Wavelet Neural Networks for Function Learning

Jun Zhang, *Member, IEEE,* Gilbert G. Walter, Yubo Miao, and Wan Ngai Wayne Lee, *Member, IEEE*

Abstract— **In this paper, a wavelet-based neural network is described. The structure of this network is similar to that of the radial basis function (RBF) network, except that here the radial basis functions are replaced by orthonormal scaling functions that are not necessarily radial-symmetric. The efficacy of this type of network in function learning and estimation is demonstrated through theoretical analysis and experimental results. In particular, it has been shown that the wavelet network has universal and L^2 approximation properties and is a consistent function estimator. Convergence rates associated with these properties are obtained for certain function classes where the rates avoid the "curse of dimensionality." In the experiments, the wavelet network performed well and compared favorably to the MLP and RBF networks.**

I. Introduction

DEVELOPING MODELS from observed data, or function learning, is a fundamental problem in many fields, such as statistical data analysis, signal processing, control, forecasting, and artificial intelligence. This problem is also frequently referred to as function estimation, function approximation, system identification, and regression analysis. There are two general approaches to function learning, namely, the parametric and the non-parametric approach. When the observed data is contaminated (such that it does not follow a pre-selected parametric family of functions or distributions closely) or when there are no suitable parametric families, the non-parametric approach provides more robust results and hence, is more appropriate.

Recently, neural networks have become a popular tool in non-parametric function learning due to their ability to learn rather complicated functions. The multi-layer perceptron (MLP) [1], along with the back-propagation (BP) training algorithm, is probably the most frequently used type of neural network in practical applications. However, due to its multi-layered structure and the greedy nature of the BP algorithm, the training processes often settle in undesirable local minima of the error surface or converge too slowly. The radial basis function (RBF) network [2], as an alternative to the MLP, has a simpler structure (one hidden layer). With some preprocessing on the training data, such as clustering, the training of RBF networks can be much easier than MPL networks. Hence, there has been considerable interest in the implementation of RBF networks [3]–[5] (also see the references in [8]) and the theoretical analysis of their properties, such as approximation ability and convergence rates [6]–[8].

From the point of view of function representation, an RBF network is a scheme that represents a function of interest by using members of a family of compactly (or locally) supported basis functions. The locality of the basis functions makes the RBF network more suitable in learning functions with local variations and discontinuities. Furthermore, the RBF networks can represent any function that is in the space spanned by the family of basis functions. However, the basis functions in the family are generally not orthogonal and are redundant. This means that for a given function, its RBF network representation is not unique and is probably not the most efficient. In this work, we replace the family of basis functions for the RBF network by an orthonormal basis, namely, the scaling functions in the theory of wavelets. The resulting network, called the wavelet neural network,[1] provides a unique and efficient representation of the given function. At the same time, it preserves most of the advantages of the RBF network. The use of orthogonal wavelets in the network also facilitates the theoretical analysis of its asymptotic properties, such as universal approximation and consistency. The wavelet-based network, however, is generally not an RBF network since multi-dimensional scaling functions can be non radial-symmetric.

The idea of using wavelets in neural networks has also been proposed recently by Zhang and Benveniste [9] and Pati and Krishnaprasad [10]. However, the wavelets they used are non-orthogonal and form frames rather than orthonormal bases.[2] In fact, the wavelet networks they studied can be considered as special cases of RBF networks. Nevertheless, the use of wavelet frames provides a good framework for determining the structure of the RBF network. Finally, while revising this paper, we became aware of Boubez and Peskin's work [11], [12] on orthogonal wavelet based networks. While our network structure has some similarity to that of Boubez and Peskin's, our work differs from theirs in several aspects. First, our work is concerned more with the estimation of arbitrary L^2 functions (energy-finite and continuous or discontinuous) rather than those for pattern classification as in [11], [12]. Secondly, we looked into different ways of determining the network weights. Thirdly, we performed theoretical analysis of the asymptotic

Manuscript received October 9, 1993; revised October 13, 1994. J. Zhang was supported in part by the NSF under Grant DIRIS-9010601 and Y. Miao was supported by a grant from Modine Manufacturing Company. The associate editor coordinating the review of this paper and approving it for publication was Prof. J. N. Hwang.

J. Zhang is with the Department of Electrical Engineering and Computer Science, University of Wisconsin-Milwaukee, Milwaukee, WI 53201 USA.

G. G. Walter is with the Department of Mathematics, University of Wisconsin-Milwaukee, Milwaukee, WI 53201 USA.

Y. Miao is with the Marshfield Clinic, Marshfield, WI 54449 USA.

W. N. W. Lee is with the Modine Manufacturing Company, Racine, WI 53403 USA.

IEEE Log Number 9411190.

[1] Although scaling functions are the ones actually used, the network is still called the wavelet network to emphasis its connection with the theory of orthonormal wavelets.

[2] Zhang and Benveniste mentioned orthogonal wavelets in their paper [9].

properties of the wavelet network. Finally, in experiments we compared the performance of the wavelet network not only with that of the MLP's, as was done in [11], [12], but also with that of the RBF networks.

This paper is organized as follows. After a brief review of the theory of orthogonal wavelets in Section II, the wavelet-based network and a theoretical analysis of its properties are described in Sections III and IV, respectively. Experimental results are described in Section V, followed by a summary in Section VI.

II. Wavelets

Here, we briefly review some results from the theory of wavelets that are relevant to this work. For more comprehensive discussions of wavelets and their applications in signal processing, see e.g., [13]–[16]. For the sake of simplicity, we first describe one-dimensional wavelets.

A. Orthogonal Wavelets

The wavelets we are interested in are functions whose dilations and translations form an orthonormal basis of $L^2(\boldsymbol{R})$, the space of all square integrable functions on $\boldsymbol{R}$. Specifically, there exists a function $\psi(t)$ (the "mother wavelet") such that

$$\psi_{m,n}(t) = 2^{m/2}\psi(2^m t - n) \tag{1}$$

form an orthonormal basis of $L^2(\boldsymbol{R})$. Therefore, the wavelet basis induces a orthogonal decomposition of $L^2(\boldsymbol{R})$

$$L^2(\boldsymbol{R}) = \bigoplus_m \boldsymbol{W}_m \tag{2}$$

where $\boldsymbol{W}_m$ is a subspace spanned by $\{2^{m/2}\psi(2^m t - n)\}_{n=-\infty}^{n=+\infty}$.

The wavelet $\psi(t)$ is often generated from a companion $\varphi(t)$, known as the scaling function (the "father wavelet"), through the following "dilation" equations:

$$\varphi(t) = \sqrt{2}\sum_k h_k \varphi(2t - k) \tag{3a}$$

$$\psi(t) = \sqrt{2}\sum_k g_k \varphi(2t - k) \tag{3b}$$

where $\{h_k\}$ and $\{g_k\}$ are a pair of discrete quadrature-mirror filters (lowpass and highpass) that are related through

$$g_k = (-1)^{k-1} h_{-(k-1)}. \tag{4}$$

The dilations and translations of the scaling function induce a multiresolution analysis (MRA) of $L^2(\boldsymbol{R})$, i.e., a nested chain of closed subspaces

$$\cdots \subset \boldsymbol{V}_{-1} \subset \boldsymbol{V}_0 \subset \boldsymbol{V}_1 \subset \boldsymbol{V}_2 \cdots \tag{5}$$

such that

$$\bigcap_m \boldsymbol{V}_m = \{0\}, \quad \operatorname{clos}\left\{\bigcup_m \boldsymbol{V}_m\right\} = L^2(\boldsymbol{R}) \tag{6}$$

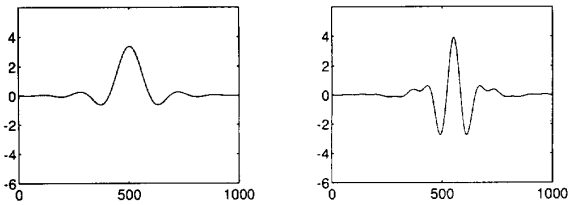


Fig. 1. Examples of Lemarie–Meyer scaling function and wavelet: (a) Scaling function; (b) Wavelet.

where $\boldsymbol{V}_m$ is the subspace spanned by $\{2^{m/2}\varphi(2^m t - n)\}_{n=-\infty}^{n=+\infty}$ or $\{\varphi_{m,n}(t)\}_{n=-\infty}^{n=+\infty}$. In addition, $\boldsymbol{V}_m$ and $\boldsymbol{W}_m$ are related by

$$\boldsymbol{V}_{m+1} = \boldsymbol{V}_m \oplus \boldsymbol{W}_m. \tag{7}$$

A typical wavelet, the Lemarie–Meyer wavelet, and its associated scaling function are shown in Fig. 1.

The above discussions suggest two schemes for decomposing a L^2 function $f(t)$, namely,

$$f(t) = \sum_{m,n} \langle f, \psi_{m,n}\rangle \psi_{m,n}(t) \tag{8a}$$

and

$$f(t) = \sum_n \langle f, \varphi_{m_0,n}\rangle \varphi_{m_0,n}(t) + \sum_{m \geq m_0, n} \langle f, \psi_{m,n}\rangle \psi_{m,n}(t) \tag{8b}$$

where $\langle \cdot, \cdot \rangle$ represents the inner product and m_0 is an arbitrary integer, representing the lowest resolution or scale in the decomposition. What is more important to this work, however, is the fact that any $f(t) \in L^2(\boldsymbol{R})$ can be approximated arbitrarily closely in $\boldsymbol{V}_M$, for some integer M. That is, for any $\epsilon > 0$, there exists an M sufficiently large such that

$$\left\| f(t) - \sum_n \langle f, \varphi_{M,n}\rangle \varphi_{M,n}(t) \right\| < \epsilon \tag{9}$$

where the norm is the L^2 norm.

