Learning Capability and Storage Capacity of Two-Hidden-Layer Feedforward Networks

Guang-Bin Huang, *Member, IEEE*

***Abstract*—The problem of the necessary complexity of neural networks is of interest in applications. In this paper, learning capability and storage capacity of feedforward neural networks are considered. We markedly improve recent results by introducing neural-network modularity logically. This paper rigorously proves in a constructive method that two-hidden-layer feedforward networks (TLFNs) with $2\sqrt{(m+2)N}$ ($\ll N$) hidden neurons can learn *any* N distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$ with any arbitrarily small error, where m is the required number of output neurons. It implies that the required number of hidden neurons needed in feedforward networks can be decreased significantly, comparing with previous results. Conversely, a TLFN with Q hidden neurons can store *at least* $Q^2/4(m+2)$ any distinct data $(\mathbf{x}_i, \mathbf{t}_i)$ with any desired precision.**

***Index Terms*—Learning capability, neural-network modularity, storage capacity, two-hidden-layer feedforward networks (TLFNs).**

I. Introduction

THE widespread popularity of neural networks in many fields is mainly due to their ability to approximate complex nonlinear mappings directly from the input samples. The necessary complexity of neural networks is one of the most interesting problems in the research and applications of neural networks. Out of many kinds of neural networks, multilayer feedforward neural networks have been investigated more thoroughly. From a mathematical point of view, research on the approximation capabilities of multilayer feedforward neural networks has focused on two aspects: universal approximation in $\mathbf{R}^n$ or one compact set of $\mathbf{R}^n$, i.e., $[a, b]^n$, and approximation in a finite set. Many researchers [3]–[15] have explored the universal approximation capabilities of standard multilayer feedforward neural networks.

In applications, neural networks are trained using finite input samples. It is important to know: 1) how many hidden neurons are needed to learn an input data set and 2) how many different input data can be learned by a neural network with predefined complexity. They are the two sides to the question of the necessary complexity of neural networks.

It is known that N arbitrary distinct samples can be learned precisely by standard single hidden layer feedforward networks (SLFNs) with N hidden neurons with almost any activation function found in applications [2]. However, in most applications large numbers of input samples often need to be dealt with. If it were necessary to use N hidden neurons to precisely learn N distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$, where $\mathbf{x}_i \in \mathbf{R}^n$ and $\mathbf{t}_i \in \mathbf{R}^m$, the complexity (size) of such neural networks would become very large with increasing number of input samples. Tamura and Tateishi [1] extended the research to two-hidden-layer feedforward networks (TLFNs) and found that a TLFN with $(N/2)+3$ hidden neurons and sigmoid activation functions can represent N distinct samples $(\mathbf{x}_i, t_i)$, where $t_i \in \mathbf{R}$, with negligibly small error. However, it is noted that such neural networks designed by Tamura and Tateishi [1] would still be very large in applications. For example, given 10 000 distinct samples $(\mathbf{x}_i, t_i)$, one such TLFN needs around 5000 hidden neurons.

Now, the question is: can a neural network with *much fewer* hidden neurons be capable of learning a large amount of samples with any negligible error?

Inspired by the ideas of Tamura and Tateishi [1] and our previous work [2], in this paper we rigorously prove the significant result that TLFNs with $2\sqrt{(m+2)N}$ hidden neurons can learn N distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$ with negligibly small error, where m is the number of output neurons. That means, the upper bound of the required hidden neurons for a TLFN can be reduced significantly and markedly, and thus, the complexity of the required TLFN can also be reduced sharply. The upper bound of the number of hidden neurons of a TLFN is much less than the number of different input samples. For the above example, one such TLFN with only 340 hidden neurons can learn 10 000 distinct samples $(\mathbf{x}_i, t_i)$, where $t_i \in \mathbf{R}$, with arbitrarily small error.

Compared with our new markedly improved results, the TLFN proposed in Tamura and Tateishi [1] needs a much greater number of neurons and connection weights, which increases the programming complexity. Moreover, in hardware (circuits, VLSI, optical, etc) implementations of neural networks, reductions in the number of neurons are highly desirable. In theory, the complexity of neural networks proposed by Tamura and Tateishi [1] is $O(\sqrt{N/m})$ times of our new results in terms of number of hidden neurons. Conversely, a two-hidden-layer feedforward network (TLFN) with Q hidden neurons can learn *at least* $Q^2/4(m+2)$ different inputs, where m is the number of output neurons.

Our results on learning capability of TLFNs are a significant extension of the work of Tamura and Tateishi [1]. Although our study was inspired by the grouping idea of Tamura and Tateishi [1], there are two key differences between our analysis and theirs. First, the method of Tamura and Tateishi [1] requires one to find a hyperplane to separate the N input vectors into two equal groups and one vector which is orthogonal to the hyperplane. However, in applications it may not be easy

Manuscript received October 11, 2001; revised October 29, 2002.

The author is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore (e-mail: egbhuang@ntu.edu.sg).

Digital Object Identifier 10.1109/TNN.2003.809401

to find such a hyperplane and its orthogonal vectors. Furthermore, such a grouping method by using hyperplanes cannot be extended to the case where more than two groups are required. In contrast, we show that it is not necessary to find such a hyperplane and its orthogonal vectors. In fact, almost any vector can be used to separate N input vectors into multiple groups. Without this multiple grouping way, we cannot get the markedly improved results shown in this paper. Second, we introduce the concept of *neural network modularity* in our paper. That means, although overall the constructive network designed in our paper is a *strictly standard* TLFN, it can be seen as a combinative network consisting of different neural subnetworks playing different roles. In essence, it includes multiple quantizers, each consisting of *two* sigmoid neurons. Logically, each neural quantizer by adjusting the weights and the biases of its neurons can inhibit inputs within some intervals. In contrast, single inhibition neurons used by Tamura and Tateishi [1] is only suitable for two group case and cannot be extended to multiple group cases.

This paper is organized as follows. The necessary preliminaries are given in Section II. Based on these preliminaries, we propose in Section III a new TLFN with $2\sqrt{(m+2)N}$ hidden neurons, which can interpolate N distinct input samples with any arbitrarily small error. Generalization capability of the proposed TLFNs are studied in Section V. Conclusions are given in Section VI.

II. Preliminaries

A. N Distinct Samples Approximation Problem in SLFNs

A standard SLFN with N hidden neurons with activation function $g(x)$ can approximate N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in \mathbf{R}^n$ and $\mathbf{t}_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in \mathbf{R}^m$, with zero error means that there exist β_i, $\mathbf{w}_i$ and b_i such that

$$\sum_{i=1}^{N} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{t}_j, \qquad j = 1, \ldots, N$$

where $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ is the weight vector connecting the ith hidden neuron and the n input neurons, $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ is the weight vector connecting the ith hidden neuron and the m output neurons, and b_i is the bias of the ith hidden neuron. $\mathbf{w}_i \cdot \mathbf{x}_j$ denotes the inner product of $\mathbf{w}_i$ and $\mathbf{x}_j$. The output neurons are chosen to be linear. The above N equations can be written compactly as $\mathbf{H}\beta = \mathbf{T}$, where $\beta = [\beta_1, \ldots, \beta_N]^T$, $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_N]$, and $\mathbf{H} = [h_{ij}]$ with $h_{ij} = g(\mathbf{w}_j \cdot \mathbf{x}_i + b_j)$, $i, j = 1, \ldots, N$. We call $\mathbf{H}$ the hidden layer output matrix[1] of the SLFN; the ith column of $\mathbf{H}$ is the output of the ith hidden neuron with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$.

