Pergamon

Mathl. Comput. Modelling Vol. 28, No. 2, pp. 37–44, 1998

Printed in Great Britain
0895-7177/98 $19.00 + 0.00

PII: S0895-7177(98)00090-9

An Explanation of Reasoning Neural Networks

R. R. TSAIH
Department of Management Information Systems
National Chengchi University, Taipei, Taiwan, R.O.C.
tsaih@mis.nccu.edu.tw

(Received July 1996; accepted October 1997)

Abstract—Reasoning Neural Networks (RN) adopts the layered feedforward network structure, and its learning algorithm belongs to the weight-and-structure-change category of learning algorithm. In this paper, we firstly explain that, in the layered feedforward network, the essential characteristic of the mapping between two consecutive layers is the level-adjacent mapping, in which level-adjacent patterns in the previous-layer space are mapped to similar patterns in the latter-layer space. Then, we explain how RN's learning algorithm handles the undesired predicaments associated with the back propagation learning algorithm. © 1998 Elsevier Science Ltd. All rights reserved.

Keywords—Reasoning neural networks, Activation field, Properly placed, Level-adjacent mapping.

1. INTRODUCTION

In [1–3][1], Reasoning Neural Networks (RN) is presented. Its learning algorithm belongs to the weight-and-structure-change category of learning algorithm, because it puts only one hidden node initially, and will recruit and prune hidden nodes during the learning process.

In literature, there are several such learning algorithms; for example, the W&S algorithm [4], the CTN algorithm [5], the tiling algorithm [6], the cascade-correlation (CC) algorithm [7], and the upstart algorithm [8]. They adjust the network structure in one of the following ways:

(1) putting excess hidden nodes initially and pruning least effective hidden nodes during the learning process; e.g., W&S algorithm and CTN algorithm;
(2) putting less hidden nodes initially and recruiting more hidden nodes during the learning process; e.g., the tiling algorithm, CC algorithm, and the upstart algorithm; and
(3) putting only one hidden node initially, and autonomously recruiting and pruning hidden nodes during the learning process; e.g., RN algorithm.

In addition, there are other distinguished features in RN's learning algorithm. For example, while the tiling algorithm and the upstart algorithm use the threshold activation function, RN uses the semilinear activation function. While the tiling algorithm, the upstart algorithm, and the CC algorithm adopt the perceptron learning rule or its variant, RN adopts the generalized delta rule. Although learning is not guaranteed to be accomplished perfectly in the CC algorithm and

[1]Those RNs are merely capable of handling the 2-classes categorization learning problem, where all of the output values are 1 or −1. The RN developed in [1] can handle the case of one output node and the input values being binary number; the one in [2] is capable of handling the case of multiple output nodes and the input values being binary number; and the one in [3] has the ability of handling the case of multiple output nodes and the input values being real number.

the W&S method, RN's learning algorithm does guarantee a perfect learning. While the W&S method proposes a new cost function for the purpose of pruning, RN adopts the original cost function of the BP learning algorithm.

The empirical results in [9–11] do show that RN outperforms BP. In this paper, we explain how RN operates and learns.

2. THE OPERATION OF THE LAYERED FEEDFORWARD NETWORKS

RN adopts the layered feedforward network structure. Let us suppose that all hidden and output nodes adopt the tanh activation function. The notations of all variables and parameters used in this paper are given in Table 1.

Table 1. The notations of all variables and parameters. The bold character indicates a vector or a matrix, and the superscript $\top$ indicates the transposition.

