ICICS-PCM 2003
15-18 December 2003
Singapore

Simplification of a Specific Two-Hidden-Layer Feedforward Networks

Lei Chen, Guang-Bin Huang, and Chee-Kheong Siew
School of Electrical and Electronic Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798.
E-Mail: {pg05561791, egbhuang, ecksiew}@ntu.edu.sg

Abstract

In this paper, we further investigate a specific Two-Hidden-Layer Feedforward Networks(TLFNs) proposed by Huang[1]. We introduce a method to simplify the structure of the TLFNs by introducing a new type of quantizers that unite two previous neurons $A^{(p)}$ and $B^{(p)}$ into a single neuron. Those new quantizers choose a special type of function as the neural network's activation function, which leads to the new TLFNs with $2\sqrt{(m+1)N}$ 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, and unlike Huang's TLFNs require $2\sqrt{(m+2)N}$ hidden neurons. Moreover, it is not necessary to estimate the quantizer value U defined in Huang's TLFNs, which is fixed in our new model of TLFNs. It can reduce significantly and markedly the complexity and computation of neural networks.

1 Introduction

In recent years, neural networks have attracted considerable attention in many fields mainly because of their capability to approximate complex nonlinear mappings directly from the input samples. Many kinds of multi-layer feedforward neural networks have been investigated thoroughly. From a mathematical point of view, research on the approximation capabilities of multi-layer feedforward neural networks has focused on universal approximation in $\mathbf{R^n}$ or one compact set of $\mathbf{R^n}$ [2]-[11].

From Huang's research[9], it is known that N arbitrarily distinct samples can be learned precisely by standard SLFNs with N hidden neurons with almost any activation function found in applications. However, in most applications, large numbers of input samples often need to be dealt with. If we 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 that would increase the running time and reduce the computation speed, thereby leading to the poor performance of the whole system. Thus, Huang[1] further introduced a kind of TLFNs with a novel neural networks' structure. Huang[1] rigorously proves that TLFNs with at most $2\sqrt{(m+2)N}$ hidden neurons can learn N distinct samples $(\mathbf{x_i}, \mathbf{t_i})$ with any negligibly small error, where m is the number of output neurons. The size of the networks can be sharply reduced.

In this paper, we will introduce a method to simplify the structure of Huang's TLFNs by introducing new quantizers that unite two neurons $A^{(p)}$ and $B^{(p)}$ into a neuron. This change makes the new TLFNs with $2\sqrt{(m+1)N}$ hidden neurons learn N distinct samples $(\mathbf{x_i}, \mathbf{t_i})$ with any negligibly small error, where m is the number of output neurons. Since in normal circumstances, N is a very large value, so the upper bound of the required hidden neurons for a TLFNs can be reduced significantly and markedly, therefore, it can simplify remarkably the computation and complexity of neural networks. The most important point is that we construct a special type of function as quantizers' activation function so that it can fix some parts of weight value, and needless to estimate previous the value U. Only to estimate the value T can reach Huang's TLFNs' objective, which also decrease sharply the complexity of neural networks.

This paper is organized as follows. The necessary preliminaries are given in Section 2. Based on these preliminaries, we construct a function and propose a new TLFN with $2\sqrt{(m+1)N}$ hidden neurons in Section 3, which can learn N distinct input samples with any arbitrarily small error. Conclusions are given in Section 4.

2 Preliminaries

A. N distinct samples approximation problem in SLFNs

Firstly, we introduce some symbols about single-hidden-layer feedforward networks(SLFNs). A standard SLFNs with N hidden neurons with activation function $g(x) = \frac{1}{1+e^{-x}}$ can approximate N arbitrary distinct samples $(\mathbf{x_i}, \mathbf{t_i})$, where $\mathbf{x_i} = [x_{i1}, x_{i2}, \cdots, x_{in}]^T \in \mathbf{R^n}$ and $\mathbf{t_i} = [t_{i1}, t_{i2}, \cdots, t_{im}]^T \in \mathbf{R^m}$, with zero error means that there

exist β_i, $\mathbf{w_i}$ and b_i such that

$$\sum_{i=1}^{N} \beta_i g(\mathbf{w_i} \cdot \mathbf{x_j} + b_i) = \mathbf{t_j}, \ \ j = 1, \cdots, N,$$