B. Upper Bounds of Number of Hidden Neurons of Standard SLFNs With Sigmoid Activation Functions

As stated in Lemma 2.3 of Huang and Babri's paper [2], we have the following lemma.

[1]This matrix has been used in [1], [2], and [16], for example.

Lemma 2.1 [2]: For any N distinct vectors $\mathbf{x}_i \in \mathbf{R}^n$, $i = 1, \ldots, N$, for any vector $\mathbf{w}$ chosen from $\mathbf{R}^n$ according to any continuous probability distribution, then with probability one, the N inner products $\mathbf{w} \cdot \mathbf{x}_1, \ldots, \mathbf{w} \cdot \mathbf{x}_N$ are different from each other.

Based on the deduction of Tamura and Tateishi [1, p. 252], we can summarize as shown in Theorem 2.1 that for an SLFN with N hidden neurons and a sigmoid activation function, all the weight vectors $\mathbf{w}_i$ connecting the n input neurons and the ith hidden neurons can be randomly chosen. Formally speaking, we have the following theorem.

Theorem 2.1: Given a standard SLFN with N hidden neurons and sigmoid activation function $g(x) = 1/(1 + e^{-x})$, for any N arbitrary distinct input vectors $\{\mathbf{x}_i | \mathbf{x}_i \in \mathbf{R}^n, i = 1, \ldots, N\}$, for any $\mathbf{w}_i$ and b_i chosen from any intervals of $\mathbf{R}^n$ and $\mathbf{R}$, respectively, according to any continuous probability distribution, then with probability one, the hidden layer output matrix $\mathbf{H}$ of the SLFN is invertible.

Proof: Since all $\mathbf{x}_i$'s are different, according to Lemma 2.1 for any vector $\mathbf{w}$ chosen from any interval of $\mathbf{R}^n$ according to any continuous probability distribution, then with probability one, $\mathbf{w} \cdot \mathbf{x}_1, \ldots, \mathbf{w} \cdot \mathbf{x}_N$ are all different. Simply choose $\mathbf{w}_i = \mathbf{w}$ for all $i = 1, \ldots, N$. Let us consider a vector $\mathbf{c}(b_i) = [g(\mathbf{w}_i \cdot \mathbf{x}_1 + b_i), \ldots, g(\mathbf{w}_i \cdot \mathbf{x}_N + b_i)]^T$, the ith column of $\mathbf{H}$, in Euclidean space $\mathbf{R}^N$, where $b_i \in (a, b)$ and (a, b) is any interval of $\mathbf{R}$.

Following the same proof method of Tamura and Tateishi [1], it can be easily proved that vector $\mathbf{c}$ does not belong to any subspace whose dimension is less than N. Hence, from any interval (a, b) it is possible to choose N bias values $b_1, \ldots, b_N$ for the N hidden neurons such that the corresponding vectors $\mathbf{c}(b_1), \mathbf{c}(b_2), \ldots, \mathbf{c}(b_N)$ span $\mathbf{R}^N$. This means that for any weight vectors $\mathbf{w}_i$ and bias values b_i chosen from any intervals of $\mathbf{R}^n$ and $\mathbf{R}$, respectively, according to any continuous probability distribution, then with probability one, the column vectors of $\mathbf{H}$ can be made full-rank. □

Based on Theorem 2.1, a TLFN with $2\sqrt{(m+2)N}$ hidden neurons which can learn any N different input data with negligible error can be constructed in Section III.

III. Constructive Feedforward Networks

Given N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in \mathbf{R}^n$ and $\mathbf{t}_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in \mathbf{R}^m$, we show in this section that TLFNs with $2\sqrt{(m+2)N}$ hidden neurons can learn them with negligible error.

Because all $\mathbf{x}_i$'s are different, according to Lemma 2.1 there exists a vector $\mathbf{w}$ such that $\mathbf{w} \cdot \mathbf{x}_1, \ldots, \mathbf{w} \cdot \mathbf{x}_N$ are all different. Without loss of generality, assume that the index, i, is reindexed so that $\mathbf{w} \cdot \mathbf{x}_1 < \mathbf{w} \cdot \mathbf{x}_2 < \cdots < \mathbf{w} \cdot \mathbf{x}_N$. For simplicity, consider N to be an integral multiple of one positive integer L. It is obvious that the finite N input vectors can be separated into L groups consisting of N/L input vectors each. Let us denote these L input vector groups as $V^{(1)}, \ldots, V^{(L)}$, where

$$V^{(p)} = \{\mathbf{x}_i | \mathbf{w} \cdot \mathbf{x}_{(p-1)N/L+1} \le \mathbf{w} \cdot \mathbf{x}_i \le \mathbf{w} \cdot \mathbf{x}_{pN/L}\} \qquad p = 1, \ldots, L. \tag{1}$$

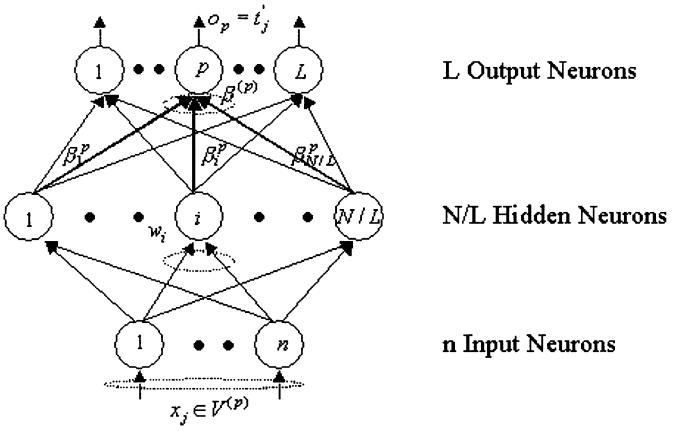

Fig. 1. Subnetwork with n linear input neurons, N/L hidden neurons and L output hidden neurons where the activation function in the hidden neurons and the output neurons is the sigmoid function. If the input vector $\mathbf{x}_j$ belongs to $V^{(p)}$ the output of the pth output neuron is t'_j.

A. One Subnetwork With N/L-Hidden Neurons

For simplicity, we first consider the case where the output dimension $m = 1$. Consider a subnetwork as shown in Fig. 1 and the following subtargets:

$$t'_i = t_i/C + 0.5, \qquad i = 1, \ldots, N \tag{2}$$

where C is a positive constant and is determined so that $t'_i \in (0, 1)$.