Parameter or Variable	Description
m, p, and q	the amount of input nodes, hidden nodes, and output nodes, respectively
$\mathbf{B}$	$\mathbf{B} \equiv (b_1, b_2, \ldots, b_m)^\top$ is a vector in the input space
$\mathbf{B}_c$	$\mathbf{B}_c \in R^m$ is the c^{th} training stimulus, and b_{cj} is the stimulus value received in the j^{th} input node when $\mathbf{B}_c$ is presented to the network
w_{ij}	the weight of the connection between the j^{th} input node and the i^{th} hidden node
θ_i	the negative of the threshold value of the i^{th} hidden node
$\mathbf{X}$	$\mathbf{X}^\top \equiv (\mathbf{X}_1^\top, \mathbf{X}_2^\top, \ldots, \mathbf{X}_p^\top)$, where $\mathbf{X}_i^\top \equiv (\theta_i, \mathbf{w}_i^\top)$ and $\mathbf{w}_i \equiv (w_{i1}, w_{i2}, \ldots, w_{im})^\top$ is the vector of weights of the connections between all input nodes and the i^{th} hidden node
$h(\mathbf{B}_c, \mathbf{X}_i)$	$h(\mathbf{B}_c, \mathbf{X}_i) \equiv \tanh(\theta_i + \sum_{j=1}^m w_{ij} b_{cj})$ is the activation value of the i^{th} hidden node when $\mathbf{B}_c$ is presented to the network
$\mathbf{h}(\mathbf{B}_c, \mathbf{X})$	the activation value vector of hidden nodes when $\mathbf{B}_c$ is presented to the network
$\mathbf{h}$	$\mathbf{h} \equiv (h_1, h_2, \ldots, h_p)^\top$ is a vector in the hidden-layer space with $h_i \in [-1, 1]\ \forall i$
r_{li}	the weight of the connection between the i^{th} hidden node and the l^{th} output node
s_l	the negative of the threshold value of the l^{th} output node
$\mathbf{Y}$	$\mathbf{Y}^\top \equiv (\mathbf{Y}_1^\top, \mathbf{Y}_2^\top, \ldots, \mathbf{Y}_q^\top)$, where $\mathbf{Y}_l^\top \equiv (s_l, \mathbf{r}_l^\top)$ and $r_l \equiv (r_{l1}, r_{l2}, \ldots, r_{lp})^\top$ is the vector of weights of the connections between all hidden nodes and the l^{th} output node
$\mathbf{Z}$	$\mathbf{Z}^\top \equiv (\mathbf{Y}^\top, \mathbf{X}^\top)$
$O(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X})$	$O(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X}) \equiv \tanh(s_l + \sum_{i=1}^p r_{li} h(\mathbf{B}_c, \mathbf{X}_i))$ is the activation value of the l^{th} output node when $\mathbf{B}_c$ is presented to the network
$\mathbf{O}(\mathbf{B}_c, \mathbf{Z})$	$\mathbf{O}(\mathbf{B}_c, \mathbf{Z}) \equiv (O(\mathbf{B}_c, \mathbf{Y}_1, \mathbf{X}), O(\mathbf{B}_c, \mathbf{Y}_2, \mathbf{X}), \ldots, O(\mathbf{B}_c, \mathbf{Y}_q, \mathbf{X}))^\top$
$\mathbf{d}_c$	$\mathbf{d}_c \in \{-1, 1\}^q$ is the desired output associated with $\mathbf{B}_c$, and d_{cl} is the corresponding desired output value of the l^{th} output node

In the operation stage, $\mathbf{Z}$ is fixed. When the stimulus input $\mathbf{B}$ is input, every $h(\mathbf{B}, \mathbf{X}_i)$ is an image of $\mathbf{B}$ and every $O(\mathbf{B}, \mathbf{Y}_l, \mathbf{X})$ is an image of $\mathbf{h}(\mathbf{B}, \mathbf{X})$. Furthermore, each mapping is what I name as the level-adjacent mapping, in which level-adjacent patterns in the previous-layer space are mapped to similar patterns in the latter-layer space. The detailed explanations are given below.

2.1. The Activation Field

With a nonzero $\mathbf{w}_i$, parallel activation level hyperplanes $\{\theta_i + \sum_{j=1}^m w_{ij} b_j = \tanh^{-1}(a),\ \forall a \in [-1, 1]\}$ are set in the $\{\mathbf{B}\}$ space. Since the threshold and weights are real numbers, these

activation level hyperplanes make a scalar field (the activation field) in the $\{\mathbf{B}\}$ space. Through each point of the input space, there passes one, and only one, activation level hyperplane which determines the associated activation value of that point.