B. Extension to Multiple Dimensions

All of the one dimensional results described above can be extended to multiple-dimensions, i.e., to $L^2(\boldsymbol{R}^d)$, where $d > 1$ is an integer. One scheme is to generate *separable* scaling functions and wavelets by the tensor products of one-dimensional scaling functions and wavelets [15]. For example, a d-dimensional scaling function can be generated by

$$\varphi_d(\boldsymbol{x}) = \varphi_d(x_1, x_2, \cdots, x_d) = \prod_{j=1}^{d} \varphi(x_j). \tag{10}$$

Later in the paper, the subscript d in (10) will be omitted, wherever there is no confusion.

Another scheme is to generate non-separable scaling functions and wavelets [17]. The extension of the MRA of (5), (6) and the wavelet representations of (8a)–(9) is straightforward.

III. Wavelet Networks

In this section, we first describe the structure and training procedure of the wavelet-based neural network and then present some theoretical results related to its properties. Without loss of generality, we restrict our discussion to multiple-input one-output wavelet networks. Due to the special structure of the wavelet networks, as will be seen in this section, the results are also applicable to multiple-output networks.

A. Wavelet Network

For the sake of simplicity, we first consider the one-dimensional (one-input) case. Let $f(t) \in L^2(\boldsymbol{R})$ be an arbitrary function. The problem of function learning can be stated as: given a set of training data,

$$\boldsymbol{T}_N = \{(t_i, f(t_i)\}_{i=1}^N \tag{11}$$

find an estimate of $f(t)$ that closely approximates $f(t)$. Here, we have assumed that the training data was not corrupted by noise. The extension to the case of noisy training data will be discussed later (in Appendix B). From the theory of wavelets, we know that $f(t)$ can be approximated arbitrarily closely by selecting a sufficiently large M such that

$$f(t) \simeq \sum_k \langle f, \varphi_{M,k}\rangle \varphi_{M,k}(t). \tag{12}$$

In a conventional neural network, each node has the following input-output relationship

$$S_{\text{out}} = \phi\left(\sum_m W_m S_{\text{in},m} + T\right)$$

where $S_{\text{in},m}$ is the mth input, W_m's are the weights, T is a threshold, and $\phi(\cdot)$ is generally a non-linear function, e.g., a sigmoidal function as in the MLP. Hence, the approximation in (12) can be implemented by a three-layer network, as shown in Fig. 2. Specifically, the input layer is just one "all-pass" node with output t. The hidden layer contains a countable number of nodes indexed by k, which runs through all integers. The weights and non-linearities of the hidden-layer nodes are identical, i.e., 2^M and $\varphi(\cdot)$, respectively. The thresholds of hidden-layer nodes, on the other hand, are different and the threshold for the kth node is k. The output layer is one linear node with threshold 0. The weights of the node, ideally, should be the representation coefficients in (12). Since in most practical applications, the functions of interest have finite support, it is possible for the hidden layer to contain only a finite number of nodes. Indeed, this is true when $\varphi(t)$ has compact support, as is the case for Daubechies' wavelets [14]. For all practical purposes, it could also be considered true even when $\varphi(t)$ does not have compact support but is fast-decaying, as is the case for the Lemarie–Meyer scaling functions [15]. Hence, without loss of generality, we assume that $f(t)$ is supported in $[-1/2, 1/2]$ and that the indices of the hidden nodes run from $-K$–K for some positive integer K.

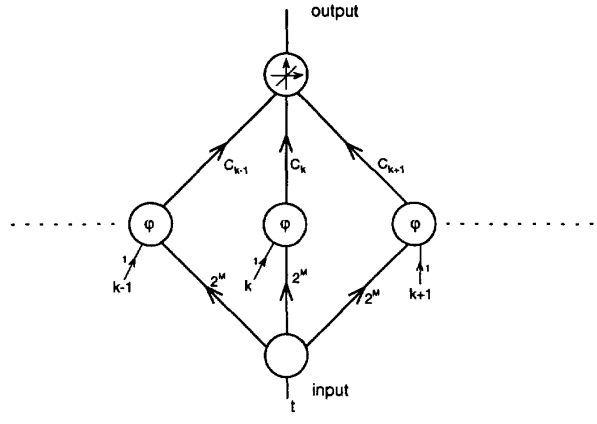


Fig. 2. The 3-layer wavelet network.

For a given set of M and K, the wavelet network described above implements a function $g(t)$

$$g(t) = \sum_{k=-K}^{K} c_k \varphi_{M,k}(t) \tag{13}$$

which could be used to approximate $f(t)$ if the weights c_k are properly selected. When a training data set $\boldsymbol{T}_N$ is available, c_k can be obtained by minimizing the mean square (training) error $e_N(f,g)$, i.e.,

$$(\hat{c}_{-K}, \cdots, \hat{c}_K) = \arg \min_{(c_{-K}, \cdots, c_K)} e_N(f,g) \tag{14a}$$

where

$$e_N(f,g) = \frac{1}{N}\sum_{i=1}^{N}(f(t_i) - g(t_i))^2 \tag{14b}$$

and the subscript N emphasizes the dependence on the training data. Equation (14a) can be solved by taking the partial derivatives of the mean square error of (14b)

$$\frac{\partial e_N(f,g)}{\partial c_k} = 0 \tag{15a}$$

which leads to a set of linear equations. Since the scaling functions are orthogonal, for sufficiently large N (15a) generally has a unique solution. However, when K is large, a direct solution of (15a), which involves the inversion of a $(2K+1)\times(2K+1)$ matrix, may be computation intensive. An alternative to the direct solution is an iterative gradient-descent procedure

$$\hat{c}_k^{(p)} = \hat{c}_k^{(p-1)} - \lambda \left.\frac{\partial e_N(f,g)}{\partial c_k}\right|_{c_k = \hat{c}_k^{(p-1)}} \tag{15b}$$

where λ is a stepsize. In fact, further computation reduction is possible in (15b) once the locality of the scaling function is taken into consideration. Specifically, the derivative in (15b) contains the sum of terms like $\varphi_{M,k}(t_i)$. When $\varphi(t)$ has compact support and t_i's are more or less evenly distributed in $[-1/2, +1/2]$ (i.e., they can be found in the support-intervals of scaling functions with different shifts), for a given k most of these terms will be zero and need not be summed.

Another way to determine the weights of the wavelet network is by simply letting

$$\hat{c}_k = \frac{1}{N}\sum_{i=1}^{N} f(t_i)\varphi_{M,k}(t_i). \tag{16}$$

This would be a good approximation to $\langle f(t), \varphi_{M,k}(t)\rangle$ if the training data is drawn *uniformly*, i.e., t_i's are i.i.d. (independent and identically distributed) random variables uniformly distributed in $[-1/2, 1/2]$. Generally, for a particular set of training samples, this scheme leads to a larger mean square error than that of (14a).

The extension of the wavelet network to the multidimensional case is straightforward. Let $\varphi(\boldsymbol{x})$ be a separable scaling function of the Mallat kind [15] in $L^2(\boldsymbol{R}^d)$ for some $d > 1$ (see (10)). It then induces a multiresolution analysis. As in the 1-D case, we assume for the rest of this section that the functions of interest can be approximated by $\boldsymbol{V}_M$ for some M and have finite supports (e.g., in the hypercube $[-1/2, +1/2]^d$). Then, the 1-D wavelet network of (13) can be extended to

$$g(\boldsymbol{x}) = \sum_{\boldsymbol{k}} c_{\boldsymbol{k}}\varphi_{M,\boldsymbol{k}}(\boldsymbol{x}) \tag{17}$$

where $\boldsymbol{k}$ is an index vector that runs in a finite set. However, there are practical difficulties when d is large, e.g., $d > 3$, as will be described in part C of this Section.

B. Practical Considerations

There are several issues concerning the practical implementation of the wavelet network. The first is how to determine the number of hidden layer nodes.

In the 1-D case, for a given M the number of $\varphi_{M,k}(t)$'s whose centers are inside $[-1/2, +1/2]$ is roughly $2^M + 1$. This stems from the fact that the scaling functions of scale 2^M are shifted versions of $\varphi(2^M t)$ whose shifts are integer multiples of 2^{-M}. Since the scaling functions used in this work either have compact support or are fast-decaying, for all practical purposes, the number of scaling functions with scale 2^M that are needed to "cover" $[-1/2, +1/2]$ or to approximate any L^2 function supported in $[-1/2, +1/2]$ is no more than $2^M + p$, where $p \geq 1$ is a small integer. Often p is selected to be greater than 1 in order to provide some safeguard, as illustrated in Fig. 4. Similarly, for d-dimensional functions, to approximate a function at scale 2^M, the number of hidden layer nodes needed is no greater than $(2^M + p)^d$.

To determine a proper M, we can use the following simple scheme:

1) Start with a small M and use the above bound (i.e., $(2^M + p)^d$) to determine the number of hidden layer nodes.
2) Find the weights of the output layer by using either (14a) or (16). The resulting wavelet network is denoted as $\hat{f}_{M,N}$, where the subscripts M and N indicate the dependence on the scale and training data.
3) Compute the mean square error $e_N(f, \hat{f}_{M,N})$.
4) If the mean square error is smaller than some threshold, stop. Otherwise, increase M by 1 and go back to step 1.

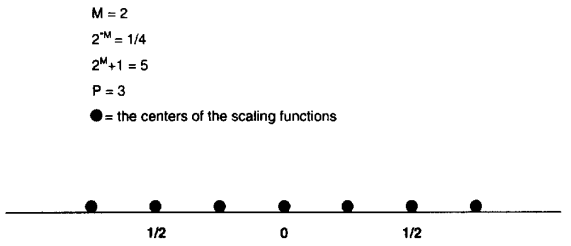


Fig. 3. An illustration of the positions of the scaling functions for a scale M.