The subnetwork has n linear input neurons, N/L hidden neurons and L output neurons. The activation function in hidden and output neurons is the sigmoid activation function $g(x) = 1/(1 + e^{-x})$.

Theorem 3.1: For any N arbitrary distinct input vectors $\{\mathbf{x}_i | \mathbf{x}_i \in \mathbf{R}^n, i = 1, \ldots, N\}$, for any weights $\mathbf{w}_i$ and b_i chosen from any intervals of $\mathbf{R}^n$ and $\mathbf{R}$, respectively, according to any continuous probability distribution, then with probability one, the hidden layer output matrix $\mathbf{H}$ of the network as shown in Fig. 1 is invertible for all input vector groups $V^{(p)}$, $p = 1, \ldots, L$.

Proof: With respect to N/L input vectors $\mathbf{x}_{(p-1)N/L+1}$, $\mathbf{x}_{(p-1)N/L+2}, \ldots, \mathbf{x}_{pN/L}$ in group $V^{(p)}$, where $p = 1, \ldots, L$, the hidden layer output matrix $\mathbf{H}$ of the network as shown in Fig. 1 can be denoted by $\mathbf{H}^{(p)} = [h_{ij}^{(p)}]$ with $h_{ij}^{(p)} = g(\mathbf{w}_j \cdot \mathbf{x}_{(p-1)N/L+j} + b_j), i, j = 1, \ldots, N/L$. According to Theorem 2.1, with respect to input vector group $V^{(1)}$ we can choose $\mathbf{w}_i$ from any interval of $\mathbf{R}^n$, and $b_1^{(1)}, \ldots, b_{N/L}^{(1)}$ from any interval $(a, b) \subset \mathbf{R}$, respectively, such that $\mathbf{H}^{(1)}$ is invertible. We now show that we can choose b_i such that all the L hidden layer output matrices $\mathbf{H}^{(p)}$'s are invertible.

Since the determinant of $\mathbf{H}^{(1)}$ is a continuous function of the bias values, if we set a positive value ϵ_1 sufficiently small, then $\forall b_i \in (b_i^{(1)} - \epsilon_1, b_i^{(1)} + \epsilon_1) \subset (a, b)$, the determinant of $\mathbf{H}^{(1)}$ does not become zero. As described in the proof of Theorem 2.1, in order to make $\mathbf{H}^{(2)}$ full-rank, each bias value of the hidden neurons can be chosen from any interval. Hence, we can choose $b_i^{(2)} \in (b_i^{(1)} - \epsilon_1, b_i^{(1)} + \epsilon_1)$, $i = 1, \ldots, N/L$, which make $\mathbf{H}^{(2)}$ full-rank. Also, since the determinant of $\mathbf{H}^{(2)}$ is a continuous function of the bias values, we can set a positive value ϵ_2 sufficiently small such that $(b_i^{(2)} - \epsilon_2, b_i^{(2)} + \epsilon_2) \subset (b_i^{(1)} - \epsilon_1, b_i^{(1)} + \epsilon_1)$, and $\forall b_i \in (b_i^{(2)} - \epsilon_2, b_i^{(2)} + \epsilon_2)$ the determinant of $\mathbf{H}^{(2)}$ does not become zero. The same procedure can go on for $\mathbf{H}^{(p)}$, $p = 3, \ldots, L$, and we get a bias value set sequence

$$(b_i^{(L)} - \epsilon_L, b_i^{(L)} + \epsilon_L) \subset (b_i^{(L-1)} - \epsilon_{L-1}, b_i^{(L-1)} + \epsilon_{L-1}) \subset \cdots \subset (b_i^{(1)} - \epsilon_1, b_i^{(1)} + \epsilon_1) \subset (a, b).$$

Therefore, $\forall b_i \in (b_i^{(L)} - \epsilon_L, b_i^{(L)} + \epsilon_L)$, $i = 1, \ldots, N/L$, the determinants of $\mathbf{H}^{(p)}$, $p = 1, \ldots, L$, are nonzero. □

Define $\beta^{(p)} = [\beta_1^{(p)}, \ldots, \beta_{N/L}^{(p)}]^T$, where $\beta_i^{(p)}$ denotes the weight vector connecting the ith hidden neuron and the pth output neuron. According to Theorem 3.1 $\beta^{(p)}$ can be chosen suitably such that for any input vector $\mathbf{x}_i \in V^{(p)}$ the expected output of the pth output neuron is t'_i. Thus, such a subnetwork has the feature that for any input vector $\mathbf{x}_i$ within the pth input vector group the output of the pth output neuron is t'_i.

B. Construction of TLFNs

Now we can construct the expected TLFN by introducing a quantizer neural module which consists of two sigmoid neurons, A and B.

First, we can add $2L$ neurons in the hidden layer as shown in Fig. 2. These $2L$ neurons are labeled as $A^{(p)}$ and $B^{(p)}$, where $p = 1, \ldots, L$. All the newly added L $A^{(p)}$ neurons and L $B^{(p)}$ neurons are fully connected to the input layer with weights $\mathbf{w}^{A^{(p)}}$ and $\mathbf{w}^{B^{(p)}}$, $p = 1, \ldots, L$, respectively. However, for each output neuron there are only two newly added neurons linking to it, that is, only two neurons $A^{(p)}$ and $B^{(p)}$ of the $2L$ newly added neurons are linked to the pth output neuron, where $p = 1, \ldots, L$ (cf. Fig. 2).

The weight vectors of the connections linking the input neurons to the newly added hidden neurons $A^{(p)}$ and $B^{(p)}$ are chosen as $\overline{\mathbf{w}}_{A^{(p)}} = T \cdot \mathbf{w}$ and $\overline{\mathbf{w}}_{B^{(p)}} = -T \cdot \mathbf{w}$, respectively, where T is any given positive value, $p = 1, \ldots, L$. Since $\mathbf{w} \cdot \mathbf{x}_1 < \mathbf{w} \cdot \mathbf{x}_2 < \cdots < \mathbf{w} \cdot \mathbf{x}_N$, regarding each newly added hidden neuron $A^{(p)}$, $p = 1, \ldots, L$, its bias $\overline{b}_{A^{(p)}}$ can be chosen as

$$\overline{b}_{A^{(p)}} = -T\left(\mathbf{w} \cdot \mathbf{x}_{pN/L} + \tfrac{1}{2} \min_{i,j} \|\mathbf{w} \cdot \mathbf{x}_i - \mathbf{w} \cdot \mathbf{x}_j\|\right) \tag{3}$$

such that $\overline{\mathbf{w}}_{A^{(p)}} \cdot \mathbf{x}_i + \overline{b}_{A^{(p)}} < 0$ for $i = 1, \ldots, pN/L$ and $\overline{\mathbf{w}}_{A^{(p)}} \cdot \mathbf{x}_i + \overline{b}_{A^{(p)}} > 0$ for $i = pN/L+1, \ldots, N$. That is, $\forall \mathbf{x}_i \in V^{(l)}$, $\overline{\mathbf{w}}_{A^{(p)}} \cdot \mathbf{x}_i + \overline{b}_{A^{(p)}} < 0$ if $l \le p$, and $\overline{\mathbf{w}}_{A^{(p)}} \cdot \mathbf{x}_i + \overline{b}_{A^{(p)}} > 0$ if $l > p$. Regarding each newly added hidden neuron $B^{(p)}$, $p = 1, \ldots, L$, its bias $\overline{b}_{B^{(p)}}$ can be chosen as