The activation level hyperplanes spread symmetrically and divergingly from the central level hyperplane where $\theta_i + \sum_{j=1}^{m} w_{ij} b_j = 0$; the central level hyperplane goes through the origin of the input space if $\theta_i = 0$. The unit normal vector of the activation field is $\mathbf{w}_i/\|\mathbf{w}_i\|$, and the activation value associated with the activation level hyperplane is monotonically increased along the direction of the normal vector. The distance between two activation level hyperplanes, $\theta_i + \sum_{j=1}^{m} w_{ij} b_j = \tanh^{-1}(a_1)$ and $\theta_i + \sum_{j=1}^{m} w_{ij} b_j = \tanh^{-1}(a_2)$, is $|\tanh^{-1}(a_1) - \tanh^{-1}(a_2)|/\|\mathbf{w}_i\|$; thus, the level activation hyperplanes are distributed more densely at the farther distance from the central level hyperplane. From the shape of the tanh function, we can see all of these are true. Thus, the orientation and distribution of these activation level hyperplanes are relevant with $\mathbf{X}_i$, and the alteration of $\mathbf{X}_i$ will remodel the corresponding activation field.

Similarly, if $\mathbf{r}_l$ is nonzero, parallel output activation level hyperplanes $\{s_l + \sum_{i=1}^{p} r_{li} h_i = \tanh^{-1}(a),\ \forall a \in [-1, 1]\}$ are set in the hidden-layer space ($\{\mathbf{h}\}$). These output activation level hyperplanes also make a scalar activation field. The orientation and distribution of these activation level hyperplanes are relevant with $\mathbf{Y}_l$, and the alteration of $\mathbf{Y}_l$ will remodel the corresponding activation field.

One hidden node sets up an activation field in the input space, and p hidden nodes set up p activation fields in the input space, but these activation fields do not interfere with each other. Also, the number of activation fields set up in the hidden-layer space is as many as the number of the output nodes, and these activation fields do not interfere with each other.

2.2. The Level-Adjacent Mapping

The essential characteristic of the mapping between two consecutive layers is the level-adjacent mapping, in which level-adjacent points in the previous-layer space are mapped to neighboring points in the latter-layer space. In the $\{\mathbf{B}\}$ space, two points, $\mathbf{B}_1$ and $\mathbf{B}_2$, are level-adjacent within δ with respect to the activation field set up by a given $\mathbf{X}_i$ if $|h(\mathbf{B}_1, \mathbf{X}_i) - h(\mathbf{B}_2, \mathbf{X}_i)| \leq \delta$. In the $\{\mathbf{h}\}$ space, two points, $\mathbf{h}(\mathbf{B}_1, \mathbf{X})$ and $\mathbf{h}(\mathbf{B}_2, \mathbf{X})$, are level-adjacent within δ with respect to the activation field set up by a given $\mathbf{Y}_l$ if $|O(\mathbf{B}_1, \mathbf{Y}_l, \mathbf{X}) - O(\mathbf{B}_2, \mathbf{Y}_l, \mathbf{X})| \leq \delta$. This property is due to the linear characteristic of computing the net input value and the semi-linear characteristic of the tanh function. While the nearness of two points in the latter-layer space is measured 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. Take the XOR problem and network A demonstrated in Figure 1 as the illustration. Let $\{\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3, \mathbf{B}_4\} = \{(1,1)^\top, (1,-1)^\top, (-1,-1)^\top, (-1,1)^\top\}$. Since $\mathbf{w}_1 = (-0.1, 0.1)^\top$, $\mathbf{B}_1$ is more level-adjacent to $\mathbf{B}_3$ than $\mathbf{B}_2$ and $\mathbf{B}_4$ with respect to $\mathbf{X}_1$, although the (direct) distance between $\mathbf{B}_1$ and $\mathbf{B}_3$ is bigger. Similarly, $\mathbf{B}_1$ is more level-adjacent to $\mathbf{B}_3$ than $\mathbf{B}_2$ and $\mathbf{B}_4$ with respect to $\mathbf{X}_2$. Therefore, the distance between $\mathbf{h}(\mathbf{B}_1, \mathbf{X})$ and $\mathbf{h}(\mathbf{B}_3, \mathbf{X})$ is smaller than the distance between $\mathbf{h}(\mathbf{B}_1, \mathbf{X})$ and $\mathbf{h}(\mathbf{B}_2, \mathbf{X})$ ($\mathbf{h}(\mathbf{B}_4, \mathbf{X})$), and $\mathbf{h}(\mathbf{B}_1, \mathbf{X})$ is more level-adjacent to $\mathbf{h}(\mathbf{B}_3, \mathbf{X})$ than $\mathbf{h}(\mathbf{B}_2, \mathbf{X})$ and $\mathbf{h}(\mathbf{B}_4, \mathbf{X})$ with respect to $\mathbf{Y}_1$.