An alternative scheme is the multiresolution scheme used by Boubez [12] which could further reduce the number of hidden layer nodes. The idea here is to start with a reasonably small M. Since the mean square (training) error contains contributions from each hidden node, it is possible to find those "bad nodes" whose contributions are above some threshold. Each "bad node" is then replaced by several "finer-resolution" nodes in a way that is similar to the two-scale dilation equation of (3a). The process repeats until no more hidden nodes need to be broken. A variation on this scheme, suggested by a referee, is to select a relatively small M and the corresponding scaling functions and then add in wavelets.

The second practical issue is that the training data are often not uniformly distributed in the support of the functions of interest (e.g., $[-1/2, +1/2]^d$). Rather, they come in the form of clusters. This is especially true for functions of high dimensions. In this case, it would be a waste of computational resources to cover the entire $[-1/2, +1/2]^d$ by scaling functions. Indeed, it would be more sensible to identify regions in $[-1/2, +1/2]^d$ that contain sufficient amount of training data (non-empty regions), in which the training data can be considered more or less as uniform, and to construct a sub wavelet network for each non-empty region. The function of interest can then be approximated by the composite of such sub-networks, as illustrated in Fig. 3. To identify the non-empty regions, we could use any one of a number of popular clustering algorithms (e.g., [22]) as was done in [3], [4].

C. Learning High-Dimensional Functions

Learning a d-dimensional function becomes considerably more difficult when d is large, e.g., $d > 3$, due to two main problems. First, the available training data is generally sparse and do not "fill out" the domain of the function. For example, suppose that to learn a 1-D function defined in [0, 1], 128 uniformly observed training points[3] are needed for adequate resolution. To learn a 10-d function defined on $[0, 1]^{10}$ with comparable resolution, it would require $128^{10} = 2^{70}$ training points! Hence, one generally does not have enough training samples to learn the function "in its entirety." The second difficulty in high-dimensional function learning is the "curse of dimensionality:" as d increases, the convergence rate of a function estimation scheme generally decreases.

Although a general solution to these problems remains to be found, for many applications, good results can be obtained by observing that in the sparse training data $\{\boldsymbol{x}_i, f(\boldsymbol{x}_i)\}, \{\boldsymbol{x}_i\}$ are

[3]Here, a training point is a pair such as $(x_i, f(x_i))$.

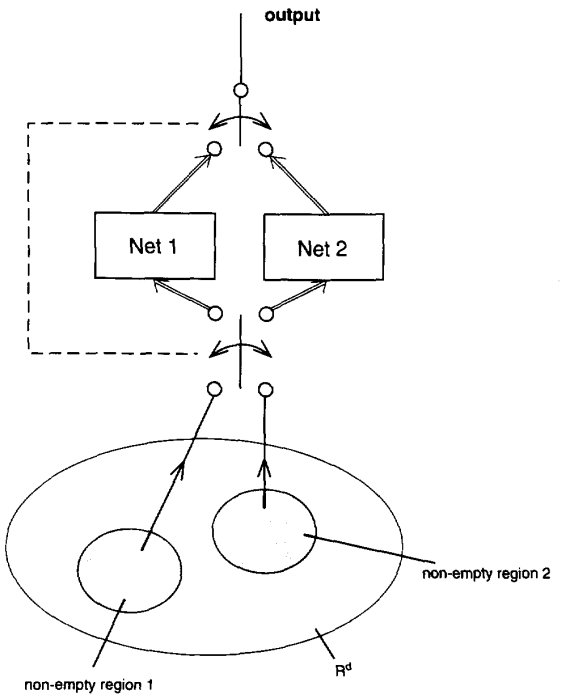


Fig. 4. A composite network.

"concentrated" in one or a small number of regions in $[0,1]^d$ whose dimensions are much less than d. Here, we assumed that the function in question is defined in $[0,1]^d$. To be more precise, in such cases the (high) d-dimensional function, $f(\boldsymbol{x})$, can be approximated by

$$f(\boldsymbol{x}) = \sum_{k=1}^{K} c_k f_k(\boldsymbol{P}_k \boldsymbol{x}) \tag{18}$$

where $f_k(\cdot)$ is a function defined for the kth region, c_k is its "weight," and $\boldsymbol{P}_k$ is the corresponding projection operator and a $d_1 \times d$ matrix, where $d_1 \ll d$. It is interesting to notice that a three-layer MLP can also be described by (18) when $f_k(\cdot)$ is the shifted sigmoid of the kth hidden-layer node and $\boldsymbol{P}_k$ is a row vector (i.e., $d_1 = 1$).

There are two ways of applying (18) in our wavelet network. The first scheme is related to the classical principle-component analysis regression (PCAR) [24]. Specifically, one selects

- $K = 1$ and $c_k = 1$,
- $\boldsymbol{P}_1$ to be a matrix whose row vectors are the significant eigenvectors of the covariance matrix of $\{\boldsymbol{x}_i\}$,
- $f_1(\cdot)$ be the wavelet network of (13).

This scheme works well when $\{\boldsymbol{x}_i\}$ are nearly Gaussian distributed (with correlated vector components) in a single region or several non-overlap regions. The second and more elaborate scheme is to use projection pursuit regression (PPR) [24], in which $c_k, f_k(\cdot)$ and $\boldsymbol{P}_k$ are optimized to minimize the training error. In particular, $f_k(\cdot)$ is expanded by scaling functions as is in (13) and hence (18) remains a wavelet network. The experiment results for high-dimensional function learning in this paper was generated by the simpler PCAR scheme while the more involved PPR scheme is currently under investigation.

During the revision of this paper, we came across a recent and interesting paper by Hwang *et al.* [25] in which PPR was used for function learning in the context of neural networks. In their approach, $f_k(\cdot)$ was expanded by Hermite polynomials which are an orthonormal basis but do not have local supports. Hwang *et al.* obtained good experimental results for two-dimensional continuous functions. Hence, we are optimistic about the wavelet-based PPR since wavelets are known to be better function estimators than polynomials (better local accuracy and faster convergence, see e.g., [26], [27]). Finally, we notice that Hwang *et al.* stressed the need to test their PPR approach for high-dimensional functions.

IV. Properties of the Wavelet Network

In this subsection, we describe some theoretical results concerning the learning capability of the wavelet networks, i.e., how good they are as function estimators. These results include approximation ability and consistency, as well as the associated convergence rates.

The approximation ability of a neural network is concerned with the following question: what kind of functions can the neural network learn? To address this question, it is helpful to think of the neural network as a function approximation scheme (i.e., it approximates the function to be learned). A function approximation scheme, according to the definition given by Xu *et al.* [8], is a set of functions $\mathcal{F}$ defined on $\boldsymbol{R}^d$, where

$$\mathcal{F} = \bigcup_n \mathcal{F}_n$$

where $\mathcal{F}_n$'s are subsets of functions. For example, for the wavelet network, $\mathcal{F}_n$ is the set of all wavelet networks constructed using scaling functions with scale $n = 2^M$. $\mathcal{F}$ is said to possess the property of *universal approximation* if it is dense in $C[\boldsymbol{U}]$, the space of continuous functions supported on a compact subset $\boldsymbol{U} \subset \boldsymbol{R}^d$. That is, for any $f \in C[\boldsymbol{U}]$, there exists a sequence $f_n, f_n \in \mathcal{F}_n$, such that $f_n \to f$ *uniformly*. If the convergence (and therefore, the sense of dense) is L^2 rather than uniform, $\mathcal{F}$ is said to have the property of L^2 approximation.

By construction, it is easy to see (e.g., from (9)) that the wavelet network has the L^2 approximation property, i.e., it can learn any L^2 function with finite support. The universal approximation property, on the other hand, requires that φ be regular. Details are provided in Appendix A. To summarize, we have the following:

Theorem 1:

The wavelet network has the properties of universal approximation and L^2 approximation.

In universal and L^2 approximation properties, it is also of interest to know how fast the sequence f_n converges or how fast the approximation error decreases. By using the relationship between delta sequences and reproducing kernel of the scaling functions [19], we can show the following:

Theorem 2:

For the wavelet network, the rates of convergence in the universal and L^2 approximation properties can be made arbitrarily rapid in the following sense: for any $\alpha > 0$, there is a Sobolev space $\boldsymbol{H}_\beta$ such that for any $f \in \boldsymbol{H}_\beta$, there exists a sequence of wavelet networks f_n, where $n = 2^M$, such that

$$||f - f_n||_u = O(n^{-\alpha}) \tag{19a}$$

and

$$||f - f_n||_{L^2} = O(n^{-\alpha}). \tag{19b}$$

Here $||\cdot||_u$ and $||\cdot||_{L^2}$ are the sup and L^2 norms, respectively.

The proof of Theorem 2 is given in Appendix A. Notice that in this theorem, we require that f be smooth (i.e., in a Sobolev space). The degree of smoothness, β, generally depends on α and d. The larger α and d are, the larger β is required to be. This smoothness requirement allows Theorem 2 to avoid the "curse of dimensionality" in that, as long as the function is smooth enough, the convergence rates above are independent of d.