$$\overline{b}_{B^{(p)}} = -T\left(\mathbf{w} \cdot \mathbf{x}_{(p-1)N/L+1} + \tfrac{1}{2} \min_{i,j} \|\mathbf{w} \cdot \mathbf{x}_i - \mathbf{w} \cdot \mathbf{x}_j\|\right) \tag{4}$$

such that $\overline{\mathbf{w}}_{B^{(p)}} \cdot \mathbf{x}_i + \overline{b}_{B^{(p)}} < 0$ for $i = (p-1)N/L+1, \ldots, N$ and $\overline{\mathbf{w}}_{B^{(p)}} \cdot \mathbf{x}_i + \overline{b}_{B^{(p)}} > 0$ for $i = 1, \ldots, (p-1)N/L$. That is, $\forall \mathbf{x}_i \in V^{(l)}$, $\overline{\mathbf{w}}_{B^{(p)}} \cdot \mathbf{x}_i + \overline{b}_{B^{(p)}} < 0$ if $l \ge p$ and $\overline{\mathbf{w}}_{B^{(p)}} \cdot \mathbf{x}_i + \overline{b}_{B^{(p)}} > 0$ if $l < p$. Thus, T can be set large enough so that, for $\forall \mathbf{x}_i \in V^{(p)}$, $p = 1, \ldots, L$, the outputs of newly added neurons $A^{(1)}, \ldots, A^{(p)}$ are almost zero and the outputs of newly added neurons $A^{(p+1)}, \ldots, A^{(L)}$ are almost one, and the outputs of newly added neurons $B^{(1)}, \ldots, B^{(p-1)}$ are almost one

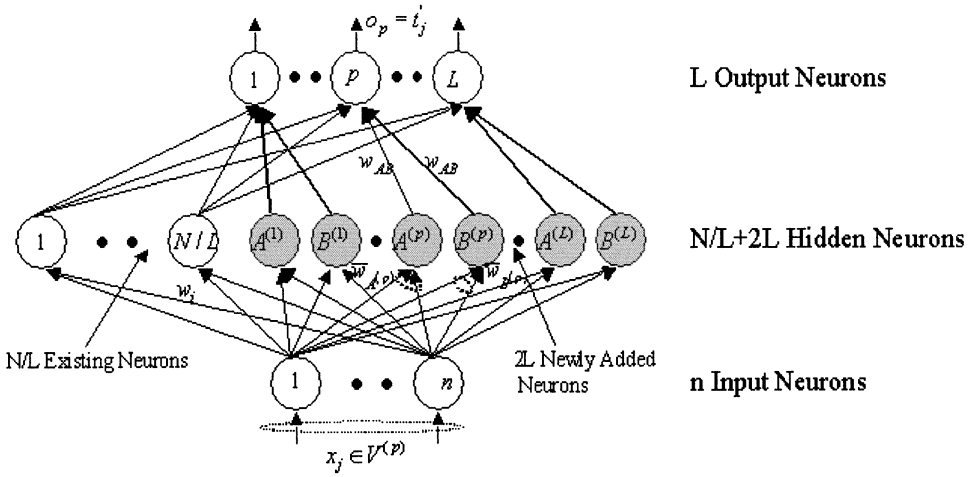

Fig. 2. $2L$ new hidden neurons are added to the subnetwork. For each output neuron p, $p = 1, \ldots, L$, only two newly added neurons $A^{(p)}$ and $B^{(p)}$ are linked to it. All the $2L$ newly added neurons are linked to all the n input neurons.

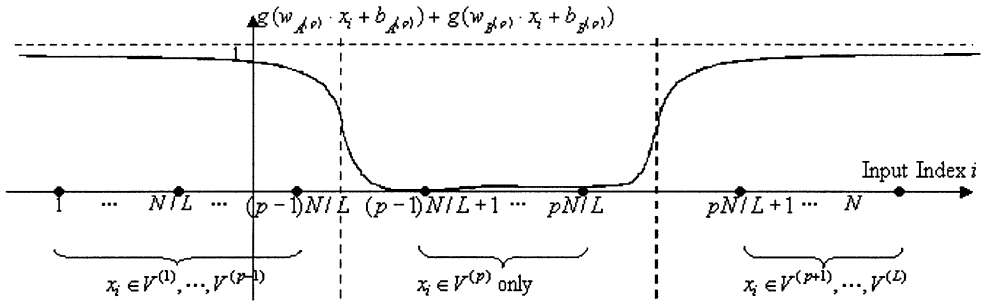

Fig. 3. Peer neurons $A^{(p)}$ and $B^{(p)}$ forms a quantizer module: the output of this module is almost zero if and only if the input vectors are within input group $V^{(p)}$, otherwise its output is almost one.

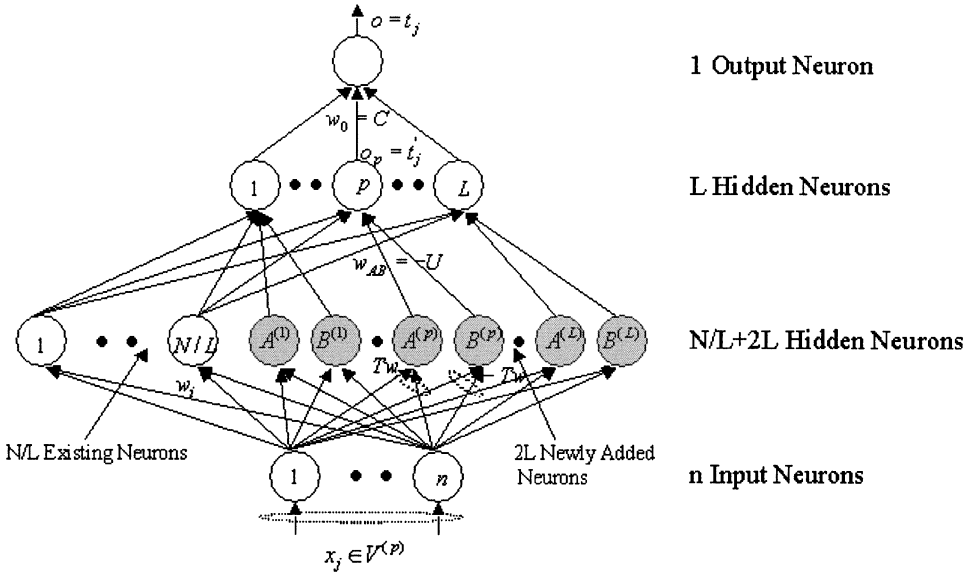

Fig. 4. One newly constructed feedforward network with $N/L + 3L$ ($\leq 2\sqrt{3N}$) hidden neurons can learn the N input samples $(\mathbf{x}_i, t_i)$ with any arbitrarily small error, where $\mathbf{x}_i \in \mathbf{R}^n$ and $t_i \in \mathbf{R}$.

and the outputs of newly added neurons $B^{(p)}, \ldots, B^{(L)}$ are almost zero. In other words, for each input vector group $V^{(p)}$, $p = 1, \ldots, L$, only the output of the pth neural quantizer consisting of neurons $A^{(p)}$ and $B^{(p)}$ is almost zero, and the outputs of the other $L - 1$ quantizers are almost one (cf. Fig. 3).

