Constructive Feedforward Neural Networks Using Hermite Polynomial Activation Functions

Liying Ma and K. Khorasani

Abstract—**In this paper, a constructive one-hidden-layer network is introduced where each hidden unit employs a polynomial function for its activation function that is different from other units. Specifically, both a structure level as well as a function level adaptation methodologies are utilized in constructing the network. The functional level adaptation scheme ensures that the "growing" or constructive network has different activation functions for each neuron such that the network may be able to capture the underlying input–output map more effectively. The activation functions considered consist of orthonormal Hermite polynomials. It is shown through extensive simulations that the proposed network yields improved performance when compared to networks having identical sigmoidal activation functions.**

Index Terms—**Constructive neural networks, functional level adaptation, Hermite polynomials, incremental training algorithms.**

I. Introduction

SINCE the early 1980s, a large number of neural network (NN) structures have been proposed and applied to various real world problems. The feedforward NNs (FNNs) are by far the most popular architectures due to their structural flexibility, good representational capabilities, and availability of a large number of training algorithms. The hidden units in a FNN usually have the same activation functions and are typically selected as sigmoidal or radial basis functions.

Many issues and problems have to be addressed and resolved when using FNNs. Among them, the automatic selection of a network structure is clearly critical. This problem has received a lot of attention by many researchers, and several promising algorithms have been proposed in the literature. Kwok and Yeung [17] surveyed some of the major constructive algorithms available in the literature. Dynamic node creation algorithm and its variants [1], [36] activity-based structure level adaptation [21], [37], cascade-correlation (CC) algorithm [8] and its variations [10], [34], constructive backpropagation (CBP) [22], adaptively constructing multilayer FNNs [28], and the constructive one-hidden-layer (OHL) algorithms [6], [12], [14], [16], [18], [24], [27], [29], [30] are among the most common constructive learning algorithms developed in the literature.

Among the constructive FNNs, the one-hidden-layer FNN (OHL-FNN) is by far the simplest in terms of both its structure and training efficiency, yet having a wide applicability due to its "universal approximation property" [3], [4], [9], [11], [13]. However, it has never been shown that the use of the same activation functions for all the hidden units is the best or the optimal choice for performance and generalization considerations. Consequently, opportunities remain for attempting to possibly improve the performance of the OHL-FNN by using more appropriate activation functions for the hidden units instead of simply using identical sigmoidal or radial basis functions.

In a constructive OHL-FNN [14], [16], [18], one or a pool of candidates with different initial weights and possibly different activation functions are tested and the one resulting in maximizing the performance index will be incorporated into the active network. However, the freedom in actually determining and selecting the activation functions will significantly increase the search space due to the difficulty in specifying the pool of activation functions that may be used. Although, the idea of using different activation functions for different units was mentioned by Fahlman [8] and other researchers [17], [18], a systematic and a rigorous algorithm and methodology for accomplishing this has not yet been developed.

In this paper, a new strategy is developed that is applicable to both fixed structure as well as the constructive network training paradigms, although our focus will be on the latter architecture. This is accomplished by using different activation functions with hierarchically varied nonlinearities, as the constructive learning of a OHL-FNN is progressing. This is motivated by the notion that a nonuniform use of activation functions may actually enhance the generalization capability of the resulting network.

Consider an input–output map that is to be realized by a OHL-FNN. The underlying function may generally contain a constant term, a linear term, second-order nonlinear terms and higher order nonlinearities. Any continuous function can be represented in this way at least in the sense of a series expansion, such as Taylor's for instance. All the series terms may in some sense be viewed as being combined in "parallel," so that their weighted sum would yield an approximation to the whole function. It is also well-known that any series expansion can be expressed to any desired level of accuracy as long as sufficient number of terms are used. This well-known fact, in principle, is very much analogous to the approximation capability of a FNN.

The idea is to basically let each unit in the FNN represent only one term of the series expansion associated with the function being approximated (along the similar lines as in the Taylor's series expansions). Toward this end, we propose to use orthonormal basis functions as the activation functions of the hidden units of a FNN. Each unit is expected to be responsible

Manuscript received July 2, 2001; revised January 17, 2005. This research was supported in part by the Natural Science and Engineering Research Council of Canada (NSERC) under Research Grant RGPIN-42515.

The authors are with the Department of Electrical and Computer Engineering, Concordia University, Montreal, QC, H3G 1M8 Canada (e-mail: kash@ece.concordia.ca).

Digital Object Identifier 10.1109/TNN.2005.851786

for approximating the corresponding nonlinearity contained in the expansion of the underlying function.

In this paper, an incremental adaptive constructive structure of a FNN [25], [26], [31] is considered. OHL-FNNs with both linear and nonlinear output layers are utilized here. Unlike the constructive learning approach proposed in [18], the initial network has no hidden units. The network is built up from a so-called "null net." In [18], the number of hidden units for the initial network was left unresolved and was to be determined by trial and error and/or heuristics. During the construction process in our proposed scheme, the hidden units are added to the active network one at a time, and the activation function of the hidden units are assigned successively from the lowest order orthonormal Hermite polynomial to the higher order ones.