No matter what kind of learning algorithm is used, the resulting mapping between two consecutive layers is always a level-adjacent mapping.

3. THE LEARNING PROCESS OF RN

Concerning each output node, the network is used as a classifier which learns to distinguish whether the stimulus is a member of one class of stimuli, called class 1, or of another class, called class 2, by being presented with examples of each class. In the learning stage, a set of training cases $\{(\mathbf{B}_1, \mathbf{d}_1), (\mathbf{B}_2, \mathbf{d}_2), \ldots, (\mathbf{B}_K, \mathbf{d}_K)\}$ is given, and the training stimuli are presented one by

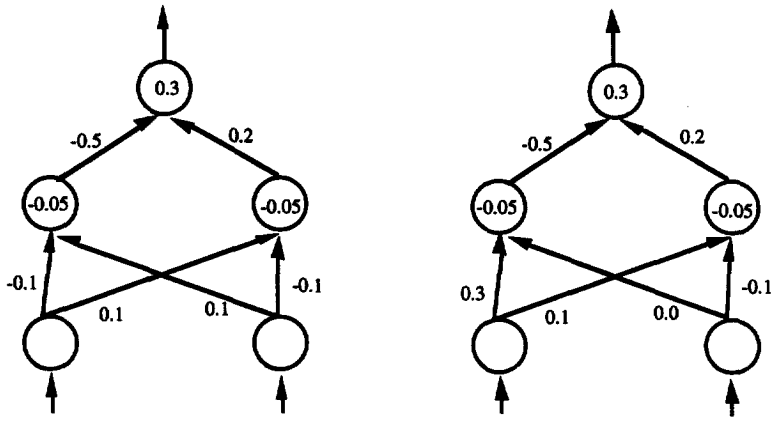

Figure 1. Two feed-forward network systems. The number in the circle is the negative of the threshold value, and the number along each line is the weight value.

one. At presenting the k^{th} given training stimulus, the cost function is defined as:

$$E(\mathbf{Z}) \equiv \sum_{c=1}^{k} \sum_{l=1}^{q} (O(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X}) - d_{cl})^2.$$

Concerning the l^{th} output node, let $K(k) \equiv \{1, \ldots, k\} \equiv K_{l1}(k) \cup K_{l2}(k)$, where $K_{l1}(k)$ and $K_{l2}(k)$ are individually the sets of indices of the first k training stimuli in classes 1 and 2, with the desired output values being 1.0 and -1.0, respectively. At presenting the k^{th} given training stimulus, the goal of learning is to seek a $\mathbf{Z}$ where, for all l,

$$d_{cl} O(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X}) \geq v, \qquad \forall c \in K(k), \tag{1}$$

with $0 < v < 1$.

3.1. The Crux of Learning

The positions of the training stimuli $\{\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_k\}$ in the $\{\mathbf{B}\}$ space are fixed; however, $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_k, \mathbf{X})\}$ are determined by the activation fields set up by $\mathbf{X}$, and each $\mathbf{O}(\mathbf{B}_c, \mathbf{Z})$ is determined with $\mathbf{h}(\mathbf{B}_c, \mathbf{X})$ and the activation fields set up by $\mathbf{Y}$. With respect to the l^{th} output node, $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_k, \mathbf{X})\}$ are properly placed if for any δ, there is a $\mathbf{Y}_l$ such that, for all pairs of (c_1, c_2) with $d_{c_1,l} \neq d_{c_2,l}$, $\mathbf{h}(\mathbf{B}_{c_1}, \mathbf{X})$ and $\mathbf{h}(\mathbf{B}_{c_2}, \mathbf{X})$ are not level-adjacent within δ with respect to the activation field set up by that $\mathbf{Y}_l$. $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_k, \mathbf{X})\}$ are properly placed if $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_k, \mathbf{X})\}$ are properly placed with respect to all output nodes; otherwise, $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_k, \mathbf{X})\}$ are improperly placed. Take the XOR problem and the network systems demonstrated in Figure 1 again as the illustration. Let $\{\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3, \mathbf{B}_4\} = \{(1,1)^\top, (1,-1)^\top, (-1,-1)^\top, (-1,1)^\top\}$. $\mathbf{X}$ of network A renders $\mathbf{h}(\mathbf{B}_1, \mathbf{X})$, $\mathbf{h}(\mathbf{B}_2, \mathbf{X})$, $\mathbf{h}(\mathbf{B}_3, \mathbf{X})$, and $\mathbf{h}(\mathbf{B}_4, \mathbf{X})$ properly placed; while $\mathbf{X}$ of network B does not.