All the weights w_{AB} of the connections linking these newly added $2L$ hidden neurons to their corresponding output neurons are chosen same value $w_{AB} = -U$. U is a positive value and can be set large enough so that the input of pth output neuron from hidden neurons $A^{(p)}$ and $B^{(p)}$ (pth quantizer) has small

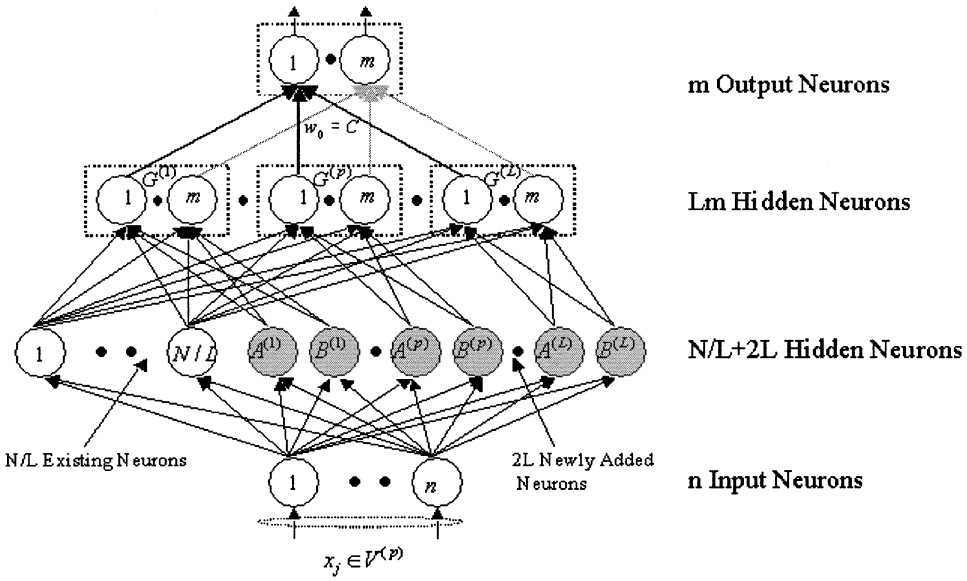

Fig. 5. One newly constructed feedforward network with $N/L + (m + 2)L$ ($\leq 2\sqrt{(m+2)N}$) hidden neurons can learn the N input samples $(\mathbf{x}_i, \mathbf{t}_i)$ with any arbitrarily small error, where $\mathbf{x}_i \in \mathbf{R}^n$ and $\mathbf{t}_i \in \mathbf{R}^m$.

values ϵ_i for any input $\mathbf{x}_i$ within input vector group $V^{(p)}$ and large negative values E_i for any input within other input vector groups $V^{(l)}$, $l \neq p$. We can make E_i and ϵ_i arbitrarily large and small, respectively, by setting T and U sufficiently large. E_i goes to negative infinity and ϵ_i goes to zero.

Similarly to Tamura and Tateishi [1], we add one more linear output neuron which is linked to all the L output neurons in the previous subnetwork. And thus, those L neurons, which functioned as output neurons in the previous subset, now forms the second hidden layer of the newly constructed network (cf. Fig. 4). The bias of the output neuron of newly constructed network is $b_o = -0.5C$ and the weights of the connections between this neuron and the second hidden layer is $w_o = C$. Thus, this new feedforward network which can learn any N different input data with arbitrarily small error.

Since Theorem 3.1 is true for multidimensional output cases, the construction method for one-dimensional case proposed as above can be straightforward extended to the multidimensional case where $\mathbf{t}_i \in \mathbf{R}^m$. For m-dimensional output cases, each output neuron in one-dimensional case as shown in Fig. 1 shall be replaced with m neurons, that means, it shall have L output neuron group $G^{(p)}$, $p = 1, \ldots, L$, each group $G^{(p)}$ consisting of m neurons. According to Theorem 3.1, we can choose weights and biases of hidden neurons such that for any input $\mathbf{x}_i$ of pth input vector group $V^{(p)}$, the output of pth output group $G^{(p)}$ is $\mathbf{t}'_i$, where $t'_{ij} = t_{ij}/C + 0.5$, $i = 1, \ldots, N$, $j = 1, \ldots, m$, where C is a positive constant and is determined so that $t'_{ij} \in (0, 1)$.

We add an output layer with m linear neurons in which the biases are $b_o = -0.5C$ and let only the ith neuron in each neuron group $G^{(p)}$ of the second hidden layer, $p = 1, \ldots, L$, be connected to the ith neuron in the new output layer with corresponding weights $w_o = C$ as shown in Fig. 5. We add L quantizers each consisting of two sigmoid neurons as described above. Thus, this newly constructed network can learn any N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$ with arbitrarily negligible error. This newly constructed feedforward network has two hidden layers. There are $N/L+2L$ and mL hidden neurons in the first and second hidden layers, respectively (cf. Fig. 5).

Note that $N/L + (m + 2)L \geq 2\sqrt{(m+2)N}$, thus, if $L = \sqrt{N/(m+2)}$ we get $N/L + (m + 2)L = 2\sqrt{(m+2)N}$. Therefore, a standard TLFN with $L_1 = \sqrt{(m+2)N} + 2\sqrt{N/(m+2)}$ neurons in first hidden layer and $L_2 = m\sqrt{N/(m+2)}$ in the second hidden layer, respectively, can represent N distinct input samples $(\mathbf{x}_i, \mathbf{t}_i)$ with any desired precision.

IV. Storage Capacity

Since a TLFN with $2\sqrt{(m+2)N}$ hidden neurons can represent N distinct input samples $(\mathbf{x}_i, \mathbf{t}_i)$ with any desired precision, in other words, a TLFN with Q hidden neurons can memorize *at least* $Q^2/4(m+2)$ any distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$ with any desired precision, where m is the number of output neurons. Interestingly, it is easily seen that the storage capacity of a TLFN could be increased by reducing m, the number of output neurons. That means, given the same number of hidden neurons, the storage capacity of a TLFN with one output neuron may be larger than that of a TLFN with multiple output neurons. In some cases one can reduce number of output neurons of a TLFN so as to increase its storage capacity.