As the network grows, naturally the activation function of the hidden units become more complicated. However, given that in our proposed methodology orthonormal Hermite polynomials are utilized, and given that these polynomials have interesting recursive characteristics, the computation and implementation concerns are alleviated to a large extent. Specifically, the orthonormal Hermite polynomials have the following desirable properties [35]: 1) any signal (or a function in a regression problem) can be represented to an arbitrarily high degree of accuracy by taking sufficient number of basis functions in a series expansion, implying that the orthonormal Hermite polynomials may be used as universal approximators; and 2) the recursive relationships for calculating the basis functions and their first-order derivatives can be utilized effectively in the constructive network design. Moreover, our extensive simulations with many noisy regression problems and classification problems (including a real-life problem) reported here have revealed that our proposed scheme actually yields FNNs that generalize much better than standard constructive OHL-FNNs having identical sigmoidal activation functions [18].

The outline of this paper is as follows. Section II presents a brief introduction to Hermite polynomials. Our proposed incremental constructive training algorithm is developed in Section III. Simulation results are provided in Section IV to demonstrate the effectiveness and potential of the new proposed constructive network. Conclusions are briefly drawn in Section V.

II. Hermite Polynomials

In this section, the Hermite polynomials with their hierarchical nonlinearities are first introduced. These polynomials will be used subsequently as the activation functions of the hidden units of the proposed constructive feedforward neural network (FNN). The orthogonal Hermite polynomials are defined over the interval $(-\infty, \infty)$ of the input space and are given formally as follows [35]:

$$H_0(x) = 1 \tag{1}$$
$$H_1(x) = 2x \tag{2}$$
$$\vdots$$
$$H_n(x) = 2xH_{n-1}(x) - 2(n-1)H_{n-2}(x), n \geq 2. \tag{3}$$

The definition of $H_n(x)$ may be given alternatively by

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n}\left(e^{-x^2}\right), \ n > 0, \quad \text{with } H_0(x) = 1. \tag{4}$$

The polynomials given in (1)–(3) are orthogonal to each other but not orthonormal. The orthonormal Hermite polynomials may then be defined according to

$$h_n(x) = \alpha_n H_n(x) \phi(x) \tag{5}$$

where

$$\alpha_n = (n!)^{-\frac{1}{2}} \pi^{\frac{1}{4}} 2^{-\frac{(n-1)}{2}} \tag{6}$$
$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}. \tag{7}$$

The first-order derivative of $h_n(x)$ can be easily obtained by virtue of the recursive nature of the polynomials defined in (3), that is

$$\frac{dh_n(x)}{dx} = (2n)^{\frac{1}{2}} h_{n-1}(x) - xh_n(x), \quad n \geq 1 \tag{8}$$
$$\frac{dh_0(x)}{dx} = \alpha_0 \frac{d\phi(x)}{dx} = -xh_0(x), \quad n = 0. \tag{9}$$

A typical set of the orthonormal Hermite polynomials are depicted in Fig. 1. In [35], the orthonormal Hermite polynomials are used as basis functions to model 1-D signals in the biomedical field for the purposes of signal analysis and detection. In [14], a selected combination of the orthonormal Hermite polynomials in a weighted-sum form is used as the *fixed* activation function of *all* the hidden units of the OHL-FNN. Hermite coefficients in [19], [23] are used as preprocessing filters, or served as the features of the process, which are then fed to (fuzzy) NNs for classification problems. An FNN is designed in [32] that uses Hermite function regression formula to approximate the hidden units activation functions to obtain an improved generalization capability. In this FNN, a fixed number of Hermite polynomials is used.

The projection pursuit learning (PPL) algorithm presented in [14], has hidden units that have an activation function that is formed by a linear combination of a fixed number of Hermite polynomial terms. However, in our constructive OHL-FNN, the activation function for a hidden unit has only one Hermite polynomial term, and the order of the activation function increases by one each time a new hidden unit is added to the network. In other words, the first hidden unit will employ $h_0(x)$ as its activation function, the second hidden unit will utilize $h_1(x)$ as its activation function, and so on. The resulting OHL-FNN with the Hermite polynomials as its activation functions has therefore the same structure as the standard OHL-FNN. However, the use of hierarchical polynomials is expected to increase the performance capabilities of the resulting network, as demonstrated in the subsequent simulation results.

Since a linear combination of a number of Hermite polynomial terms is used, PPL has good approximation ability when this number is properly chosen. However, the number of Hermite polynomial terms used for each hidden unit has to be specified a priori, which is problem-dependent and data-driven.