where $\mathbf{w_i} = [w_{i1}, w_{i2}, \cdots, w_{in}]^T$ is the weight vector connecting the i-th hidden neuron and the n input neurons, $\beta_i = [\beta_{i1}, \beta_{i2}, \cdots, \beta_{im}]^T$ is the weight vector connecting the i-th hidden neuron and the m output neurons, and b_i is the bias of the i-th 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 $\beta = [\beta_1, \cdots, \beta_N]^T$, $\mathbf{T} = [\mathbf{t_1}, \cdots, \mathbf{t_N}]$, and $\mathbf{H} = [h_{ij}]$ with $h_{ij} = g(\mathbf{w_j} \cdot \mathbf{x_i} + b_j)$, $i, j = 1, \cdots, N$. We call $\mathbf{H}$ as the hidden layer output matrix of the SLFNs; the i-th column of H is the output of the ith hidden neuron with respect to inputs $\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_N}$.

B. N distinct samples approximation problem in TLFNs

We briefly introduce TLFNs mentioned in Huang's paper[1]. He divides N arbitrary distinct samples $(\mathbf{x_i}, \mathbf{t_i})$ into L groups, then he introduce the concept of neural network modularity, i.e., new hidden neurons $A^{(p)}$ and $B^{(p)}$. The bias of hidden neuron $A^{(p)}$ is chosen as

$$\begin{aligned} b_{A^{(p)}} &= -T\Big(\mathbf{w} \cdot \mathbf{x_{pN/L}} \\ &+ \frac{1}{2} \min_{i,j} \|\mathbf{w} \cdot \mathbf{x_i} - \mathbf{w} \cdot \mathbf{x_j}\|\Big) \end{aligned} \quad (1)$$

such that $\mathbf{w_{A^{(p)}}} \cdot \mathbf{x_i} + b_{A^{(p)}} < 0$ for $i = 1, \ldots, pN/L$ and $\mathbf{w_{A^{(p)}}} \cdot \mathbf{x_i} + b_{A^{(p)}} > 0$ for $i = pN/L+1, \ldots, N$, where $\mathbf{w_{A^{(p)}}} = T \cdot \mathbf{w}$ is the weight vector connecting the input neurons and the first-hidden-layer neurons $A^{(p)}$ and the vector $\mathbf{w}$ makes the N inner products $\mathbf{w} \cdot \mathbf{x_1}, \cdots, \mathbf{w} \cdot \mathbf{x_N}$ different from each other.

Similarly, the bias of hidden neuron $B^{(p)}$ is chosen as

$$\begin{aligned} b_{B^{(p)}} &= T\Big(\mathbf{w} \cdot \mathbf{x_{(p-1)N/L+1}} \\ &- \frac{1}{2} \min_{i,j} \|\mathbf{w} \cdot \mathbf{x_i} - \mathbf{w} \cdot \mathbf{x_j}\|\Big) \end{aligned} \quad (2)$$

such that $\mathbf{w_{B^{(p)}}} \cdot \mathbf{x_i} + b_{B^{(p)}} < 0$ for $i = (p-1)N/L+1, \ldots, N$ and $\mathbf{w_{B^{(p)}}} \cdot \mathbf{x_i} + b_{B^{(p)}} > 0$ for $i = 1, \ldots, (p-1)N/L$, where $\mathbf{w_{B^{(p)}}} = -T \cdot \mathbf{w}$ is the weight vector connecting the input neurons and the first-hidden-layer neurons$B^{(p)}$.

All the weights w_{AB} of the connections linking these $2L$ hidden neurons $A^{(p)}$ and $B^{(p)}$ to their corresponding the second hidden-layer neurons are chosen same value $w_{AB} = -U$, weight U is a positive value. For more detail contents, you can reference Huang's paper[1].