The adjustment of $\mathbf{X}$ will drift these corresponding points $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_k, \mathbf{X})\}$, and the adjustment of $\mathbf{Y}$ will alter the characteristic of the activation fields set up in the $\{\mathbf{h}\}$ space. $\mathbf{X}$ and $\mathbf{Y}$ are usually adjusted simultaneously and iteratively by adopting some deterministic optimization technique which can be the gradient descent method (the generalized delta rule), the conjugate gradient method, or the quasi-Newton method. After every adjustment of $\mathbf{Z}$, $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_k, \mathbf{X})\}$ are either properly placed or improperly placed with respect to every output node. When we have an $\mathbf{X}$ such that $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_k, \mathbf{X})\}$ are properly placed, then by fixing current $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_k, \mathbf{X})\}$ and applying some

technique (e.g., the delta learning rule or the pseudo-inverse method [12]) to adjust $\mathbf{Y}$, the requirement (1) can be achieved very easily. Therefore, the crux of learning is to adjust $\mathbf{X}$ such that $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_k, \mathbf{X})\}$ become properly placed.

3.2. The Linearly Separating Condition

The effort of checking regularly if the current corresponding points $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_k, \mathbf{X})\}$ are properly placed is huge. Furthermore, it is useless when the network is a defective one where there is no such $\mathbf{X}$ that $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_k, \mathbf{X})\}$ are properly placed. Those arguments account for the reason of developing the alternative stopping criterion: the Linearly Separating Condition (LSC).

Let LSC(k, l) denote the condition

$$\min_{c \in K_{l1}(k)} O(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X}) > \max_{c \in K_{l2}(k)} O(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X}),$$

and LSC(k) denote the condition

$$\min_{c \in K_{l1}(k)} O(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X}) > \max_{c \in K_{l2}(k)} O(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X}), \qquad \forall l.$$

Via the LSC(k, l), the network can use the following threshold value for correct classification:

$$v_l = \frac{\min_{c \in K_{l1}(k)} O(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X}) + \max_{c \in K_{l2}(k)} O(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X})}{2},$$

that is,

$$\mathbf{B} \in \begin{cases} \text{Class}_{l1}, & \text{if } O(\mathbf{B}, \mathbf{Y}_l, \mathbf{X}) \geq v_l, \\ \text{Class}_{l2}, & \text{if } O(\mathbf{B}, \mathbf{Y}_l, \mathbf{X}) < v_l. \end{cases}$$

Note that LSC(k) is a sufficient, but not a necessary, condition of a properly placed distribution of $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_k, \mathbf{X})\}$.

The extra effort of checking LSC(k) is to get $\min_{c \in K_{l1}(k)} O(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X})$ and $\max_{c \in K_{l2}(k)} O(\mathbf{B}_c, \mathbf{Y}_l, \mathbf{X})$ for all l, and then compare them. Intuitively, this effort is rather small.

3.3. The Design of the Learning Procedure of RN

The block diagram of the learning process is shown in Figure 2. There are three mechanisms in RN's learning process: the thinking mechanism, the cramming mechanism, and the pruning mechanism. In the following sections, the reasons behind the design of those mechanisms are given in details.