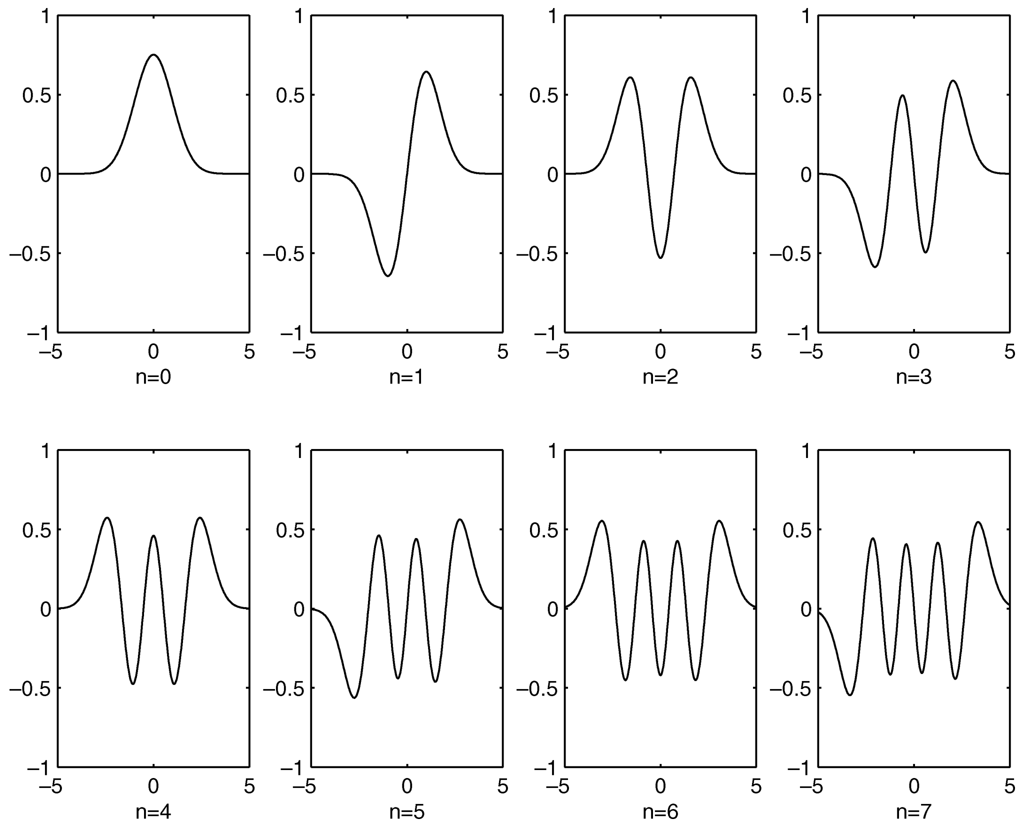


Fig. 1. Samples of the orthonormal Hermite polynomials, $h_n(\cdot)$ $(n = 0, 1, \cdots, 7)$.

Moreover, the PPL training process is computationally intensive. In [20], a pooling projection pursuit network (PPPN) is proposed to alleviate the critical requirement of selecting an adequate order of Hermite polynomials, but this method adds further computational burden and complexity.

In addition, selecting the set of pooling polynomials of different orders poses a new problem. On the other hand, our proposed algorithm is fully constructive in nature, and the network training is simple. Furthermore, in [14] only linear output activation functions is permitted which can only be applied to regression problems. However, our proposed scheme permits not only linear but also nonlinear output nodes so that it can be applied to both regression as well as classification problems.

It was indicated earlier that the Hermite polynomials are chosen due to their suitable properties. There are other polynomials that have similar properties, such as, consider the following typical polynomials: Legendre polynomials, Tchebycheff polynomials, and Laguerre polynomials [15]. However, the input range for these polynomials does not meet the requirement for a hidden activation function which should take values ranging from $-\infty$ to $+\infty$. If one uses a polynomial with a limited input range as an activation function for a hidden unit, one should then restrict the input-side weight space in which an optimal input-side weight vector has to be searched for under a given training criterion. The restriction of the input-side weight space is not desirable as it would limit the representational capability of the network. The justification and rationale for choosing the Hermite polynomials are therefore further motivated by their restriction-free input range characteristic.

Alternatively, the previous polynomials may still be considered as activation functions of the hidden units provided that the input to a hidden unit is normalized each time its input-side weight vector is updated. Although, the weight vector is now unconstrained, and only the input is normalized by a properly selected constant, this constant is actually a function of the weight vector. Therefore, the input to the hidden unit during the input-side training phase will be no longer linear with respect to the weight vector. Consequently, as far as the weight training is concerned the input-side training will now have two types of nonlinearities: one arising from the input to the hidden unit, and the other due to the activation function of the hidden unit. The corresponding optimization problem is clearly more complicated as compared to the one with only the nonlinearity of the activation function. It is, therefore, our conclusion that Hermite polynomials are expected to be the most suitable choice for the activation functions of the hidden units of a FNN as applied to the problems considered in this paper.