C. Two lemmas

Two lemmas, which will be referred to late, are:

Lemma 2.1. [1, 9] *For any N distinct vectors $\mathbf{x_i} \in \mathbf{R^n}$, $i = 1, \cdots, N$, for any vector $\mathbf{w}$ chosen from $\mathbf{R^n}$ according to any continuous probability distribution, then with probability one, the N inner products $\mathbf{w} \cdot \mathbf{x_1}, \cdots, \mathbf{w} \cdot \mathbf{x_N}$ are different from each other.*

Lemma 2.2. [1] *For any N arbitrary distinct input vectors $\{\mathbf{x_i} | \mathbf{x_i} \in \mathbf{R^n}, \mathbf{i} = \mathbf{1}, \cdots, \mathbf{N}\}$, 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 Figure is invertible for all input vector groups $V^{(p)}$, $p = 1, \cdots, L$.*

Given N arbitrary distinct samples $(\mathbf{x_i}, \mathbf{t_i})$, where $\mathbf{x_i} = [x_{i1}, x_{i2}, \cdots, x_{in}]^T \in \mathbf{R^n}$ and $\mathbf{t_i} = [t_{i1}, t_{i2}, \cdots, t_{im}]^T \in \mathbf{R^m}$, we show in following section that TLFNs with $2\sqrt{(m+1)N}$ hidden neurons can learn them with negligible error.

3 Constructive Feedforward Networks

Now, we construct the two hidden neuron layers networks in this paper by uniting two neurons $A^{(p)}$ and $B^{(p)}$ into a neuron.

Because all $\mathbf{x_i}$ are different, according to Lemma 2.1 there exists a vector $\mathbf{w}$ such that $\mathbf{w} \cdot \mathbf{x_1}$, $\cdots$, $\mathbf{w} \cdot \mathbf{x_N}$ are all different. Without loss of generality, assume that the index, i, is reindexed so that $\mathbf{w} \cdot \mathbf{x_1} < \mathbf{w} \cdot \mathbf{x_2} < \cdots < \mathbf{w} \cdot \mathbf{x_N}$ and choose $\mathbf{w_i} = \mathbf{w}$. 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 $V^{(1)}, \cdots, V^{(L)}$, where

$$\begin{aligned} V^{(p)} &= \{\mathbf{x_i} | \mathbf{w} \cdot \mathbf{x_{(p-1)N/L+1}} \leq \mathbf{w} \cdot \mathbf{x_i} \\ &\leq \mathbf{w} \cdot \mathbf{x_{pN/L}}, \ p = 1, \cdots, L.\} \end{aligned} \quad (3)$$

For simplicity, we first consider the case where the output dimension $m = 1$. Ahead of being added new quantizers, the subnetwork has n linear input neurons, N/L the first hidden-layer neurons, L the second hidden-layer neurons and one output neurons. The activation function in hidden and output neurons is the sigmoid activation function $g(x) = \frac{1}{1+e^{-x}}$.

Now we add L neurons into the networks. These L neurons are labeled as $C^{(p)}$, where $p = 1, \cdots, L$. All the newly added L $C^{(p)}$ neurons are fully connected to the input layer with weights $\mathbf{w_{C^{(p)}}}$, $p = 1, \cdots, L$, respectively. However, for the L second hidden-layer neurons, there are only one newly added neurons linking to it, that is, only one neurons $C^{(p)}$ of the L newly added neurons are linked to the p-th the second hidden-layer neurons, where $p = 1, \cdots, L$.(cf. Figure 1)

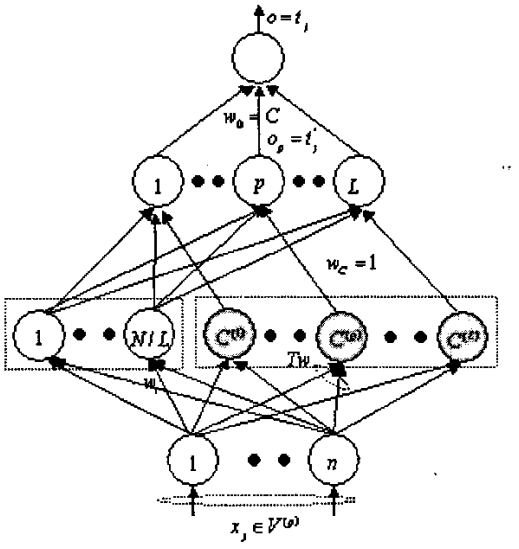

Figure 1: One newly constructed feedforward network with $N/L + 2L$ hidden neurons can learn the N input samples $(\mathbf{x}_i, \mathbf{t}_i)$ with any arbitrarily small error, where $\mathbf{x}_i \in \mathbf{R}^n$ and $t_i \in \mathbf{R}$.