Yamasaki [17] showed that $n \cdot \lceil l_1/2 \rceil + \lceil l_1/2 \rceil \cdot \lceil (l_2/2) - 1 \rceil + \cdots + \lceil l_{K-1}/2 \rceil \lceil (l_K/2) - 1 \rceil$ examples in the general position[2] can be memorized by a K-hidden-layer network which has n input units in the input layer, l_k hidden units in the kth hidden layer, and a single output unit in the output layer. Especially, the lower bound of the storage capacity for two-hidden-layer network is $n \cdot \lceil l_1/2 \rceil + \lceil l_1/2 \rceil \cdot \lceil (l_2/2) - 1 \rceil \leq (n-1) \cdot (l_1/2) + (l_1/2) \cdot (l_2/2)$, which is much less than ours for single-output case, $(l_1 + l_2)^2/12$.

[2]A set of n-dimensional vectors is said to be in general position if no subset of n or fewer vectors are linearly dependent [18], [17].

V. Classifiers and Generalization

In the previous section, we have constructed a two-hidden-layer feedforward network with $L_1 = \sqrt{(m+2)N} + 2\sqrt{N/(m+2)}$ neurons in first hidden layer and $L_2 = m\sqrt{N/(m+2)}$ in the second hidden layer, respectively, which can interpolate N distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$ with arbitrary small error. One natural concern is whether the proposed TLFNs can have good generalization capability. Without going into detail to quantify them with full generality in this paper, we can analyze a simplified situation where the proposed architecture is adopted in pattern classification applications.

For simplicity, we can consider the proposed network architecture (*classifier*) with a single output neuron. Assume that we have N training patterns $(\mathbf{x}_i, t_i)$, where $\mathbf{x}_i \in \mathbf{R}^n$ and $t_i \in \{0, 1\}$. In pattern classification applications, the linear output neuron of the proposed network can be simply replaced with a threshold neuron[3] $\operatorname{sgn}(x)$.

A. Generalization Performance

Similar to (2), for the proposed network architecture used as a classifier in classification applications (cf. Fig. 4), we choose

$$t_i' = \lambda \operatorname{sgn}(t_i - 0.5), \qquad i = 1, \ldots, N \tag{5}$$

where λ is a positive factor, $0 < \lambda < 1$, and can be adjusted to achieve optimum generalization performance for the proposed classifier. The weights connecting the second hidden layer to the output neuron are $w_0 = \lambda$, and the bias of the output neuron is $b_0 = 0.5\lambda^2$.

As analyzed in the previous section, similarly for this classifier the weight $\mathbf{w}_i$ and bias b_i of the ith neuron in the first hidden layer (except those $2N/L$ quantizers) can be chosen from any intervals of $\mathbf{R}^n$ and $\mathbf{R}$, respectively. Thus, the values of parameters $\mathbf{w}_i$, b_i can be chosen as small as needed. The weight $\beta^{(p)}$ connecting the neurons (except those quantizer neural modules) of first hidden layer to the pth neuron in the second hidden layer can be chosen suitably such that for any input vector $\mathbf{x}_i \in V^{(p)}$ the expected output of the pth output neuron is t_i'

$$\beta^{(p)} = \mathbf{H}^{-1}(\mathbf{w}_1, \ldots, \mathbf{w}_{N/L}, b_1, \ldots, b_{N/L} \mathbf{x}_{(p-1)N/L+1}, \ldots, \mathbf{x}_{pN/L}) \cdot \begin{bmatrix} \ln \dfrac{1+\lambda\operatorname{sgn}(t_{(p-1)N/L+1} - 0.5)}{1-\lambda\operatorname{sgn}(t_{(p-1)N/L+1} - 0.5)} \\ \vdots \\ \ln \dfrac{1+\lambda\operatorname{sgn}(t_{pN/L} - 0.5)}{1-\lambda\operatorname{sgn}(t_{pN/L} - 0.5)} \end{bmatrix}$$

and we have $\lim_{\lambda \to 0} \|\beta^{(p)}\| = 0$. Thus, the values of parameters $\beta^{(p)}$, w_0, b_0 can be chosen as small as needed by adjusting weight size factor λ.

The weights connecting the input neurons to the quantizers of the first hidden neurons are $T\mathbf{w}$ or $-T\mathbf{w}$. The biases of the pth quantizers A and B [according to equations (3) and (4)] are

$$\bar{b}_{A^{(p)}} = -T\left(\mathbf{w} \cdot \mathbf{x}_{pN/L} + \tfrac{1}{2} \min_{i,j} \|\mathbf{w} \cdot \mathbf{x}_i - \mathbf{w} \cdot \mathbf{x}_j\|\right)$$

$$\bar{b}_{B^{(p)}} = -T\left(\mathbf{w} \cdot \mathbf{x}_{(p-1)N/L+1} + \tfrac{1}{2} \min_{i,j} \|\mathbf{w} \cdot \mathbf{x}_i - \mathbf{w} \cdot \mathbf{x}_j\|\right).$$

All the weights linking the quantizers to second hidden layer are $w_{AB} = -U$.

Based on Bartlett theorem [19, part 2 of Theorem 28] we can study how parameters T, U, λ affect the generalization performance of the proposed network. Note that sigmoid function $g(x) = 1/(1+e^{-x})$ satisfies Lipschitz condition: for all $x_1, x_2 \in \mathbf{R}$, $|g(x_1) - g(x_2)| \le \max_{x \in \mathbf{R}} |g'(x)| = (1/4)|x_1 - x_2|$. Denote by w_i^l the ith weight of computation layer l (l is 1, 2, 3 for the first hidden, second hidden, and output layer, respectively) and ϕ_w the network architecture Φ corresponding to a set w of weights. Thus, a slightly modified Bartlett theorem for sigmoid two-hidden-layer feedforward networks is

Theorem 5.1 [19]: Suppose P is a probability distribution on Z: $X \times \{0, 1\}$, with $X = \mathbf{R}^n$, $0 < \gamma \le 1$, and $0 < \delta < 1/2$. Let $X = \{x \in \mathbf{R}^n : \|x\|_\infty \le B\}$. Given a standard two-hidden-layer sigmoid feedforward network architecture Φ, if $\sum_{i=1}^{l_{\max}} |w_i^l| \le \Lambda$ for $\Lambda \ge 1$, where $l_{\max}$ is the number of neurons in computation layer l, $l = 1, 2, 3$, then with probability at least $1 - \delta$ over a training sample $z \in Z^N$ chosen according to P, very ϕ_w in Φ has

$$er_P(\phi_w) < \widehat{er}_z^{\gamma}(\phi_w) + \sqrt{\frac{c}{N}\left(\frac{B^2\Lambda^{12}}{4^{12}\gamma^6}\log n \log^2 N + \log(1/\delta)\right)} \tag{6}$$

for a constant c.

Constant c depends only on the number of layers (except input layer) of the network architecture Φ, thus, it is fixed for any two-hidden-layer network architecture. $er_P(\phi_w)$, the misclassification probability of ϕ_w that a new subsequent pattern $(\mathbf{x}, t)$ randomly drawn from the same probability distribution P as the training samples is mislabeled by the trained network, depends on two factors: the *sample estimate error* and *complexity penalty* [20]. $\widehat{er}_z^{\gamma}(\phi_w)$ is the estimate error[4] for the N training samples $(\mathbf{x}_i, t_i)$. The complexity penalty $\sqrt{(c/N)((B^2\Lambda^{12}/4^{12}\gamma^6)\log n \log^2 N + \log(1/\delta))}$ depends on the size of the parameters in the network, rather than on the number of parameters.