III. Proposed Incremental Constructive Training Algorithm

Consider a typical constructive OHL-FNN with a linear output layer and a polynomial-type hidden layer, as shown in Fig. 2. The output layer can also be nonlinear, as will be shown in our simulation results. The hidden unit with input and output connections denoted by dotted lines is the present neuron that is being trained before it is allowed to join the other existing hidden units in the network. Suppose, without loss of generality, that a given regression problem has an M-dimensional input

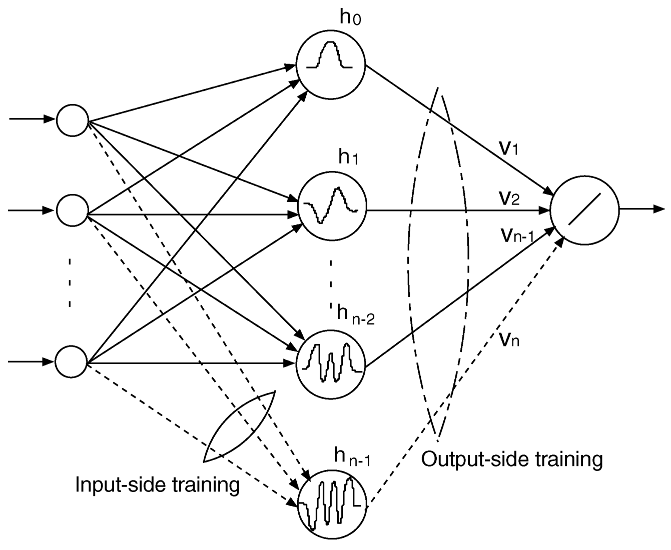


Fig. 2. Structure of a constructive OHL-FNN that utilizes the orthonormal Hermite polynomials as its activation functions for the hidden units.

vector and a scalar one-dimensional (1-D) output. The jth input and output samples are denoted by $\mathbf{X}^j = (x_0^j, x_1^j, \cdots, x_M^j)$ ($x_0^j = 1$ denoting the bias) and d^j (target), respectively, where $j = 1, 2, \cdots, P$ (P is the number of training data samples). The OHL constructive algorithm starts from a null network without any hidden units. At any given point during the constructive learning process, suppose there are $n-1$ hidden units in the hidden layer, and the nth hidden unit is being trained before it is added to the existing network (see Fig. 2).

The output of the nth hidden unit for the jth training sample is given by

$$f_n\left(s_n^j\right) = h_{n-1}\left(\sum_{m=0}^{M} w_{n,m} x_m^j\right) \tag{10}$$

where $h_n(\cdot)$ denotes the nth-order Hermite orthonormal polynomial. Its derivative can be calculated recursively from (8) and s_n^j is the input to the nth hidden unit, given by

$$s_n^j = \sum_{m=0}^{M} w_{n,m} x_m^j \tag{11}$$

where $w_{n,0}$ is the bias weight of the node with its input $x_0^j = 1$, and $w_{n,m}$ $(m \neq 0)$ is the weight from the mth component of the input vector to the nth hidden node.

The output of the network may now be expressed as follows:

$$\begin{aligned} y_{n-1}^j &= v_0 + \sum_{i=1}^{n-1} v_i h_{i-1}\left(s_i^j\right) \\ &= v_0 + v_1 h_0\left(s_1^j\right) + v_2 h_1\left(s_2^j\right) + v_3 h_2\left(s_3^j\right) \\ &\quad + \cdots + v_n h_{n-1}\left(s_{n-1}^j\right) \end{aligned} \tag{12}$$

where v_0 is the bias of the output node, and $v_1, v_2, \cdots, v_n$ are the weights from the hidden layer to the output node.

Clearly, the previous expression has a very close resemblance to a functional series expansion that utilizes the orthonormal Hermite polynomials as its basis functions, and where each additional term in the expansion contributes to improving the accuracy of the function that is being approximated. Through this series expansion, the approximation error will become smaller as more terms, having higher order nonlinearities, are included. This situation is actually quite similar to that of an incrementally constructing OHL-FNN, in the sense that the error is expected to become smaller as new hidden units having hierarchically higher order nonlinearities are successively added to the network. In fact, it has been shown that under certain circumstances incorporating new hidden units in a OHL-FNN can decrease monotonically the training error [18]. Therefore, in this sense adding a new hidden unit to the network is somewhat equivalent to the addition of a higher order term in a Hermite polynomial-based series expansion. It is in this sense that the present network is envisaged to be more suitable for constructive learning paradigm as compared to a fixed-structure FNNs. Note that, only if $s_1^j = s_2^j = \cdots = s_{n-1}^j$, then y_{n-1}^j will be an exact Hermite polynomial-based series expansion. Otherwise, as structured in the previous algorithm the expansion would be only approximate since the weights associated with each added neuron is adjusted separately, resulting in different inputs to the activation functions.