The weight vectors of the connections linking the input neurons to the newly added hidden neurons $C^{(p)}$ are chosen as $\mathbf{w}_{C^{(p)}} = T \cdot \mathbf{w}$, respectively, where T is any given positive value, $p = 1, \cdots, L$.

Next, we will fix the weights between the new quantizers and the second hidden-layer neurons by constructing a special kind of activation function.

We consider two functions, named

$$g_1(x) = \frac{1}{1+e^{-x}}, \text{(cf. Figure 2(a))} \tag{4}$$

and

$$g_2(x) = \frac{1}{1+e^{x}}, \text{(cf. Figure 2(b))} \tag{5}$$

then we unit the two functions (4) and (5) into a function by setting different bias.

$$\overline{f}_{(p)}(x) = g_1(T\mathbf{w}\mathbf{x}_\mathbf{i} + b_{A^{(p)}}) + g_2(T\mathbf{w}\mathbf{x}_\mathbf{i} + b_{B^{(p)}}), \tag{6}$$

(cf. Figure 2(c)) where the bias $b_{A^{(p)}}$ and $b_{B^{(p)}}$ are defined in equations (1) and (2).

Similar to Huang[1]'s analysis, for each input vector group $V^{(p)}$, $p = 1, \cdots, L$, when T is set large enough, only the output of the pth neural quantizer consisting of neurons $C^{(p)}$ is almost zero, the outputs of the other $L-1$ quantizers are almost one. We construct the following function as the p-th quantizers' activation function

$$f_{(p)}(x) = \frac{\overline{f}_{(p)}(x)}{\overline{f}_{(p)}(x) - 1}, \text{(cf. Figure 2(d))} \tag{7}$$

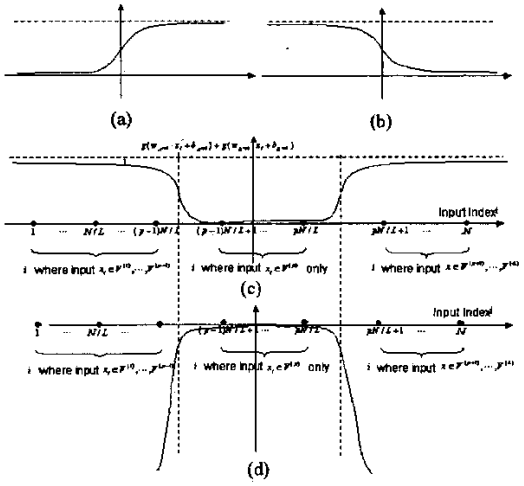

Figure 2: Four activation functions.

This activation function has the following properties. For each input vector group $V^{(p)}$, $p = 1, \cdots, L$, when T is set large enough, i.e., $\lim_{x\to-\infty} f(x) = \lim_{x\to\infty} f(x) = -\infty$, and only the output of the p-th neural quantizer consisting of neurons $C^{(p)}$ is almost zero, the outputs of the other $L-1$ quantizers are almost negative infinite. So we can set the weights w_c of the connections linking these L hidden neurons $C^{(p)}$ to their corresponding second hidden-layer neurons are chosen same value $w_c = 1$(cf. Figure 1). Then the input of p-th output neuron from hidden neurons $C^{(p)}$ (the p-th quantizer) has small values ϵ_i for any input $\mathbf{x}_i$ belonging to 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 sufficiently large. E_i goes to negative infinity and ϵ_i goes to zero. According to lemma 2.2, we can learn N arbitrary distinct samples $(\mathbf{x_i}, \mathbf{t_i})$ with negligible error.

For multi-dimension output case, we can also obtain corresponding results. The multi-output networks are shown in (cf. Figure 3)

Remark: We note that all the number of hidden neurons are $N/L + (m+1)L \geq 2\sqrt{(m+1)N}$, thus, if $L = \sqrt{N/(m+1)}$, we get $N/L + (m+1)L = 2\sqrt{(m+1)N}$. Therefore, a standard two-hidden-layer feedforward network (TLFN) with $L_1 = \sqrt{(m+1)N} + 2\sqrt{\frac{N}{m+1}}$ neurons in first hidden layer and $L_2 = m\sqrt{\frac{N}{m+1}}$ in the second hidden layer, respectively, can represent N distinct input samples $(\mathbf{x}_i, \mathbf{t}_i)$ with any small error. Because in normal circumstances, N is very large, so it can simplify remarkably the number of neurons in neural networks, i.e., the upper bound of the required hidden neurons for a TLFNs. The most important point is that in this improvement, because