A function estimator, such as the wavelet network, is said to be *consistent* if it converges to the function to be estimated as the number of training samples approaches infinity. Let the function to be estimated be denoted by $f(\boldsymbol{x}) \in L^2(\boldsymbol{R}^d)$. Assume, for the moment, that $f(\boldsymbol{x}) \in \boldsymbol{V}_M$ for some sufficiently large M. Then,

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{k}} \langle f, \phi_{M,\boldsymbol{k}} \rangle \varphi_{M,\boldsymbol{k}}(\boldsymbol{x}). \tag{20a}$$

Let $\hat{f}_{M,N}(\boldsymbol{x})$ be a wavelet network estimate of $f(\boldsymbol{x})$ from training data. Then,

$$\hat{f}_{M,N}(\boldsymbol{x}) = \sum_{\boldsymbol{k}} \hat{c}_{N,\boldsymbol{k}} \varphi_{M,\boldsymbol{k}}(\boldsymbol{x}) \tag{20b}$$

where the coefficients are obtained through the d-dimensional extension of either (14a) or (16). For a given sequence of training sets $\boldsymbol{T}_N$, the wavelet network $\hat{f}_{M,N}$ is said to be L^2 consistent if $||f - \hat{f}_{M,N}|| \to 0$ as $N \to \infty$, where the norm is the L^2 norm. Generally, since the training samples are obtained through random sampling or measurement, $\boldsymbol{T}_N$ is a set of random variables. Hence, the L^2 convergence has to be in some well-defined probabilistic sense, e.g., almost surely, mean square, in probability, in distribution, etc. In this work, we consider the mean square sense. That is, $\hat{f}_{M,N}$ is consistent if

$$\lim_{N\to\infty} E[||f - \hat{f}_{M,N}||^2] = 0. \tag{21}$$

But

$$\begin{aligned} E[||f - \hat{f}_{M,N}||^2] &= E\left[\int_{\boldsymbol{R}^d} (f(\boldsymbol{x}) - \hat{f}_{M,N}(\boldsymbol{x}))^2 \, d\boldsymbol{x}\right] \\ &= E\left[\sum_{\boldsymbol{k}} (\langle f, \varphi_{M,\boldsymbol{k}} \rangle - \hat{c}_{N,\boldsymbol{k}})^2\right] \\ &= \sum_{\boldsymbol{k}} E[(\langle f, \varphi_{M,\boldsymbol{k}} \rangle - \hat{c}_{N,\boldsymbol{k}})^2] \\ &= \sum_{\boldsymbol{k}} E[(c_{\boldsymbol{k}}^{(0)} - \hat{c}_{N,\boldsymbol{k}})^2] \end{aligned} \tag{22}$$

where $c_{\boldsymbol{k}}^{(0)} = \langle f, \varphi_{M,\boldsymbol{k}} \rangle$. Hence, if for all $\boldsymbol{k}$, $\hat{c}_{N,\boldsymbol{k}} \to c_{\boldsymbol{k}}^{(0)}$ as $N \to \infty$ in the mean square sense, the wavelet network is consistent. Indeed, we have the following:

Theorem 3:

Assume that in the training data set

$$\boldsymbol{T}_N = \{(\boldsymbol{x}_1, f(\boldsymbol{x}_i))\}_{i=1}^N$$

$\boldsymbol{x}_i, i = 1, \cdots, N$, are i.i.d. and uniformly distributed. Assume that the weights of the wavelet network of (20b), denoted by $\hat{c}_{N,\boldsymbol{k}}$, are obtained by either (14a) or (16). Then, for any $\boldsymbol{k}$ in the wavelet network of (20b)

$$\lim_{N\to\infty} \hat{c}_{N,\boldsymbol{k}} = c_{\boldsymbol{k}}^{(0)} \tag{23}$$

in the mean square sense and hence, the wavelet network of (20b) is L^2 consistent in the mean square sense. Furthermore, the rate of convergence in (23) is $O(1/N)$.

The proof of Theorem 3 is given in Appendix B. When $f \in L^2(\boldsymbol{R}^d)$, but $f \notin \boldsymbol{V}_M$ for any M, we can let $N > n = 2^M$. Then, by using Theorems 1 and 3, the consistency result follows when we let $n \to \infty$. Specifically, according to Theorem 1, there exists a sequence $f_n \in \boldsymbol{V}_M$ and $f_n \to f$ as $n \to \infty$. Similarly, according to Theorem 3, there exists a sequence $\hat{f}_{M,N}$ and $\hat{f}_{M,N} \to f_n$ as $N \to \infty$. Thus for sufficiently large n and N, we have

$$E[||f - \hat{f}_{M,N}||^2] \le ||f - f_n||^2 + E[||f_n - \hat{f}_{M,N}||^2] \le 2\epsilon. \tag{24}$$

That is, $\hat{f}_{M,N} \to f$ in the mean square sense.

D. Relation to RBF and Wavelet Frame Networks

The function represented by a d-input and one-output RBF network can be represented as

$$g_n(\boldsymbol{x}) = \sum_{k=1}^{n} c_k \phi(||\boldsymbol{x} - \boldsymbol{r}_k||_{\Sigma^{-1}_k}) \tag{25a}$$

where

$$||\boldsymbol{x} - \boldsymbol{r}_k||_{\Sigma_k^{-1}} = (\boldsymbol{x} - \boldsymbol{r}_k)^t \Sigma_k^{-1} (\boldsymbol{x} - \boldsymbol{r}_k). \tag{25b}$$

Here, $\phi(\cdot)$ is a localized basis function, e.g., a Gaussian function, and $\boldsymbol{r}_k$ and Σ_k^{-1} are the center and covariance matrix (spread).

For a given set of training data, the parameters of an RBF network, namely, $c_k, \boldsymbol{r}_k$ and Σ_k^{-1}, can be found by minimizing the training error $e_N(f, g_n)$. However, due to the non-linear nature of $\phi(\cdot)$, the minimization often gets stuck in an undesirable local minimum which results in useless parameter estimates. One practical solution to this problem is to estimate $\boldsymbol{r}_k$ and Σ_k^{-1} first by performing some sort of clustering procedure on the training data [3], [4]. Then, c_k's are found by minimizing the training error. This leads to a set of linear equations, similar to (15a) for the wavelet network.

However, unlike the case for the wavelet network, this set of linear equations usually does not have a unique solution and therefore, a least squares procedure may have to be used. As a result, the c_k's found in this way, do not necessarily minimize the training error (although sometimes they may be good enough for approximating a given function).

When a wavelet frame (non-orthogonal wavelets) replaces the the radial basis functions in an RBF network, $\mathbf{r}_k$ and Σ_k^{-1} are replaced by the shifts and scales of the wavelets. For a wavelet frame network, it is possible to determine the shifts and scales by using the theory of wavelets rather than by the more heuristic clustering procedures. On the other hand, due to the redundancy in the wavelet frame, the set of linear equations for c_k may still require a least squares solution, as is the case for the RBF networks. The orthogonal wavelet network not only has the advantage of the wavelet frame network, the equation for solving its weights (15a) also has a unique solution. Furthermore, it provides an efficient representation or approximation of a function due to the orthogonality of the scaling functions.

It is interesting to compare the theoretical results for the wavelet network with those for the RBF network. Both types of networks have universal and L^2 approximation properties.[4] As for convergence rates, we can compare the results in Theorem 2 with those obtained for the RBF network by Corradi and White [6], Girosi and Anzellotti [7], and Xu *et al.* [8]. The convergence rates obtained by Girosi and Anzellotti for the uniform and L^2 convergence are both $O(n^{-1/2})$. These rates are independent of dimensions d provided that some smoothness conditions similar to those of Theorem 2 are satisfied. Corradi and White and Xu *et al.* obtained similar rates for L^2 convergence which are $O(n^{-(1/1+d/2q)})$, where q is the number of derivatives that f has. These rates become *approximately* independent of d by for very large q. Compared to these results, the rates in Theorem 2 are somewhat better in that the exponent α can be made arbitrarily large. Notice here that the n in the RBF network and the n in the wavelet network are closely related. In the RBF network, n is the number of hidden-layer nodes, while in the wavelet network n is the scale of the scaling functions. However, when the scheme of Section III.B for determining the number of hidden nodes is used, the number of nodes equals $2^M + p = n + p$.

As for statistical consistency, Xu *et al.* [8] have shown that the RBF network is L^2 and pointwise consistent in the sense of almost surely, in probability, and completely. We have shown in this work (Theorem 3 and the remarks that follow) that the wavelet network is L^2 consistent in the mean square sense which implies in probability.

V. Experimental Results

The wavelet neural network described in Section III was tested in function learning simulations where it was also compared with the MLP and RBF networks of similar complexity. Experiments have been performed for learning both low and high-dimensional functions, as described below.

[4]For the universal and L^2 approximation properties of the RBF networks, see e.g., Xu *et al.* [8].

A. Learning One and Two-dimensional Functions

Typical results for 1-D and 2-D functions are shown in Figs. 5–7. In Fig. 5, the wavelet network was used to learn two 1-D functions from i.i.d. uniformly distributed training data. The two functions are defined on $[-1, +1]$. The number of hidden nodes was determined to be 41 by using the procedure described in Section III-B. Here, the large number of hidden nodes, or fine scale, is due to the jumps and transitions (local high-frequency components) in the functions. This is consistent with the theory of wavelets. The output layer weights were determined by using the iterative solution of (15b) rather than by inverting a 41×41 matrix. The amount of computation is further reduced by noticing that the scaling functions used (namely, the Lemarie–Meyer scaling functions), have, for all practical purposes, compact support, as described in Section III-A. For both functions, the training process converged within 10 iterations.[5] As shown in Fig. 5, the wavelet network captured the shapes of the functions accurately except with some small ripples. These ripples can be attributed to the Gibbs' phenomenon that exists in wavelet representations (similar to that in Fourier series representations). Indeed, it can be shown that Gibbs' phenomenon exists for the Lemarie–Meyer and the Daubechies wavelets, but not for the Haar wavelets [19]. However, the Lemarie–Meyer and Daubechies wavelets have many desirable properties that the Haar wavelets do not have (e.g., convergence properties and compactly supported in frequency domain) and are more widely used. Hence, the Haar wavelets (or scaling functions) are not used in our experiments.