The problem addressed now is to determine how to train the nth hidden unit that is to be added to the active network. There are many objective functions (see [8] and [17] for more details) that can be considered for the input-side training of this hidden unit. A simple, but a general cost function that has been shown in the literature to work quite well, is as follows [8]:

$$J_{\text{input}} = \left|\sum_{j=1}^{P}\left(e_{n-1}^j - \bar{e}_{n-1}\right)\left(f_n\left(s_n^j\right) - \bar{f}_n\right)\right| \tag{13}$$

where

$$f_n\left(s_n^j\right) = h_{n-1}\left(s_n^j\right) \tag{14}$$

$$\bar{e}_{n-1} = \frac{1}{P}\sum_{j=1}^{P} e_{n-1}^j \tag{15}$$

$$\bar{f}_n = \frac{1}{P}\sum_{j=1}^{P} h_{n-1}\left(s_n^j\right) \tag{16}$$

$$e_{n-1}^j = d^j - y_{n-1}^j \tag{17}$$

where $f_n(s_n^j)$ (or $h_{n-1}(s_n^j)$) is an orthonormal Hermite polynomial of order $n-1$ used as the activation function of the nth hidden unit, e_{n-1}^j is the network output error when the network has $n-1$ hidden units and d^j is the jth target output to be used for the network training.

The derivative of J_{input} with respect to the weight $w_{n,i}$ is calculated according to (refer to the Appendix for further details)

$$\begin{aligned} \frac{\partial J_{\text{input}}}{\partial w_{n,i}} &= \operatorname{sgn}\left(\sum_{j=1}^{P}\left(e_{n-1}^j - \bar{e}_{n-1}\right)\left(f_n\left(s_n^j\right) - \bar{f}_n\right)\right) \\ &\quad \times \sum_{j=1}^{P}\left(e_{n-1}^j - \bar{e}_{n-1}\right) h'_{n-1}\left(s_n^j\right) x_i^j \end{aligned} \tag{18}$$

where $\mathrm{sgn}(\cdot)$ is a sign function. The first-order derivative $h'_{n-1}(s_n^j)$ in the previous expression can be easily evaluated by using the recursive expression (8) for $n \geq 2$ and (9) for $n = 1$.

A candidate unit that maximizes the objective function (13) will be incorporated into the network as the nth hidden unit. Following this stage, the output-side training is performed by solving a least squared (LS) problem given that the output layer has a linear activation function (see Fig. 2). Specifically, once the input-side training is accomplished, the network output y^j with n hidden units may now be expressed as follows:

$$y^j = \sum_{k=0}^{n} v_k f_k\left(s_k^j\right), \quad j = 1, 2, \cdots, P \tag{19}$$

where v_k, $(k = 1, 2, \cdots, n)$ are output-side weights of the kth hidden unit, and v_0 is the bias of the output unit with its input being fixed to $f_0 = 1$. The corresponding output neuron tracking error is now given by

$$\begin{aligned} e^j &= d^j - y^j \\ &= d^j - \sum_{k=0}^{n} v_k f_k\left(s_k^j\right), \quad j = 1, 2, \cdots, P. \end{aligned} \tag{20}$$

Subsequently, the output-side training is performed by solving the following LS problem given that the output layer has linear activation function, that is

$$J_{\text{output}} = \frac{1}{2}\sum_{j=1}^{P}\left(e^j\right)^2 = \frac{1}{2}\sum_{j=1}^{P}\left\{d^j - \sum_{k=0}^{n} v_k f_k\left(s_k^j\right)\right\}^2.$$

After performing the output-side training, a new error signal e^j is calculated for the next cycle of input-side and output-side training. Our proposed constructive FNN algorithm may now be summarized according to the following steps.

Step 1) **Initialization of the network**

Start the network training process with a OHL-FNN having one hidden unit. Set $n = 1$, $e^j = d^j$, and the activation function $= h_0(\cdot)$.

Step 2) **Input-side training for the n-th hidden unit**

Train only the input-side weights associated with the nth hidden unit $h_{n-1}(\cdot)$. The input-side weights for the existing hidden units, if any, are all frozen. One candidate for the nth hidden unit with random initial weights is trained using the objective function defined in (13), based on the "quickprop" algorithm [7]. If instead, for the nth unit, a pool of candidates are trained, then the one candidate that yields the maximum objective function will be chosen as the nth hidden unit to be added to the active network. The details for the weight adjustments are provided in the Appendix.

Step 3) **Output-side training**

Train the output-side weights of all the hidden units in the present network. Given that When the output layer has a linear activation function the output-side training may be performed by, for example, invoking the pseudoinverse operation resulting from the least square optimization criterion as indicated earlier. When the activation function of output layer is nonlinear, the quasi-Newton algorithm is to be used for training.

Step 4) **Network performance evaluation and training control**

Evaluate the network performance by monitoring the metric known as fraction of variance unexplained (FVU) [18] metric on the training data set. The FVU measure is defined according to

$$FVU = \frac{\sum_{j=1}^{P}\left(\hat{g}(\mathbf{x}^j) - g(\mathbf{x}^j)\right)^2}{\sum_{j=1}^{P}\left(g(\mathbf{x}^j) - \bar{g}\right)^2} \tag{21}$$

where $g(\cdot)$ is the function being implemented by the FNN, $\hat{g}(\cdot)$ is an estimate of $g(\cdot)$ or the output of the trained network, and $\bar{g}$ is the mean value of $g(\cdot)$. The network training is terminated provided that certain stopping conditions are satisfied. These conditions may be specified formally, for example, in terms of a prespecified FVU threshold or a maximum number of permissible hidden units, among others. If the stopping conditions are not satisfied, then the network output y^j and the network output error $e^j = d^j - y^j$ are computed, n is increased by 1, i.e., $n = n + 1$, and we proceed to Step 2).