For the proposed network (classifier), weight size factor λ can be adjusted to make $\|\beta^{(p)}\|$ small enough so that the bound Λ of the weights of the proposed network architecture will only be affected by quantizer factors T and U. In order to get small estimate error $\widehat{er}_z^{\gamma}(\phi_w)$, one can set parameters T, U large enough. However, making parameters T and U too large leads to large values of weight bound Λ, and thus large complexity penalty. On the other hand, small values of parameters T, U (and subsequently small value of weight bound Λ) gives a small complexity penalty, but perhaps with some increases in the estimate error $\widehat{er}_z^{\gamma}(\phi_w)$. Thus, *there is a tradeoff between interpolation accuracy and generalization capability, and proper values of T and U should be chosen in order to optimize generalization performance*. As pointed out by Bartlett [21], for networks with

[3]Threshold function $\operatorname{sgn}: \mathbf{R} \to \{0, 1\}$ is defined as $\operatorname{sgn}(x) = 0$ if $x < 0$, otherwise, $\operatorname{sgn}(x) = 1$.

[4]Strictly speaking, the error estimate counts the proportion of training examples that are not correctly classified by the network (before thresholding the output) with a margin of γ. For appropriate definition, readers can refer to Bartlett [19].

many small weights but small squared error on the training examples, then the Vapnik-Chervonenkis (VC) dimension (and, hence, number of parameters) is irrelevant to the generalization performance. Instead, the magnitude of the weights in the network is more important. We believe that it is true for our proposed network.

B. VC Dimension

The VC dimension is a measure of flexibility of the network Φ showing the number of data points in a space that can be classified arbitrarily. Roughly speaking, assume that Φ is a neural-network architecture with n inputs and m threshold outputs, which is actually a function set of ϕ_w each corresponding to a set w of weights including biases, ϕ_w: $\mathbf{R}^n \rightarrow \{0, 1\}^m$. Given a subset $\mathbf{X} \subseteq \mathbf{R}^n$: $\mathbf{X} = \{\mathbf{x}_i | \mathbf{x}_i \in \mathbf{R}^n, i = 1, \ldots, N\}$. $\mathbf{X}$ is said to be shattered by Φ if for each Boolean function f: $\mathbf{X} \rightarrow \{0, 1\}^m$ there exists some weight set w so that $\phi_w(x) = f(\mathbf{x})$ for all $\mathbf{x} \in \mathbf{X}$. The VC dimension of Φ represented by $\mathrm{VC}\dim(\Phi)$ is the maximal size of a subset $\mathbf{X} \subseteq \mathbf{R}^n$ that is shattered by Φ.

Many researchers have investigated the VC dimension of various feedforward networks. Sakurai [22] showed a lower bound and an upper bound of the VC-dimension of feedforward neural networks with single hidden layer of neurons and single output neuron, where all the neurons have the same activation function of the threshold function and the input patterns are in general position. Sakurai [23] showed a lower bound and an upper bound for the VC-dimension of a set of neural networks of neurons with piecewise polynomial activation functions. Koiran and Sontag [24] showed that W^2 is a lower bound of VC dimension of sigmoid network architectures with $O(W)$ weights. It is shown [20], [25] that there is a feedforward network Φ with K layers and a total of W parameters, where every hidden neuron has sigmoid activation function and the output neuron has threshold function, $\mathrm{VC}\dim(\mathbf{\Phi}) \geq \lfloor K/2 \rfloor \lfloor W/2 \rfloor$. (For brevity, we are ignoring the details of VC theory in this paper since it is not the aim of the current work. The readers can refer to a good reference [20].)

Baum and Haussler [26] showed that $\mathrm{VC}\dim(\Phi) \geq O(W \log K)$ for feedforward network Φ with W weights and K computation units.[5] Shawe-Taylor and Anthony [27] extended Baum and Haussler [26] result from single-output threshold network to multi-output threshold network and showed that it also holds. It is known [20] that any set of input patterns shattered by a network Φ of linear threshold neurons is also shattered by a network Φ' with the same structure as Φ, but with the threshold activation functions replaced by a sigmoid activation function in all nonoutput computation units. Hence, these lower bound results also hold for standard sigmoid networks. It is shown [20], [28], [29] that for a feedforward network $\mathbf{\Phi}$ with W parameters and K computation units, in which each computation unit other than the output unit has the standard sigmoid activation function (the output unit being a linear threshold unit), the best known upper bound on the VC dimension is $O((WK)^2)$.

[5]Computation units of a network Φ are neurons except input neurons.

Thus, our proposed feedforward network architecture Φ satisfies

$$O(W \log K) \leq \mathrm{VC}\dim(\Phi) \leq O((WK)^2). \tag{7}$$

Since we have n input neurons, $\sqrt{(m+2)N} + 2\sqrt{N/(m+2)}$ and $m\sqrt{N/(m+2)}$ neurons in the first and second hidden layers, and m output neurons (in applications m, $n \ll N$), we have

$$O(N \log N) \leq \mathrm{VC}\dim(\Phi) \leq O(N^3). \tag{8}$$

VI. Conclusion

In this paper, we have rigorously proved in a constructive way that TLFNs with $L_1 = \sqrt{(m+2)N} + 2\sqrt{N/(m+2)}$ neurons and $L_2 = m\sqrt{N/(m+2)}$ neurons in the first and second hidden layers, respectively, can learn N distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$ with any arbitrarily small error, where $\mathbf{x}_i \in \mathbf{R}^n$ and $\mathbf{t}_i \in \mathbf{R}^m$. On the other hand, given a TLFN with Q hidden neurons, we know that it can learn *at least* $Q^2/4(m+2)$ distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$ with any desired precision.

Our proposed network architecture has several features:

1) it has large first hidden layer and narrow second hidden layer;
2) the weights connecting the inputs to the first hidden layer can be prefixed and most of weights connecting the first hidden layer and second hidden layer can be determined analytically;
3) it may be trained by only adjusting weight size factor λ and quantizer factors T and U so as to optimize generalization performance;
4) it may have smaller values of parameters (weights and biases);
5) it may be able to *overfit* samples with any arbitrarily small error.

It should come as no surprise that the proposed network architecture may be with good generalization capability if we recall some results from other researchers. Baum [18] claimed that *larger* (threshold) networks can also give interesting generalizations. Such a network might have a very large first hidden layer and a very narrow second hidden layer. Baum [18] also claimed that one may fix the connections on one level and simply adjust the connections on the other level and no gain is possible by using an algorithm able to adjust the weights on both levels simultaneously. Bartlett [19] stated that learning algorithms like back propagation are unlikely to find a network that accurately classifies the training data since they avoid choosing a network that *overfits* the data and they are not powerful enough to find *any* good solution. It is shown [19] that in pattern classification applications, if a large network with small weights has small squared error on the training samples, then the generalization performance depends on the bound of the weights rather than the number of weights. It is interesting to note that our proposed network not only has the features expected by Baum [18] and Bartlett [19], but also shows Baum's claim valid for sigmoid networks. In particular, the generalization performance of such sigmoid networks may be optimized by only adjusting weight size factor λ and quantizer factors T and U. However, the problem

how to adjust weight size factor λ and quantizer factors T and U so as to optimize generalization performance still remains open, we will address this issue in subsequent work (in preparation).