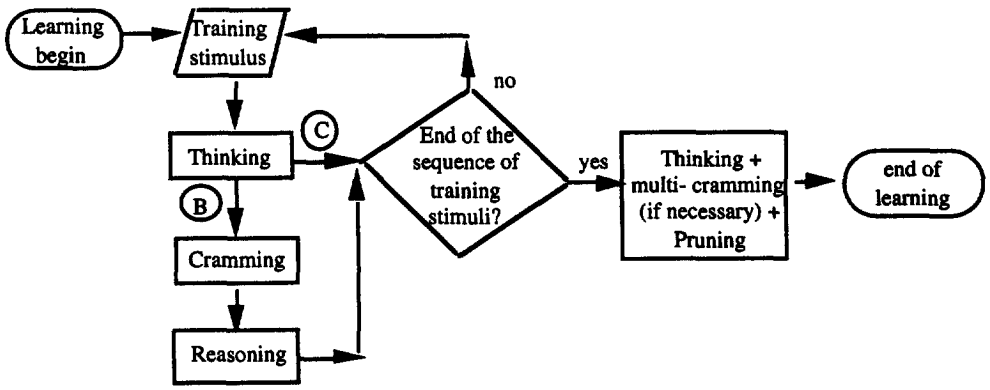

Figure 2. The program flowchart of RN's learning procedure. The details of thinking, reasoning, and cramming are shown in Figure 3 and Figure 4, respectively.

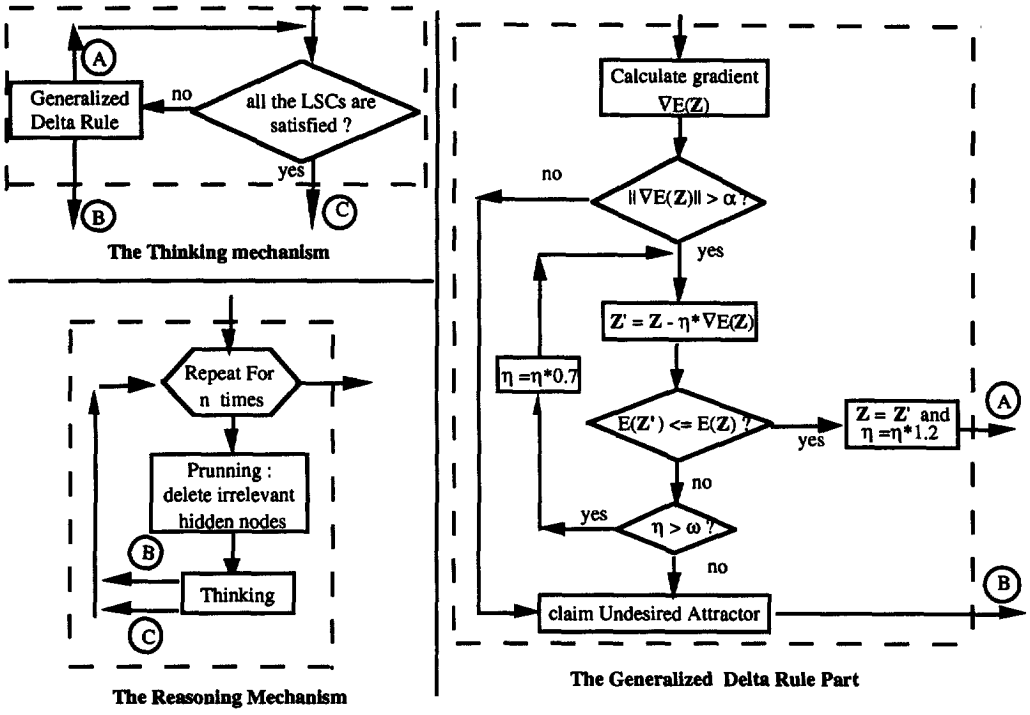

Figure 3. The thinking mechanism, the reasoning mechanism, and the generalized delta rule part. The values of given constants α and ω in the generalized delta rule part are tiny.