It is well-known in the constructive OHL-FNNs literature [18] that the determination of a proper initial network size is a problem that needs careful attention. This initialization may significantly influence the effectiveness and efficiency of the network training that follows. However, in our proposed scheme, the network training starts from the smallest possible architecture (that is the "null" network). This is an important advantage of our proposed algorithm over the similar methods previously presented in the literature.

Although the proposed algorithm is presented in the context of a regression problem, it is straightforward to extend the algorithm to also classification problems by simply changing the activation function of the output node from a linear one to a sigmoidal one. As pointed out previously, when the output node is considered to be nonlinear, the pseudoinverse based LS solution to the output-side training can not be applied any longer. Other appropriate algorithms for the nonlinear LS minimization problem should now be chosen. For instance, second-order algorithms such as the quasi-Newton algorithm may be utilized.

IV. Simulation Results

In this section, simulated results are presented to demonstrate the effectiveness and the superiority of the proposed constructive OHL network with Hermite polynomial activation functions. It is found through extensive simulations that for "simple" regression problems the performance of the new network is similar to a conventional (from here onwards referred to as "standard") constructive OHL networks that employ sigmoidal activation functions. Our proposed constructive networks, however,

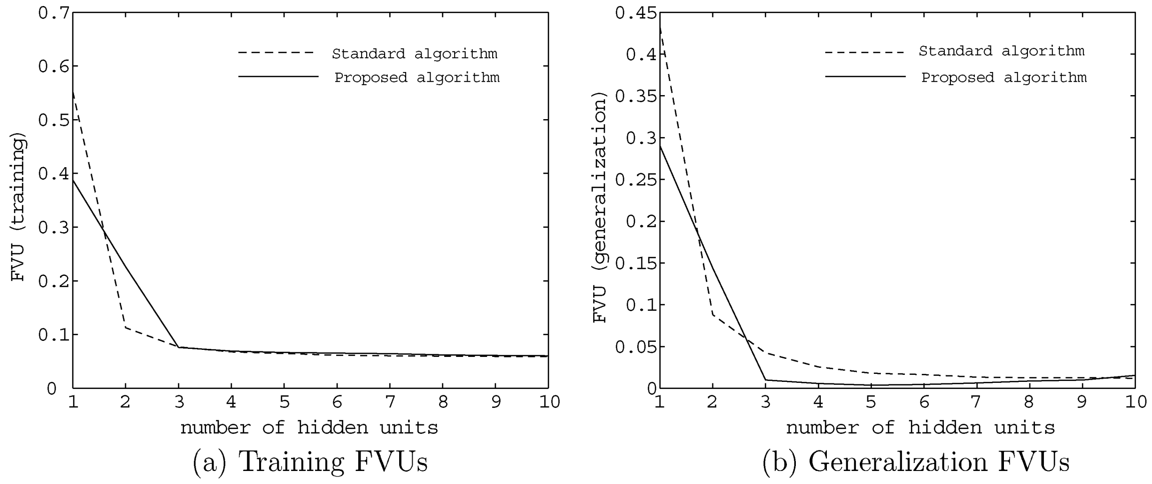


Fig. 3. (a) Training and (b) generalization FVUs of the proposed and the "standard" constructive OHL networks for Example I.

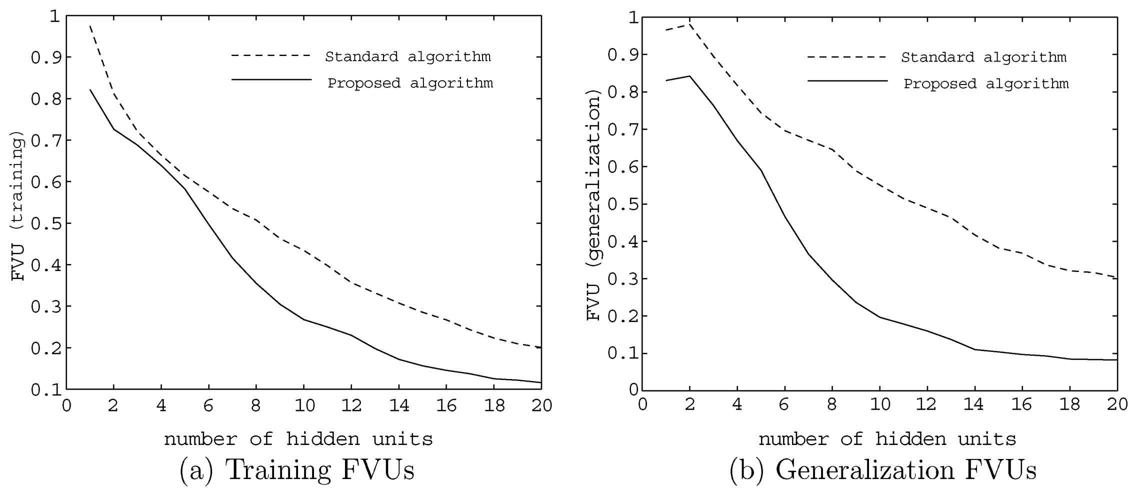


Fig. 4. (a) Training and (b) generalization FVUs of the proposed and the "standard" constructive OHL networks for Example II (the HF function).

provide improved performance when applied to "complicated" regression or classification problems.