For comparison, we have also shown in Fig. 5 the results obtained by a 3-layer MLP (one hidden layer) with 41 hidden layer nodes. This particular MLP was chosen so that it has comparable complexity to the wavelet network. After 300 iterations of training, the MPL captured the global shapes of the functions, but not the transitions in the functions. More training iterations (up to 10 000) did not lead to significant improvement. It may be possible to improve the results by using more hidden layers; however, this will make the network more complex and longer to train.

Similar comparative experiments were also performed for the RBF network. The Gaussian function, which is a popular choice for RBF networks, was used as the basis function. Since the training data has a uniform distribution, clustering procedures would be ineffective in deciding the number of hidden layer nodes. To keep the complexity of the Gaussian RBF network comparable to the wavelet network, the number of hidden layer nodes was selected to be 41 (we also tried smaller numbers of hidden layer nodes, e.g., 21 and 31, but 41 seemed to provide the best results). To illustrate how the centers and variances of the Gaussian basis functions affect the learning results, we performed the following training experiments:

1) Minimizing the training error with respect to the centers and variances of the Gaussian functions as well as the output layer weights,

[5]The number of iterations could be further reduced if the conjugate gradient method [23] is used since the training error surface is quadratic.

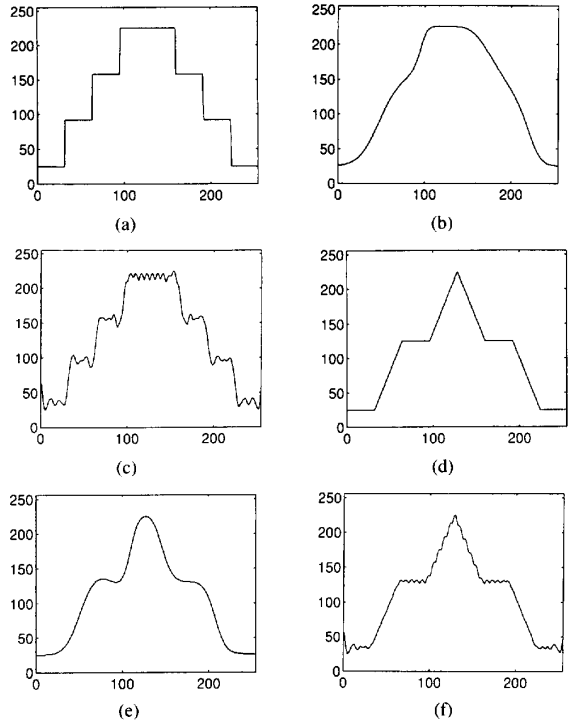


Fig. 5. One-dimensional function learning by the wavelet network and the MLP: Function 1; (b) MLP; (c) wavelet network; (d) Function 2; (e) MLP; (f) wavelet network.

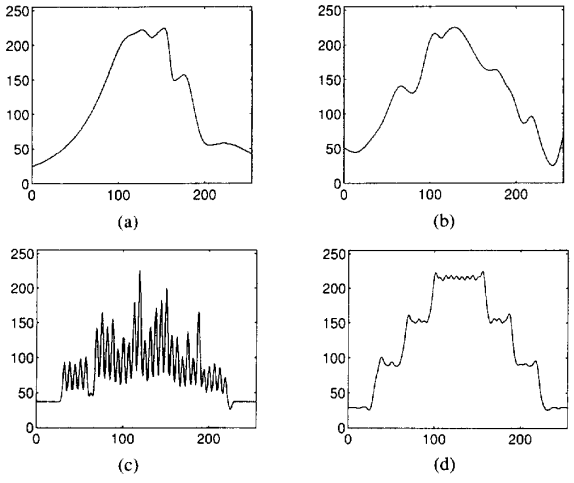


Fig. 6. One-dimensional function learning by the Gaussian RBF network: (a) Nothing fixed; (b) cent. fixed; (c) cent. and var. fixed, $s = 2$; (d) cent. and var. fixed, $s = 1$.

2) Fixing the centers of the Gaussian functions while minimizing the training error with respect to the variances of the Gaussian functions and the output layer weights,
3) Fixing the centers and variances of the Gaussian functions while minimizing the training error with respect to the output layer weights.

In 2 and 3 the centers of the Gaussian functions were fixed such that they coincide with the centers of the scaling functions of the wavelet network. In 3. the variances of the Gaussian functions were fixed to be σ^2 such that neighboring Gaussians overlap by $s\sigma$, where s is a positive number. Notice that as σ increases, so does s since the number of Gaussian functions was fixed to be 41. In all three experiments, the parameters that were not fixed were estimated from training data by a gradient descent algorithm.

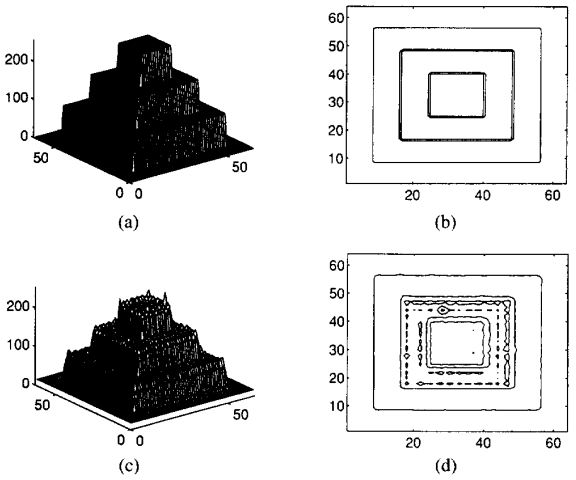


Fig. 7. Two-dimensional function learning by the wavelet network: (a) 2-D function; (b) cont. map of (a); (c) wavelet net. approximation; (d) cont. map of (c).

Experiments were performed for the Gaussian RBF network according to the setup described above for both functions of Fig. 5. Since the results are similar in nature for both functions, only those for the staircase function are shown in Fig. 6. It can be seen that only after both the centers and variances of the Gaussians were set properly, could the network perform as well as the wavelet network. However, unlike the centers and scales for the scaling functions in the wavelet network, the centers and variances of the Gaussian basis functions have to be determined heuristically by trial and error (indeed, the structure of the wavelet network was used as a guide).

The wavelet network was also tested in learning 2-D functions with typical results shown in Fig. 7. Here, the 2-D function is a simple extension of the 1-D staircase function and the wavelet network, as is in the 1-D case, performed well except for some small ripples. The number of iterations is 30.

B. Learning 10-dimensional Functions

The PCAR scheme described in Section III.C was used in the wavelet network to learn high-dimensional functions, with typical results shown in Figs. 8 and 9. The function to be learned in Fig. 8 is a 10-dimensional staircase function whose training data is generated as follows. First, 200 random 10-dimensional vectors, $\boldsymbol{x}_i, i = 1, 2, \cdots, 200$, were generated. For each $\boldsymbol{x}_i, x_{i1}$ is uniformly distributed in $[-1, 1]$ while $x_{ik}, 2 \leq k \leq 10$ are Gaussian distributed with zero-mean and variance 0.5, $f(\boldsymbol{x}_i)$ takes the value of one of the levels of the stairs depending on where $\boldsymbol{x}_i$ is. The training data set is shown in Fig. 8(a). Since we can not show all the 10 dimensions, only $\{(x_{i1}, x_{i2}, f(\boldsymbol{x}_i)\}$ are shown. This is similar to the scatter diagram frequently used in visualizing high-dimensional data

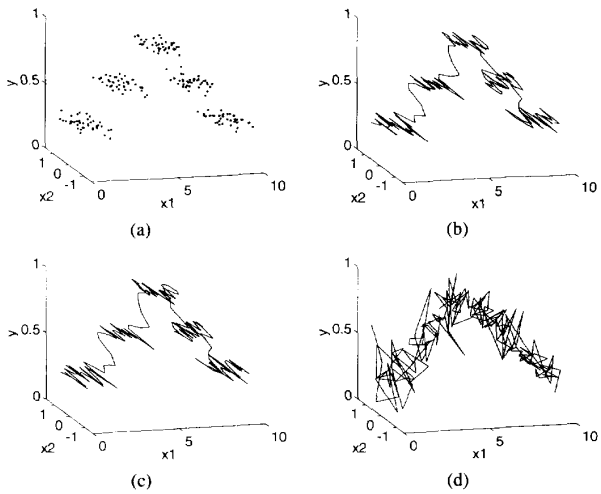


Fig. 8. Learning a 10-D staircase: (a) Training data; (b) wavelet net; (c) Gaussian RBF net; (d) MLP.

[24]. Function learning was performed with

$$\hat{f}(\boldsymbol{x}) = f_{\text{wn}}(\boldsymbol{p}^t \boldsymbol{x}) \tag{26}$$

where $\boldsymbol{p}$ is the most significant eigenvector of the covariance matrix of $\{\boldsymbol{x}_i\}$ and $f_{\text{wn}}(\cdot)$ is the wavelet network. The learned functions by the wavelet network, the Gaussian RBF network (guided by the parameters of the wavelet network, see Section V-A) and the MLP, all with 41 hidden-layer nodes, are shown in Fig. 8(b)–(d), respectively.

In Fig. 8(b)–(d), we have only shown $\{(\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, \hat{f}(\boldsymbol{x}_i))\}$ (other scatter diagrams are similar in nature) and lines are used to connect data points to help to visualize the surface corresponding to the learned functions. Clearly, for this function, the wavelet network and Gaussian RBF performed better than the MLP which was able only to capture the more global features of the function due to discontinuities in the function.

