-6} \min_{c \in I(k-1)} |\boldsymbol{\alpha}^\top({}_k\mathbf{x} - {}_c\mathbf{x})|$.

Because λ is a large value and $\zeta = 10^{-6} \min_{c \in I(k-1)} |\boldsymbol{\alpha}^\top({}_k\mathbf{x} - {}_c\mathbf{x})|$, $h({}_c\mathbf{x}, {}_2\mathbf{w}_{p-1}) = \tanh(\zeta + \lambda(\boldsymbol{\alpha}^\top {}_c\mathbf{x} - \boldsymbol{\alpha}^\top {}_k\mathbf{x})) \cong \tanh(\lambda(\boldsymbol{\alpha}^\top {}_c\mathbf{x} - \boldsymbol{\alpha}^\top {}_k\mathbf{x}))$ and $h({}_c\mathbf{x}, {}_2\mathbf{w}_p) = \tanh(\zeta + \lambda(\boldsymbol{\alpha}^\top {}_k\mathbf{x} - \boldsymbol{\alpha}^\top {}_c\mathbf{x})) \cong -\tanh(\lambda(\boldsymbol{\alpha}^\top {}_c\mathbf{x} - \boldsymbol{\alpha}^\top {}_k\mathbf{x}))$. Thus, $h({}_c\mathbf{x}, {}_2\mathbf{w}_{p-1}) + h({}_c\mathbf{x}, {}_2\mathbf{w}_p) \cong 0.0$, $\forall c \in I(k-1)$.

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

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* $|O({}_k\mathbf{x}, \mathbf{w}) - {}_kt| \geq \varepsilon$. *When the following two hidden nodes are recruited,* $h({}_c\mathbf{x}, {}_2\mathbf{w}_{p-1}) \equiv \tanh({}_2w_{p-10} + \sum_{j=1}^m {}_2w_{p-1j}\, {}_cx_j$, *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 the introduction of two hidden nodes, respectively. Also let ${}_c$net be the value of ${}_3w_0 + \sum_{i=1}^{p-2} {}_3w_i h({}_c\mathbf{x}, {}_2\mathbf{w}_i)$ before the introduction of two hidden nodes. Thus, ${}_cy' = \tanh({}_c\text{net})$ and ${}_cy = \tanh({}_c\text{net} + {}_3w_{p-1} h({}_c\mathbf{x}, {}_2\mathbf{w}_{p-1}) + {}_3w_p h({}_c\mathbf{x}, {}_2\mathbf{w}_p))$.

Because $|O({}_c\mathbf{x}, \mathbf{w}) - {}_ct| < \varepsilon$, $\forall c \in I(k-1)$ and $|O({}_k\mathbf{x}, \mathbf{w}) - {}_kt| \geq \varepsilon$ before introducing two hidden nodes, $|{}_cy' - {}_ct| < \varepsilon$, $\forall c \in I(k-1)$ and $|{}_ky' - {}_kt| \geq \varepsilon$.

Let $\lambda, {}_2\mathbf{w}_{p-1}$ 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 $\gamma = (\tanh^{-1}({}_kt) - {}_k\text{net})/(2\tanh(\zeta))$.

Since ${}_3w_{p-1} = {}_3w_p$ and $h({}_c\mathbf{x}, {}_2\mathbf{w}_{p-1}) + h({}_c\mathbf{x}, {}_2\mathbf{w}_p) \cong 0.0$, $\forall c \in I(k-1)$, ${}_cy \cong {}_cy'$, $\forall c \in I(k-1)$. Therefore, $|{}_cy - {}_ct| < \varepsilon$, $\forall c \in I(k-1)$ since $|{}_cy' - {}_ct| < \varepsilon$, $\forall c \in I(k-1)$.

$h({}_k\mathbf{x}, {}_2\mathbf{w}_{p-1}) + h({}_k\mathbf{x}, {}_2\mathbf{w}_p) = 2\tanh(\zeta)$, so ${}_ky = \tanh({}_k\text{net} + \gamma 2\tanh(\zeta)) = {}_kt$. Thus, $|{}_ky - {}_kt| < \varepsilon$. ∎

REFERENCES

1. S. Blank, Chaos in futures market? A nonlinear dynamical analysis, *J. Futures Markets* **11**, 711–728, (1991).
2. G. DeCoster, W. Labys and D. Mitchell, Evidence of chaos in commodity futures prices, *J. Futures Markets* **12**, 291–305, (1992).
3. G. Grudnitski and L. Osburn, Forecasting S & P and gold futures prices: An application of neural networks, *J. Futures Markets* **13**, 631–643, (1993).
4. D. Rumelhart, G. Hinton and R. Williams, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1*, (Edited by D. Rumelhart and J. McClelland), pp. 318–362, MIT Press, Cambridge, MA, (1986).
5. J. Hutchinson, A. Lo and T. Poggio, A nonparametric approach to pricing and hedging derivative securities via learning networks, *J. Finance* **49** (3), 851–889, (1994).

6. R. Tsaih, Y. Hsu and C. Lai, Forecasting S&P 500 stock index futures with the Hybrid AI system, *Decision Support Systems* **23** (2), 161–174, (1998).
7. R. Tsaih, An explanation of reasoning neural networks, *Mathl. Comput. Modelling* **28** (2), 37–44, (1998).
8. X. Yao, A review of evolutionary artificial neural networks, *International Journal of Intelligent Systems* **8** (4), 539–567, (1993).
9. M. Me'zard and J. Nadal, Learning in feedforward layered networks: The tiling algorithm, *Journal of Physics A* **22**, 2191–2204, (1989).
10. S. Fahlman and C. Lebiere, The cascade-correlation learning architecture, In *Advances in Neural Information Processing Systems II*, (Edited by D. Touretzky), Morgan Kaufmann, San Mateo, CA, (1990).
11. M. Frean, The upstart algorithm: A method for constructing and training feedforward neural networks, *Neural Computation* **2**, 198–209, (1990).
12. E. Watanabe and H. Shimizu, Algorithm for pruning hidden nodes in multi-layered neural network for binary pattern classification problem, *Proc. International Joint Conference on Neural Networks I*, pp. 327–330, (1993).
13. Y. Chen, D. Thomas and M. Nixon, Generating-shrinking algorithm for learning arbitrary classification, *Neural Networks* **7**, 1477–1489, (1994).
14. R. Tsaih, The softening learning procedure, *Mathl. Comput. Modelling* **18** (8), 61–64, (1993).
15. R. Tsaih, The reasoning neural networks, In *Mathematics of Neural Networks: Models, Algorithms and Applications*, (Edited by S. Ellacott, J. Mason and I. Anderson), pp. 366–371, Kluwer Academic, London, (1997).
16. D. Rumelhart, G. Hinton and R. Williams, Learning internal representations by error propagation, In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1*, (Edited by D. Rumelhart and J. McClelland), pp. 335–337, MIT Press, Cambridge, MA, (1986).
17. D. Ackley, G. Hinton and T. Sejnowski, A learning algorithm for Boltzmann machines, *Cognitive Science* **9**, 147–169, (1985).