In all the following simulations the corresponding learning parameters for both the "standard" OHL constructive algorithm and our "proposed" algorithm are set independently so that each network solution would demonstrate its "best" achievable performance on average. Furthermore, it is our opinion that this will make the comparisons shown in the following more appropriate. Ten independent runs (where each run has a different set of random initial weights) are conducted to obtain an average basis for comparison. Examples I–III deal with the regression problem whereas Examples IV–V are concerned with the classification problem.

A. Regression Cases

Example I: Consider the 1-D function

$$g(x) = 0.2\left\{1+\frac{1}{10}(x-7)^2\right\}\cos(2x) + 0.5e^{-2x}\sin(2x-0.1\pi). \quad (23)$$

The first term of this function was used in [5]. The second term is added here to make the function even more complicated. One hundred $(P = 100)$ uniformly distributed random samples over [0, 1] were chosen for network training. The signal-to-noise ratio (SNR) is selected to be 10 [dB]. The number of uniformly sampled noiseless data for evaluating the generalization FVU is 200. The training and the generalization FVUs are shown in Fig. 3. It is seen clearly that the proposed algorithm works slightly better than the "standard" algorithm in terms of the generalization FVU.

Example II: Consider the two-dimensional (2-D) harmonic function (HF)

$$g(x_1,x_2) = 42.659\left(0.1+(x_1-0.5)\left(0.05+(x_1-0.5)^4 -10(x_1-0.5)^2(x_2-0.5)^2+5(x_2-0.5)^4\right)\right). \quad (23)$$

Two hundred and twenty-five $(P = 225)$ uniformly distributed random samples were generated from the interval [0, 1] for network training. 10,000 uniformly sampled points from the same interval without noise were used for generalization. The simulation results are given in Fig. 4. It can be observed from this figure that our proposed new algorithm results in considerably smaller generalization FVU than the "standard" algorithm. This suggests that our proposed algorithm may be more effective for more "complicated" regression problems. Figs. 5 and 6 depict the generalized surfaces for both the proposed and the "standard" algorithms for the regression function HF, respectively.