Unlike that in Fig. 8, the function to be learned in Fig. 9 is a continuous function. The 500 training data points are generated in a similar way to those of Fig. 8. In particular, for each $\boldsymbol{x}_i, x_{i1}$ is normally distributed in [0, 6] with mean 3 and variance 10 while $x_{ik}, 2 \leq k \leq 10$ are Gaussian distributed with zero mean and variance 0.1. The $\boldsymbol{x}_i$'s so generated were then rotated 10 times along each axis, each time by 45°. Fig. 9(a) shows the scatter diagram of $\{(x_{i1}, x_{i2}, f(\boldsymbol{x}_i))\}$ where $f(\cdot)$ follows a sinusoidal function. The learned functions are shown in Fig. 9(b)–(d) and in this case, due to the smoothness of the function, all networks performed well. All the networks have 28 hidden-layer nodes.

VI. Summary

In this paper, we described a wavelet-based neural network for learning functions from training data. The structure of this network is similar to that of the RBF network, except that here the radial basis functions are replaced by orthonormal scaling functions that are not necessarily radial-symmetric. The efficacy of this type of network in function learning is demonstrated through theoretical analysis of their asymptotic properties and experimental results. In particular, we have shown in the theoretical analysis that the wavelet network has universal and L^2 approximation properties and is a consistent function estimator. We have also obtained the convergence rates associated with these properties. In the experiments, the wavelet network performed well and compared favorably with the MLP and RBF networks. Compared to MLP networks with similar complexity, it is better at approximating functions with discontinuities and its training requires much less computation. Compared to RBF and wavelet frame based networks, its parameters (e.g., number of hidden nodes and weights) are easier to determine. Furthermore, the wavelet network has somewhat better convergence rates. There is, however, a shortcoming of the wavelet network, which is also shared by the RBF network. That is, when the number of input d is large, many hidden layer nodes are needed. This shortcoming can be overcome by projective data reduction techniques, such as PCAR and PPR described in Section III-C, and by using a multiresolution scheme similar to that of Boubez and Peskin [11], [12]. On the other hand, compared to the MPL networks, this added complexity is what makes the solution for the weights simpler, the convergence faster, and the function approximation more accurate for the wavelet network.

For future work, it would be of interest to investigate the PPR wavelet networks proposed in Section III.C and networks based on the wavelet packets [20].

Appendix A

In this appendix, we prove first the universal approximation property of wavelet network (Theorem 1). This has already been done for $d = 1$ in [21]. It involves expressing the right side of (12) in the form of an integral

$$\begin{aligned} f_M(t) &= \sum_k \langle f, \varphi_{M,k} \rangle \varphi_{M,k}(t) \\ &= \int_{\boldsymbol{R}} \left[\sum_k \varphi_{M,k}(x) \varphi_{M,k}(t) \right] f(x)\, dx \\ &= \int_{\boldsymbol{R}} q_M(x,t) f(x)\, dx \end{aligned} \tag{A1}$$

where $q_M(x,t)$ is the *reproducing kernel* [15] of $\boldsymbol{V}_M$. Notice here, $f_M(t)$ could also be written as $f_{2^M}(t)$ to emphasize the dilation of the scaling functions and this is actually more precise. However, we use $f_M(t)$ to simplify notations. It was shown in [21] that for $d = 1, q_M(x,t)$ is a quasi-positive delta-sequence, i.e., a sequence that converges to $\delta(x-t)$ and satisfies certain regularity conditions (see conditions (i)–(iii) below). It then follows that

$$\sup_{t \in \boldsymbol{R}} |f_M(t) - f(t)| \to 0, \qquad \text{as } M \to \infty \tag{A2}$$

for continuous $f(t)$.

If $d > 1$ and $\varphi(\boldsymbol{x})$ is separable, i.e., $\varphi(\boldsymbol{x}) = \Pi_{j=1}^d \varphi(x_j)$, then so is $q_M(\boldsymbol{x}, \boldsymbol{t})$,

$$q_M(\boldsymbol{x}, \boldsymbol{t}) = \prod_{j=1}^d q_M(x_j, t_j). \tag{A3}$$

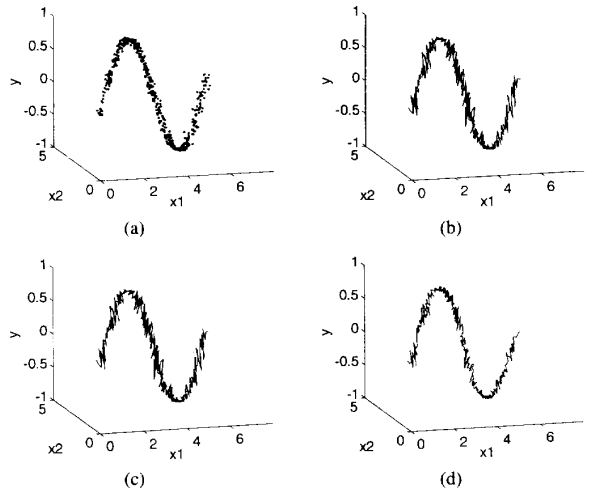


Fig. 9. Learning a 10-D continuous function: (a) Training data; (b) wavelet net; (c) Gaussian RBF net; (d) MLP.

Thus

$$f_M(\boldsymbol{t}) = \int_{\boldsymbol{R}^d} q_M(\boldsymbol{x},\boldsymbol{t}) f(\boldsymbol{x})\, d\boldsymbol{x})$$

and $q_M(\boldsymbol{x},\boldsymbol{t})$ is a quasi-positive delta sequence on $\boldsymbol{R}^d$. This means that $q_M(\boldsymbol{x},\boldsymbol{t}) \in L^1(\boldsymbol{R}^d)$ as a function of $\boldsymbol{x}$ for each $\boldsymbol{t} \in \boldsymbol{R}^d$ and

(i) $\int_{\boldsymbol{R}^d} |q_M(\boldsymbol{x},\boldsymbol{t})|\, d\boldsymbol{x} \le C, \boldsymbol{t} \in \boldsymbol{R}^d, M \in \boldsymbol{Z}$;

(ii) there is a $b > 0$ such that

$$\int_{||\boldsymbol{x}-\boldsymbol{t}||\le b} q_M(\boldsymbol{x},\boldsymbol{t})\, d\boldsymbol{x} \to 1, \qquad \text{as } M \to \infty$$

uniformly on a bounded subset of $\boldsymbol{R}^d$;

(iii) For each $\gamma > 0$

$$\sup_{||\boldsymbol{x}-\boldsymbol{t}||\ge\gamma} |q_M(\boldsymbol{x},\boldsymbol{t})| \to 0, \qquad \text{as } M \to \infty.$$

The same argument used in [21] for the justification of the 1-D version of these conditions carries over to $\boldsymbol{R}^d$. From this we can prove the uniform convergence for f continuous on $\boldsymbol{R}^d$ with compact support. Indeed, we have

$$\begin{aligned} f_M(\boldsymbol{t}) - f(\boldsymbol{t}) &= f(\boldsymbol{t})\left[\int_{||\boldsymbol{x}-\boldsymbol{t}||\le\gamma} q_M(\boldsymbol{x},\boldsymbol{t})\, d\boldsymbol{x} - 1\right] \\ &\quad + \int_{||\boldsymbol{x}-\boldsymbol{t}||\le\gamma} q_M(\boldsymbol{x},\boldsymbol{t})[f(\boldsymbol{x}) - f(\boldsymbol{t})]\, d\boldsymbol{x} \\ &\quad + \int_{||\boldsymbol{x}-\boldsymbol{t}||>\gamma} q_M(\boldsymbol{x},\boldsymbol{t}) f(\boldsymbol{x})\, d\boldsymbol{x} \\ &= I_1 + I_2 + I_3. \end{aligned} \tag{A4}$$

We now choose $0 < \gamma < b$ such that $|f(\boldsymbol{x}) - f(\boldsymbol{t})| < \epsilon$, whenever $||\boldsymbol{x} - \boldsymbol{t}|| < \gamma$. Then, I_2 satisfies

$$|I_2| \le \epsilon \int_{||\boldsymbol{x}-\boldsymbol{t}||\le\gamma} |q_M(\boldsymbol{x},\boldsymbol{t})|\, d\boldsymbol{x} \le \epsilon C \tag{A5}$$

by (i). Similarly, I_3 satisfies, for $M \ge M_1$,

$$|I_3| \le \sup_{||\boldsymbol{x}-\boldsymbol{t}||\ge\gamma} |q_M(\boldsymbol{x},\boldsymbol{t})| \cdot ||f||_1 < \epsilon ||f||_1.$$

Finally, we choose $M_2 \ge M_1$ so large that

$$\left| \int_{||\boldsymbol{x}-\boldsymbol{t}||\le\gamma} q_M(\boldsymbol{x},\boldsymbol{t})\, d\boldsymbol{x} - 1 \right| < \epsilon \tag{A6}$$

for $M > M_2$. Then, I_1 satisfies

$$|I_1| \le \sup_{\boldsymbol{t}\in\boldsymbol{R}^d} \epsilon |f(\boldsymbol{t})| = \epsilon ||f||_\infty \tag{A7}$$

Then for $M \ge M_2$, (A5)–(A7) hold and hence

$$|f_M(\boldsymbol{t}) - f(\boldsymbol{t})| \le \epsilon [C + ||f||_1 + ||f||_\infty]$$

which gives us our uniform convergence (universal approximation property).

In order to prove Theorem 2, we use the corresponding result of [18] for $d = 1$. This says that for φ satisfying certain conditions, the uniform rate of convergence of f_M to f is given by

$$\sup_{t\in\boldsymbol{R}} |f_M(t) - f(t)| = O(2^{-M(\beta-(1/2))}) \tag{A8}$$

for $f \in \boldsymbol{H}^\beta$, where $\beta > 1/2$. The requirement on φ is that it be r-regular [14] and $\int \varphi(t) t^k\, dt = 0$, for $k = 1, 2, \cdots, r$, where $r > \beta - 1/2$.