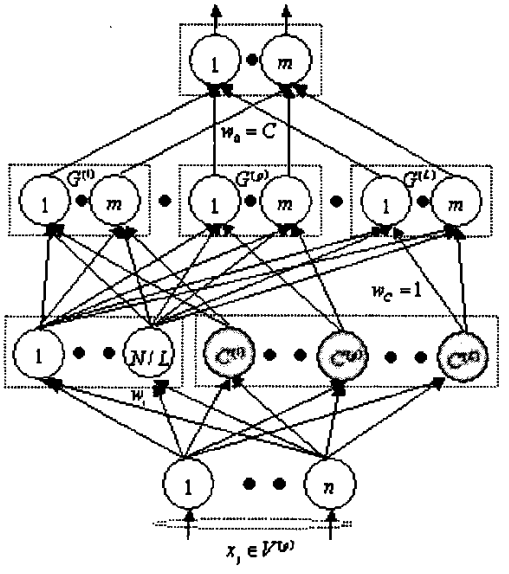

Figure 3: One newly constructed feedforward network with $N/L + (m+1)L$ hidden neurons can learn the N input samples $(\mathbf{x}_i, \mathbf{t}_i)$ with any arbitrarily small error, where $\mathbf{x}_i \in \mathbf{R}^n$ and $\mathbf{t}_i \in \mathbf{R}^m$.

we construct a special type of function (cf. equation (7)) as quantizers' activation function, we also omit the previous weight value U by fixing the parts of neurons' value, then we only need to estimate the value T to learn N arbitrary distinct samples $(\mathbf{x_i}, \mathbf{t_i})$ with negligible error. But because in Huang's constructive method, the value U must choose very large positive number, moreover, U and T have close relations each other, so U's estimation is very difficult. However, in this paper, needless to estimate U, we can reach the networks' objective, so the complexity of the revised TLFNs can also be reduced sharply.

4 Conclusions

In this paper, we have simplified the structure of Huang's TLFNs by introducing new quantizers that unite two neurons $A^{(p)}$ and $B^{(p)}$ into a neuron. This changing leads to the new TLFNs with $2\sqrt{(m+1)N}$ 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. Moreover, we construct a special kind of function as new quantizers' activation function so that only the output of the p-th neural quantizer consisting of neurons $C^{(p)}$ is almost zero, the outputs of the other $L-1$ quantizers are almost negative infinite, which needn't estimate previous the value U. These improvements can reduce the complexity and computation of Huang's networks.

There remains some future work to be done. One is to select a proper value of T to a problem such that neural networks have better perfomance. The other is to prove the structure's universal approxamation in theory.

References

[1] G.-B. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 274–281, 2003.

[2] Y. Ito, "Approximation of functions on a compact set by finite sums of a sigmoid function without scaling," *Neural Networks*, vol. 4, pp. 817–826, 1991.

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

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

[5] T. Chen, H. Chen, and R.-W. Liu, "Approximation capability in $C(\overline{\mathbf{R}^n})$ by multilayer feedforward networks and related problems," *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 25–30, 1995.

[6] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoid function," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 930–945, 1993.

[7] E. Lavretsky, "On the geometric convergence of neural approximation," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 274–282, 2002.

[8] G.-B. Huang and H. A. Babri, "General approximation theorem on feedforward networks," in *Proceedings of the International Conference on Information, Communications and Signal Processing(ICICS'97)*, (Singapore), pp. 698–702, 9-12 September, 1997.

[9] 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 Transactions on Neural Networks*, vol. 9, no. 1, pp. 224–229, 1998.

[10] G.-B. Huang, Y.-Q. Chen, and H. A. Babri, "Classification ability of single hidden layer feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 799–801, 2000.

[11] S. Tamura and M. Tateishi, "Capabilities of a four-layer feedforward neural networks: Four layers versus three," *IEEE Transactions on Neural Networks*, vol. 8, no. 2, pp. 251–255, 1997.