Recruiting an extra hidden node with $\mathbf{w}_{p+1} = \mathbf{B}_k$, $\theta_{p+1} = -\mathbf{B}_k^{\mathrm{T}}\mathbf{B}_k + \varepsilon$, $r_{l,p+1} = 0$ for every $l \in L$, and, for every $l \notin L$,

$$r_{l,p+1} = \begin{cases} \max\limits_{u \in K_{l2}(k-1)} \sum\limits_{i=1}^{p} r_{li} h(\mathbf{B}_u, \mathbf{X}_i) - \sum\limits_{i=1}^{p} r_{li} h(\mathbf{B}_k, \mathbf{X}_i), & \text{if } d_{kl} = 1.0, \\ \min\limits_{v \in K_{l1}(k-1)} \sum\limits_{i=1}^{p} r_{li} h(\mathbf{B}_v, \mathbf{X}_i) - \sum\limits_{i=1}^{p} r_{li} h(\mathbf{B}_k, \mathbf{X}_i), & \text{if } d_{kl} = -1.0. \end{cases}$$

Then $p + 1 \rightarrow p$.

Figure 4. The cramming mechanism for the case of one output node and the input values being binary number. $l \in L$ if LSC(k, l) is satisfied before implementing the cramming mechanism.

3.3.1. The thinking mechanism

The thinking mechanism is used to iteratively adjust $\mathbf{Z}$, and it currently executes the momentum version of the generalized delta rule. This thinking mechanism cannot avoid the encountering with an undesirable solution. In Figure 2, Ⓑ indicates the situation when the result of implementing the thinking mechanism is an undesirable solution. A perfect thinking mechanism that can avoid the locally optimal solution is desirable, because it renders the defective network architecture the only cause of imperfect learning result. Unfortunately, there is no such perfect thinking mechanism currently. Under this confinement, together with the consideration of the computing complexity, the current way of implementing the thinking mechanism is adopted. The consideration of the computing complexity is very important because the thinking mechanism will be triggered very frequently during the learning process.

3.3.2. The mechanisms for handling the convergence to an undesirable network solution

With concerning each (new) training stimulus, when an undesirable network solution is obtained after implementing the first thinking mechanism, the proposed strategy is recruiting extra hidden nodes. This leads to the setup of the cramming mechanism.

Let us take the proof in [1] as the illustration. There is a way of implementing the cramming mechanism: first, recruiting a new hidden node with the following activation function:

$$f(x,\lambda) \equiv \frac{1.0 - \exp(-\lambda x)}{1.0 + \exp(-\lambda x)},$$

where x is the net input and λ, the gain parameter, equals 2^{τ} with τ being a large positive integer; then gradually reducing the value of λ to 2. With the associated weights assigned as in Figure 4, the activation value of the new hidden node corresponding to the k^{th} training stimulus is almost $+1$, and the activation values of the new hidden node corresponding to all other training stimuli are almost -1. In the view point of the activation field, recruiting an extra hidden node has introduced an extra dimension into the hidden-layer space. Furthermore, the arrangement of such a new-added hidden node has made the new corresponding point $\mathbf{h}(\mathbf{B}_k, \mathbf{X})$ placed near the $+1$ corner of this extra dimension, and all other new corresponding points placed closely near the -1 corner of this extra dimension. Therefore, if the corresponding $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_{k-1}, \mathbf{X})\}$ are properly placed before introducing the k^{th} training stimulus, then, after recruiting such a hidden node, the new corresponding $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_k, \mathbf{X})\}$ are properly placed.

The above facts also show that LSC(k) can be satisfied via recruiting a hidden node with the tanh activation function and proper (large) weights and threshold. In other words, after recruiting only one hidden node with the tanh activation function, LSC(k) could probably be satisfied via implementing the generalized delta rule.

In summary, the idea of recruiting extra hidden nodes is adopted to handle the defective network structure as well as the undesired predicament of converging to an undesired solution. Recruiting extra hidden nodes will increase the dimension of the weight space such that the trapped attractor could be no longer an attractor. The reasoning mechanism proceeds immediately to render LSC(k) satisfied.

3.3.3. The pruning mechanism

The cramming mechanism handles the imperfect learning result without concerning the reason behind. Thus, the situation of converging to a locally optimal solution will also trigger the cramming mechanism. In other words, the cramming mechanism may recruit excess hidden nodes, some of which later become irrelevant. In a $\mathbf{Z}$, the i^{th} hidden node is irrelevant with respect to the l^{th} output node if LSC(k, l) is still satisfied with the same $\mathbf{Z}$ except $r_{li} = 0$, and a hidden node is irrelevant if it is irrelevant with respect to all output nodes. The irrelevant hidden nodes are useless with respect to LSC; furthermore, they may contribute some significant effort to some output nodes and make the network have a bad generalization ability. So, it is necessary to prune irrelevant hidden nodes.