This result in turn is based on the fact that the delta sequence $\{q_M(x,t)\}$ satisfies [18]

$$||q_M(\cdot,t) - \delta(\cdot,t)||_{-\alpha} = O(2^{-M(\alpha-(1/2))}) \tag{A9}$$

uniformly for $t \in \boldsymbol{R}$ and $r > \alpha - 1/2$. Here the norm is that of the negative order Sobolev space $\boldsymbol{H}^{-\alpha}$. If an inequality similar to (A9) holds in higher dimensions, a result similar to (A8) will also.

The Sobolev norm in $\boldsymbol{R}^d$ is given by

$$||f||^2_{-\alpha,d} = \int_{\boldsymbol{R}^d} \frac{|\hat{f}(\boldsymbol{\omega})|^2}{(||\boldsymbol{\omega}||^2 + 1)^\alpha}\, d\boldsymbol{\omega}$$

where $\hat{f}(\boldsymbol{\omega})$ is the Fourier transform of $f(\boldsymbol{t})$. For $d = 2$, this is just

$$||f||^2_{-\alpha,2} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \frac{|\hat{f}(\omega_1,\omega_2)|^2}{(\omega_1^2 + \omega_2^2 + 1)^\alpha}\, d\omega_1\, d\omega_2.$$

We need the following inequality in order to reduce the d-dimensional case to 1-dimension

$$\begin{aligned} &\omega_1^2 + \omega_2^2 + \cdots + \omega_d^2 + d \\ &\quad \ge [(\omega_1^2 + 1)(\omega_2^2 + 1)\cdots(\omega_d^2 + 1)]^{1/d}. \end{aligned} \tag{A10}$$

This is obvious by induction since

$$\begin{aligned} &(\omega_1^2 + \omega_2^2 + \cdots + \omega_d^2 + d)^2 \\ &\quad \ge (\omega_1^2 + \omega_2^2 + \cdots + \omega_{d-1}^2 + d - 1)(\omega_d^2 + 1) \\ &\quad \ge [(\omega_1^2 + 1)(\omega_2^2 + 1)\cdots(\omega_{d-1}^2 + 1)]^{1/d-1}(\omega_d^2 + 1) \end{aligned}$$

and hence by the same argument

$$(\omega_1^2 + \omega_2^2 + \cdots \omega_d^2 + d)^d \ge \prod_{i\ne j} (\omega_i^2 + 1)^{d/2(d-1)} (\omega_j^2 + 1)^{d/2}$$
$$j = 1, 2, \cdots, d.$$

The same inequality holds for the geometric mean of the right hand side

$$\left[\prod_{i=1}^d (\omega_i^2 + 1)^{(d(d-1)/2(d-1))+(d/2)}\right]^{1/d} = \prod_{i=1}^d (\omega_i^2 + 1).$$

Thus, there is a constant $c_d > 0$ such that

$$||\boldsymbol{\omega}||^2 + 1 \geq c_d \prod_{i=1}^{d} (\omega_i^2 + 1)^{1/d}.$$

Hence if $f(\boldsymbol{t})$ is separable, so is $\hat{f}(\boldsymbol{\omega}) = \hat{f}_1(\omega_1)\hat{f}_2(\omega_2)\cdot\hat{f}_d(\omega_d)$ and

$$\begin{aligned}||f||^2_{-\alpha,d} &= \int_{-\infty}^{+\infty}\cdots\int_{-\infty}^{+\infty} \frac{|\hat{f}_1(\omega_1)\hat{f}_2(\omega_2)\cdots\hat{f}_d(\omega_d)|^2}{(||\boldsymbol{\omega}||^2+1)^{\alpha}} \\ &\quad d\omega_1\, d\omega_2 \cdots d\omega_d \\ &\leq \frac{1}{c_d}\int_{-\infty}^{+\infty} \frac{|\hat{f}_1(\omega_1)|^2}{(\omega_1^2+1)^{\alpha/d}}\, d\omega_1 \\ &\quad \cdot \int_{-\infty}^{+\infty} \frac{|\hat{f}_2(\omega_2)|^2}{(\omega_2^2+1)^{\alpha/d}}\, d\omega_2 \cdots \\ &\quad \cdot \int_{-\infty}^{+\infty} \frac{|\hat{f}_d(\omega_d)|^2}{(\omega_d^2+1)^{\alpha/d}}\, d\omega_d \\ &= \frac{1}{c_d}\prod_{i=1}^{d} ||f_i||^2_{-\alpha/d,1}. \end{aligned} \tag{A11}$$

We now return to the d-dimensional version of (A9). Since $q_M(\boldsymbol{x},\boldsymbol{t}) = q_M(x_1,t_1)q_M(x_2,t_2)\cdots q_M(x_d,t_d)$ and therefore is separable and since

$$\delta(\boldsymbol{x}-\boldsymbol{t}) = \delta(x_1-t_1)\delta(x_2-t_2)\cdots\delta(x_d-t_d))$$

it follows that $q_M(\boldsymbol{x},\boldsymbol{t}) - \delta(\boldsymbol{x},\boldsymbol{t})$ is a finite sum of separable terms. For $d = 2$, it is

$$\begin{aligned}&q_M(x_1,t_1)q_M(x_2,t_2) - \delta(x_1,t_1)\delta(x_2,t_2) \\ &\quad = q_M(x_1,t_1)(q_M(x_2,t_2) - \delta(x_2-t_2)) \\ &\qquad + (q_M(x_1,t_1) - \delta(x_1-t_1))\delta(x_2-t_2).\end{aligned}$$

Hence in the case of $d = 2$, we have by (A11)

$$\begin{aligned}&||q(\cdot,\boldsymbol{t}) - \delta(\cdot-\boldsymbol{t})||_{-\alpha,2} \\ &\quad \leq \frac{1}{c_2}[||q_M(\cdot,t_1)||_{-\alpha/2,1}\cdot||q_M(\cdot,t_2) \\ &\qquad - \delta(\cdot-t_2)||_{-\alpha/2,1} \\ &\qquad + ||q_M(\cdot,t_1) - \delta(\cdot-t_1)||_{-\alpha/2,1} \\ &\qquad \cdot ||\delta(\cdot-t_1)||_{-\alpha/2,1}].\end{aligned} \tag{A12}$$

The Sobolev norm of $\delta(x-t)$, $||\delta(\cdot-t)||_{-\alpha/2,1} \leq C_0$ for all t when $\alpha/2 > 1/2$ and $||q_M(\cdot,t)||_{-\alpha/2,1} \leq C_1$ for all t and all M for such α. Hence we have

$$||q(\cdot,\boldsymbol{t}) - \delta(\cdot-\boldsymbol{t})||_{-\alpha,2} = O(2^{-M((\alpha/2)-(1/2))}))$$

for $r > \alpha/2 - 1/2$ and $d = 2$. The corresponding result for general d is

$$||q(\cdot,\boldsymbol{t}) - \delta(\cdot,\boldsymbol{t})||_{-\alpha,d} = O(2^{-M((\alpha/d)-(1/2))}).$$

The conclusion of Theorem 2 now follows from Schwarz's inequality for Sobolev spaces. For $f \in \boldsymbol{H}^{\beta}$ and $\alpha = d\beta + 1/2$, we have

$$\begin{aligned}|f_M(\boldsymbol{t}) - f(\boldsymbol{t})| &\leq ||q_M(\cdot,\boldsymbol{t}) - \delta(\cdot-\boldsymbol{t})||_{-d\beta-1/2,d} \\ &\quad \cdot ||f||_{d\beta+1/2,d} = O(2^{-M\beta}).\end{aligned} \tag{A13}$$

The $f_n(\boldsymbol{t})$ of Theorem 2 is just $f_M(\boldsymbol{t})$ when $2^M = n$ (recall that $f_M(\boldsymbol{t})$ can also be written as $f_{2^M}(\boldsymbol{t})$).

The L^2 convergence on bounded subsets $\boldsymbol{U}$ of $\boldsymbol{R}^d$ follows from the uniform convergence since

$$\int_{\boldsymbol{U}} |f(\boldsymbol{t})|^2\, d\boldsymbol{t} \leq ||\boldsymbol{U}|| \sup_{\boldsymbol{t}\in\boldsymbol{U}} |f(\boldsymbol{t})|^2$$

where $||\boldsymbol{U}||$ is the size of $\boldsymbol{U}$. So follow the convergence rates.

APPENDIX B

In this Appendix we prove Theorem 3, i.e., the consistency of the wavelet network. Here the function to be approximated is assumed to be in $\boldsymbol{V}_M$ for some M. As described in Section 3.C, all we need to show is that $\hat{c}_{N,\boldsymbol{k}} \to c_{\boldsymbol{k}}^{(0)}$ in the mean square sense as $N \to \infty$. For the sake of simplicity, we show this for the 1-D case. The extension to multiple dimensions is straight forward.