ACKNOWLEDGMENT

The author wishes to thank H. A. Babri, P. L. Bartlett, J. Shawe-Taylor, L.-P. Wang, N. Sundararajan, P. Saratchandran and C. K. Siew for helpful discussions and comments. The author also wishes to thank the anonymous reviewers for their invaluable suggestions.

REFERENCES

[1] S. Tamura and M. Tateishi, "Capabilities of a four-layered feedforward neural network: Four layers versus three," *IEEE Trans. Neural Networks*, vol. 8, pp. 251–255, Mar. 1997.

[2] G.-B. Huang and H. A. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," *IEEE Trans. Neural Networks*, vol. 9, pp. 224–229, Jan. 1998.

[3] ——, "On 'approximation capability in $C(\overline{\mathbf{R}}^n)$ by multilayer feedforward networks and related problems'," *IEEE Trans. Neural Networks*, vol. 9, pp. 714–715, July 1998.

[4] T. Poggio and F. Girosi, "A theory of networks for approximation and learning," Artif. Intell. Lab., Mass. Inst. Technol., A.I. Memo 1140, 1989.

[5] V. Kůrková, "Kolmogorov's theorem and multilayer neural networks," *Neural Networks*, vol. 5, pp. 501–506, 1992.

[6] V. Y. Kreinovich, "Arbitrary nonlinearity is sufficient to represent all functions by neural networks: A theorem," *Neural Networks*, vol. 4, pp. 381–383, 1991.

[7] K. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.

[8] A. Gallant and H. White, "There exists a neural network that does not make avoidable mistakes," in *Artificial Neural Networks: Approximation and Learning Theory*, H. White, Ed. Oxford, U.K.: Blackwell, 1992, pp. 5–11.

[9] M. Stinchcombe and H. White, "Universal approximation using feedforward networks with nonsigmoid hidden layer activation functions," in *Artificial Neural Networks: Approximation and Learning Theory*, H. White, Ed. Oxford, U.K.: Blackwell, 1992, pp. 29–40.

[10] C.-H. Choi and J. Y. Choi, "Constructive neural networks with piecewise interpolation capabilities for function approximations," *IEEE Trans. Neural Networks*, vol. 5, pp. 936–944, Nov. 1994.

[11] Y. Ito, "Approximation of continuous functions on $\mathbf{R}^d$ by linear combinations of shifted rotations of a sigmoid function with and without scaling," *Neural Networks*, vol. 5, pp. 105–115, 1992.

[12] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, pp. 251–257, 1991.

[13] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Networks*, vol. 6, pp. 861–867, 1993.

[14] J. Wray and G. G. Green, "Neural networks, approximation theory and finite precision computation," *Neural Networks*, vol. 8, no. 1, pp. 31–37, 1995.

[15] T. Chen, H. Chen, and R.-W. Liu, "Approximation capability in $C(\overline{\mathbf{R}}^n)$ by multilayer feedforward networks and related problems," *IEEE Trans. Neural Networks*, vol. 6, no. jan., pp. 25–30, 1995.

[16] F. Albertini, E. D. Sontag, and V. Maillot, "Uniqueness of weights for neural networks," in *Artificial Neural Networks for Speech and Vision*, R. J. Mammone, Ed. London, U.K.: Chapman and Hall, 1993, pp. 113–125.

[17] M. Yamasaki, "The lower bound of the capacity for a neural network with multiple hidden layers," in *Proc. 1993 World Congr. Neural Networks*, 1993, pp. 544–547.

[18] E. Baum, "On the capabilities of multilayer perceptrons," *J. Complexity*, vol. 4, pp. 193–215, 1988.

[19] P. L. Barlett, "The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network," *IEEE Trans. Inform. Theory*, vol. 44, pp. 525–536, Apr. 1998.

[20] M. Anthony and P. L. Bartlett, *Neural Network Learning: Theoretical Foundations*. Cambridge, U.K.: Cambridge Univ. Press, 1999.

[21] P. L. Bartlett, "For valid generalization, the size of the weights is more important than the size of the network," in *Advances in Neural Information Processing Systems'1996*, M. Mozer, M. Jordan, and T. Petsche, Eds. Cambridge, MA: MIT Press, 1997, vol. 9, pp. 134–140.

[22] A. Sakurai, "Tighter bounds of the VC-dimension of three-layer networks," in *Proc. 1993 World Congr. Neural Networks*, 1993, pp. 540–543.

[23] ——, "Tight bounds for the VC-dimension of piecewise polynomial networks," in *Advances in Neural Information Processing Systems'1998*, M. Kearns, S. Solla, and D. Cohn, Eds. Cambridge, MA: MIT Press, 1999, vol. 11, pp. 323–329.

[24] P. Koiran and E. D. Sontag, "Neural networks with quadratic VC dimension," *J. Comput. Syst. Sci.*, vol. 54, pp. 190–198, 1997.

[25] P. L. Bartlett, V. Maiorov, and R. Meir, "Almost linear VC-dimension bounds for piecewise polynomial networks," *Neural Comput.*, vol. 10, pp. 2159–2173, 1998.

[26] E. B. Baum and D. Hausslet, "What size net gives valid generalizations?," *Neural Comput.*, vol. 1, pp. 151–160, 1989.

[27] J. Shawe-Taylor and M. Anthony, "Sample sizes for multiple output threshold networks," *Network*, vol. 2, pp. 107–117, 1991.

[28] J. Shawe-Taylor, "Sample sizes for sigmoidal neural networks," in *Proc. 8th Annu. Conf. Computational Learning Theory*, 1995, pp. 258–264.

[29] M. Karpinski and A. Macintyre, "Polynomial bounds for vc dimension of sigmoidal neural networks," in *Proc. 27th Annu. ACM Symp. Theory Computing*, 1995, pp. 200–208.

Guang-Bin Huang (M'98) received the B.Sc. degree in applied mathematics and M.Eng. degree in computer engineering from Northeastern University, P.R. China, in 1991 and 1994, respectively, and Ph.D. degree in electrical engineering from Nanyang Technological University, Singapore, in 1999.

From June 1998 to May 2001, he worked as Research Fellow in Singapore Institute of Manufacturing Technology (formerly known as Gintic Institute of Manufacturing Technology) where he has led/implemented several key industrial projects. Since May 2001, he has been working as an Assistant Professor in the Information Communication Institute of Singapore (ICIS), School of Electrical and Electronic Engineering, Nanyang Technological University. His current research interests include soft computing and networking.