A collection of more training stimuli may lead to more concise information about the environment of the application problem. In other words, a collection of more training stimuli might lead to less hidden nodes. Therefore, having the pruning mechanism is practical, even when the perfect thinking mechanism is adopted. This also accounts why extra mechanisms (the thinking mechanism and the pruning mechanism) have been arranged at the end of RN to make sure that the number of hidden nodes is as few as possible. If it is allowable to obtain an imperfect learning result after presenting all training stimuli, then, as shown in Figure 2, extra cramming mechanism has been arranged to make sure that LSC is satisfied.

4. SUMMARY AND FUTURE WORK

RN adopts the layered feedforward network structure, thus the essential characteristic of the mapping between two consecutive layers is the level-adjacent mapping. 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 find an $\mathbf{X}$ such that the corresponding $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_k, \mathbf{X})\}$ are properly placed.

In order to spare the (huge and sometimes useless) effort in regularly checking if the current $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_k, \mathbf{X})\}$ are properly placed, LSC is adopted instead. One future work is to derive an effective mechanism for checking regularly if the current corresponding $\{\mathbf{h}(\mathbf{B}_1, \mathbf{X}), \mathbf{h}(\mathbf{B}_2, \mathbf{X}), \ldots, \mathbf{h}(\mathbf{B}_k, \mathbf{X})\}$ are properly placed.

When an undesirable solution is obtained after implementing the first thinking mechanism concerning each (new) training stimulus, the current strategy of RN is: recruiting extra hidden nodes. Another future work is to derive a more effective mechanism for recruiting extra hidden nodes.

Irrelevant hidden nodes are useless with respect to LSC; furthermore, they may occasionally contribute some significant effort to some output nodes and cause a bad generalization ability of the network. So, it is necessary to prune irrelevant hidden nodes. The thinking mechanism and the pruning mechanism are included together in the reasoning mechanism. Another future work is to derive a more effective reasoning mechanism.

REFERENCES

1. R. Tsaih, The softening learning procedure, *Mathl. Comput. Modelling* **18** (8), 61–64 (1993).
2. R. Tsaih, The softening learning procedure for the networks with multiple output nodes, *MIS Review* **4**, 89–93 (1994).
3. R. Tsaih, Learning procedure that guarantees obtaining the desired solution of the 2-classes categorization learning problem, *The 1st Asia-Pacific Conference on Simulated Evolution and Learning*, Korea, pp. 446–453, (1996).
4. E. Watanabe and H. Shimizu, Algorithm for pruning hidden nodes in multi-layered neural network for binary pattern classification problem, *Proceedings of 1993 International Joint Conference on Neural Networks* **I**, 327–330 (1993).
5. Y. Chen, D. Thomas and M. Nixon, Generating-shrinking algorithm for learning arbitrary classification, *Neural Networks* **7**, 1477–1489 (1994).
6. M. Mézard and J. Nadal, Learning in feedforward layered networks: The tiling algorithm, *Journal of Physics* **A 22**, 2191–2204 (1989).
7. S. Fahlman and C. Lebiere, The cascade-correlation learning architecture, In *Advances in Neural Information Processing Systems II*, Denver, 1989, (Edited by D. Touretzky), Morgan Kaufmann, San Mateo, (1990).
8. M. Frean, The upstart algorithm: A method for constructing and training feedforward neural networks, *Neural Computation* **2**, 198–209 (1990).
9. 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).
10. H. Lin, R. Tsaih and R. Jee, Exploring the relative abilities of neural networks and VAR models in forecasting Taiwan bond prices, *Review of Securities and Futures Markets* **9** (1), 63–113 (1997).
11. R. Tsaih, Y. Hsu and C. Lai, Forecasting S&P 500 stock index futures with a hybrid AI system, *Decision Support Systems* (to appear).
12. J. Hertz, A. Krogh and R. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, Reading, MA, (1991).
13. D. Rumelhart, G. Hinton and R. Williams, Learning internal representations by error propagation, In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, (Edited by D. Rumelhart and J. McClelland), pp. 318–362, MIT Press, Cambridge, MA, (1986).