First, we assume that $\hat{c}_{N,k}$'s are obtained by using (16) and to emphasize this connection, they are denoted as $\hat{c}_{N,k}^{(2)}$, where superscript 2 denotes "the second scheme for estimating the weights." Since $t_i, i = 1,2,\cdots,N$, are i.i.d. and uniformly distributed in $[-1/2,+1/2]$, from (16) we have

$$\begin{aligned}E[\hat{c}_{N,k}^{(2)}] &= \frac{1}{N}\sum_{i=1}^{N} E[f(t_i)\varphi_{M,k}(t_i)] \\ &= \frac{1}{N}\sum_{i=1}^{N}\int_{-1/2}^{+1/2} f(t)\varphi_{M,k}(t)\, dt \\ &= \int_{-\infty}^{+\infty} f(t)\varphi_{M,k}(t)\, dt \\ &= \langle f, \varphi_{M,k}\rangle = c_k^{(0)}\end{aligned} \tag{B1}$$

for F with support in $[-1/2,1/2]$. Similarly, we have

$$\begin{aligned}E[(\hat{c}_{N,k}^{(2)})^2] &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} E[f(t_i)f(t_j)\varphi_{M,k}(t_i)\varphi_{M,k}(t_j)] \\ &= \frac{1}{N}[E[f^2(t)\varphi_{M,k}^2(t)]] + \left(1-\frac{1}{N}\right) \\ &\quad \cdot E^2[f(t)\varphi_{M,k}(t)] \\ &= \frac{1}{N}\langle f^2, \varphi_{M,k}^2\rangle + \left(1-\frac{1}{N}\right)(c_k^{(0)})^2.\end{aligned} \tag{B2}$$

Here t is a uniformly distributed random variable. From (B1), (B2), we have

$$\begin{aligned}E[(c_k^{(0)} - \hat{c}_{N,k}^{(2)})^2] &= Var[\hat{c}_{N,k}^{(2)}] = E[(\hat{c}_{N,k}^{(2)})^2] - (E[\hat{c}_{N,k}^{(2)}])^2 \\ &= \frac{1}{N}[\langle f^2, \varphi_{M,k}^2\rangle + (\langle f, \varphi_{M,k}\rangle)^2].\end{aligned} \tag{B3}$$

Since f has compact support, both inner products in the right hand side exist and are bounded. Hence, as $N \to \infty, \hat{c}_{N,k}^{(2)} \to c_k^{(0)}$ in the mean square sense and the rate of convergence is $O(1/N)$.

Now, consider the case when $\hat{c}_{N,k}$'s are obtained by using (14a) and are denoted now by $\hat{c}_{N,k}^{(1)}$. Let the corresponding wavelet network (the one with $\hat{c}_{N,k}^{(1)}$ as output layer weights)

be denoted by $\hat{f}^{(1)}_{M,N}$. Then, the consistency of $\hat{f}^{(1)}_{M,N}$ is easy to show if we have

$$E\left[\sum_{k=-K}^{K}(c_k^{(0)}-\hat{c}_{N,k}^{(1)})^2\right] \le E\left[\sum_{k=-K}^{K}(c_k^{(0)}-\hat{c}_{N,k}^{(2)})^2\right] \quad \text{(B4)}$$

To see (B4), we notice that from (14a), (15a) and (16),

$$\frac{1}{N}\sum_{i=1}^{N}(f(t_i)-\hat{f}^{(1)}_{M,N}(t_i))^2 \le \frac{1}{N}\sum_{i=1}^{N}(f(t_i)-\hat{f}^{(2)}_{M,N}(t_i))^2 \quad \text{(B5)}$$

where $\hat{f}^{(2)}_{M,N}$ is the wavelet network whose hidden layer weights are obtained by (16). Taking the expectation on both sides, we have

$$E[(f(t)-\hat{f}^{(1)}_{M,N}(t))^2] \le E[(f(t)-\hat{f}^{(2)}_{M,N}(t))^2]. \quad \text{(B6)}$$

However, notice that

$$E[(f(t)-\hat{f}^{(1)}_{M,N}(t))^2] = E\left[\sum_{k=-K}^{K}(c_k^{(0)}-\hat{c}_{N,k}^{(1)})^2\right] \quad \text{(B7)}$$

and similarly, that

$$E[(f(t)-\hat{f}^{(2)}_{M,N}(t))^2] = E\left[\sum_{k=-K}^{K}(c_k^{(0)}-\hat{c}_{N,k}^{(2)})^2\right] \quad \text{(B8)}$$

(B4) then follows.

If the training data is corrupted by additive white noise, it will be given by

$$T_N = \{t_i, y_i\}_{i=1}^{N}$$

where $y_i = f(t_i) + w_i$ and w_i is the white noise. In this case, using the same procedure as above, it can be shown that the conclusion of Theorem 3 still holds. The only difference is that the variance of the approximation error in (B3) increases. Nevertheless, the increase is inside the braces (by a term corresponding to the noise variance), hence the right-hand-side of (3B) still converges to zero with rate $O(1/N)$.

REFERENCES

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, J. L. McClelland and D. E. Rumelhart, Eds. Cambridge, MA: MIT Press, 1986.

[2] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol. 78, pp. 1481–1497, Sept. 1990.

[3] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, pp. 281–294, 1989.

[4] E. J. Hartman, J. D. Keeler, and J. M. Kowalski, "Layered neural networks with Gaussian hidden units as universal approximations," *Neural Computation*, vol. 2, pp. 210–215, 1990.

[5] B. Muller and J. Reinhardt, *Neural Networks*. Berlin, Germany: Springer–Verlag, 1991.

[6] V. Corradi and H. White, "Regularized neural networks: Some convergence rate results," preprint, 1993.

[7] F. Girosi and G. Anzellotti, "Rates of convergence for radial basis functions and neural networks," preprint, 1993.

[8] L. Xu, A. Krzyzak, and A. Yuille, "On radial basis function nets and kernel regression: Statistical consistency, convergence rates and receptive field size," preprint, 1993.

[9] Q. Zhang and A. Benveniste, "Wavelet networks," *IEEE Trans. Neural Net.*, vol. 3, Nov. 1992.

[10] Y. C. Pati and P. S. Krishnaprasad, "Analysis and synthesis of feedforward neural networks using discrete affine wavelet transformations," *IEEE Trans. Neural Net.*, vol. 4, Jan. 1993.

[11] T. Boubez and R. L. Peskin, "Wavelet neural networks and receptive field partitioning," reprint, 1993.

[12] T. Boubez, "Wavelet neural networks and receptive field partitioning," reprint, 1993.

[13] Y. Meyer, *Wavelets: Algorithms and Applications*, translated by R. D. Ryan. Philadelphia, PA: SIAM Press, 1993.

[14] I. Daubechies, *Ten Lectures on Wavelets*. Philadelphia, PA: SIAM Press, 1992.

[15] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet transform," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 674–693, July 1989.

[16] C. K. Chui, *An Introduction to Wavelets*. New York: Academic Press, 1992.

[17] J. Kovacevic and M. Vetterli, "Nonseparable multidimensional perfect reconstruction banks and wavelet for R^n," *IEEE Trans. Info. Theory*, vol. 38, pp. 533–555, Mar. 1992.

[18] G. G. Walter, "Approximation of the delta function by wavelets," *J. Approx. Theory*, vol. 71, pp. 329–343, Dec. 1992.

[19] H. T. Shim, "Gibbs' phenomenon in wavelets and how to avoid it," preprint, 1993.

[20] R. Coifman, Y. Meyer, and M. V. Wickerhauser, "Wavelet analysis and signal processing," in *Wavelets and Their Applications*, M. B. Ruskai, *et al.*, Ed. Boston, MA: Jones & Bartlett, 1992.

[21] G. G. Walter, "Pointwise convergence of wavelet expansions,"*J. Approx. Theory*, 1993.

[22] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*. Reading, MA: Addison–Wesley, 1979.

[23] D. G. Luenberger, *Introduction to Linear and Nonlinear Programming*. Reading, MA: Addison–Wesley, 1973.

[24] D. W. Scott, *Multivariate Density Estimation*. New York: Wiley, 1992.

[25] J.-N. Hwang, S.-R. Lay, M. Maechler, R. D. Martin, and J. Schimert, "Regression modeling in back-propagation and projection pursuit learning," *IEEE Trans. Neural Net.*, vol. 5, pp. 342–353, May 1994.

[26] G. G. Walter, *Wavelets and Other Orthogonal Systems*. CRC Press, 1994.

[27] S. E. Kelly, M. A. Kon, and L. A. Raphael, "Pointwise convergence of wavelet expansions," *Bull. AMS*, vol. 30, pp. 87–94, 1994.

Jun Zhang (M'88) received the B.S. degree in electrical and computer engineering from Harbin Shipbuilding Engineering Institute, Harbin, Heilongjiang, China, in 1982, and was admitted to the graduate program of the Radio Electronic Department of Tsinghua University. After a brief stay there, he traveled to the U.S. on a scholarship from the Li Foundation, Glen Cover, NY. He received the M.S. and Ph.D. degrees in electrical engineering from Rensselaer Polytechnic Institute in 1985 and 1988, respectively.

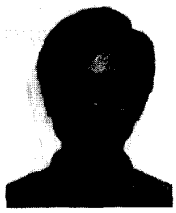
He joined the faculty of the Department of Electrical Engineering and Computer Science of the University of Wisconsin, Milwaukee, and is currently an Associate Professor. His research interests include image processing, computer vision, signal processing, and digital communications.

Gilbert G. Walter received the B.S. degree in electrical engineering from New Mexico State University and the Ph.D. degree in mathematics from the Unversity of Wisconsin, Madison, in 1962.

Since then, he has been a faculty member at the University of Wisconsin, Milwaukee, and is currently a Professor and Chair of the Department of Mathematical Sciences. His current research interests include special functions, generalized functions, sampling theorems, and wavelets.

Yubo Miao received the B.S. and M.S. degrees in computer science from the University of Minnesota in 1991, and the University of Wisconsin, Milwaukee, in 1993, respectively.

He was a research assistant in the development of neural networks for manufacturing quality control while obtaining the M.S. degree. Since 1993, he has been employed as a programmer/systems analyst at Marshfield Clinic and has worked in object-oriented design and programming. His interests include digital image and signal processing and visualization in scientific computing.



Wan Ngai Wayne Lee (M'93) received the B.S. degree in electrical engineering from the University of Wisconsin, Milwaukee, in 1991. He is currently working towards the M.S. degree in electrical engineering at the same institution.

His interests are in computer architecture, languages, operating systems, artificial intelligence, and signal processing.