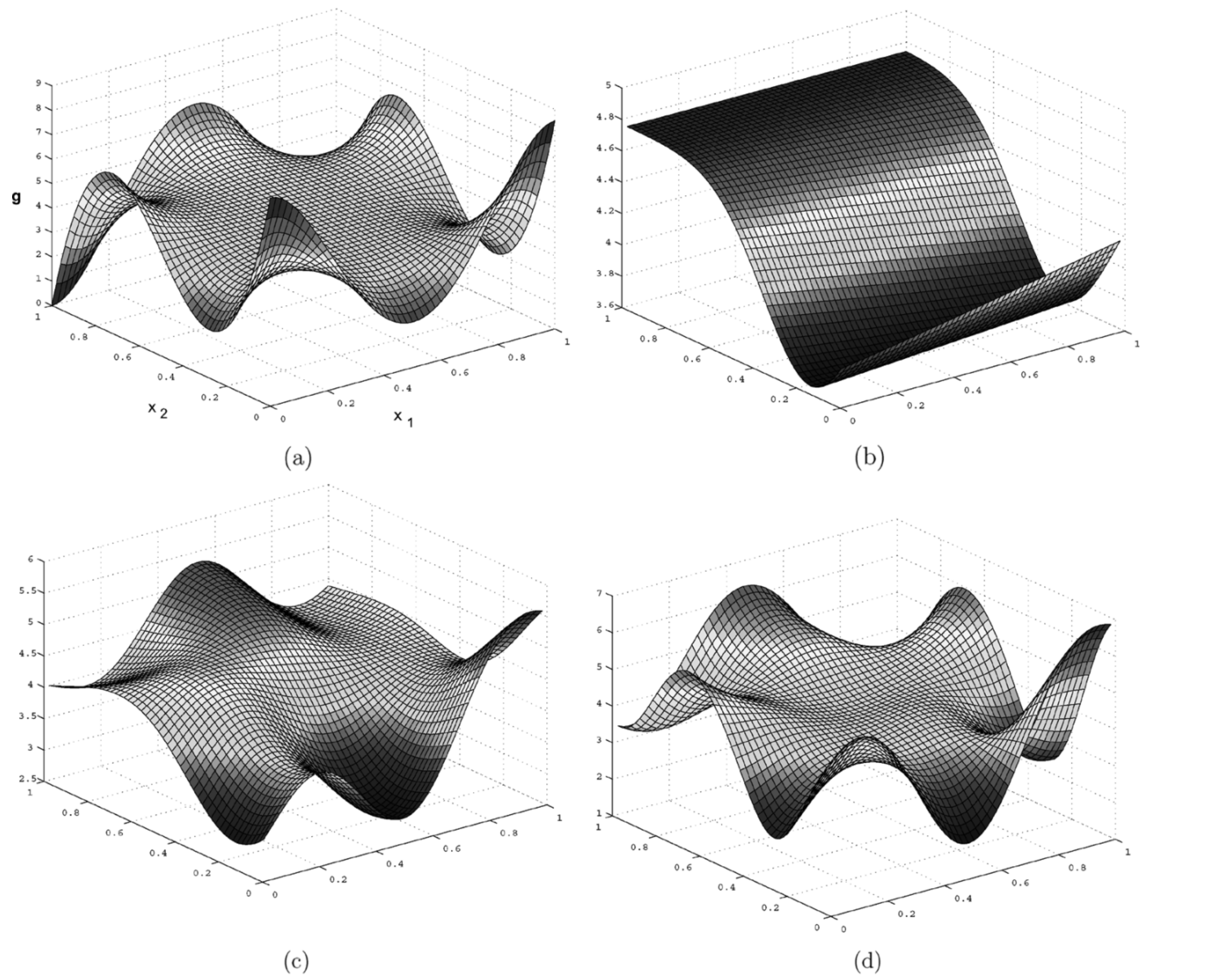


Fig. 5. Original and generalized HF represented by our proposed algorithm. HU denotes hidden unit. (a) HF (original). (b) Generalized HF with 1 HU. (c) Generalized HF with 5 HUs. (d) Generalized HF with 20 HUs.

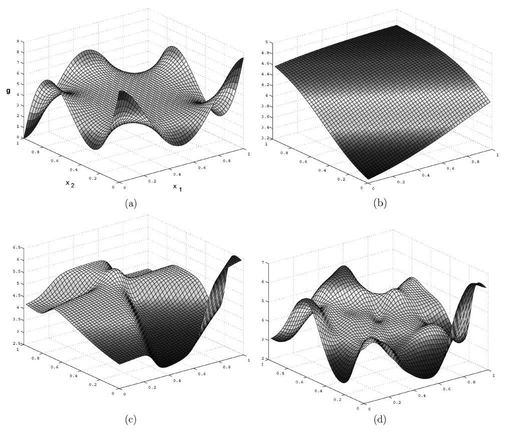


Fig. 6. Original and generalized HF represented by the "standard" algorithm. HU denotes hidden unit. (a) HF (original). (b) Generalized HF with 1 HU. (c) Generalized HF with 5 HUs. (d) Generalized HF with 20 HUs.

To further verify and validate the performance improvements possible by our proposed architecture, four other 2-D regression functions, as used in [14] and [18] and given in the following are used for simulations.

TABLE I
MEAN FVU VALUES FOR THE TRAINING AND THE GENERALIZATION OF THE PROPOSED AND THE "STANDARD" CONSTRUCTIVE OHL NETWORKS FOR THE FIVE TWO-DIMENSIONAL REGRESSION FUNCTIONS CONSIDERED IN [18]

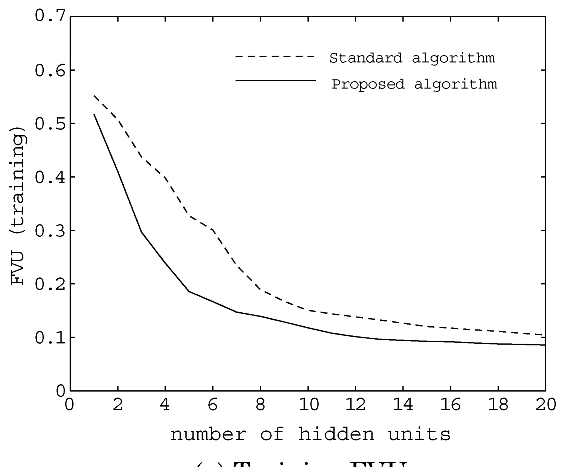
		Number of hidden units (HUs)							
		Training				Generalization			
Function	Approach	2	5	10	20	2	5	10	20
CIF	"standard"	0.582	0.345	0.196	0.107	0.620	0.403	0.206	0.111
	proposed	0.556	0.348	0.138	0.087	0.594	0.349	0.095	0.039
AF	"standard"	0.630	0.208	0.132	0.101	0.700	0.156	0.073	0.050
	proposed	0.501	0.218	0.151	0.109	0.538	0.166	0.076	0.027
HF	"standard"	0.812	0.614	0.434	0.201	0.980	0.744	0.551	0.303
	proposed	0.726	0.582	0.268	0.116	0.842	0.590	0.197	0.082
RF	"standard"	0.634	0.224	0.115	0.073	0.668	0.212	0.097	0.048
	proposed	0.441	0.155	0.092	0.074	0.443	0.091	0.027	0.019
SIF	"standard"	0.297	0.153	0.099	0.072	0.298	0.119	0.048	0.043
	proposed	0.334	0.157	0.100	0.079	0.284	0.095	0.031	0.026

- *Additive function (AF)*

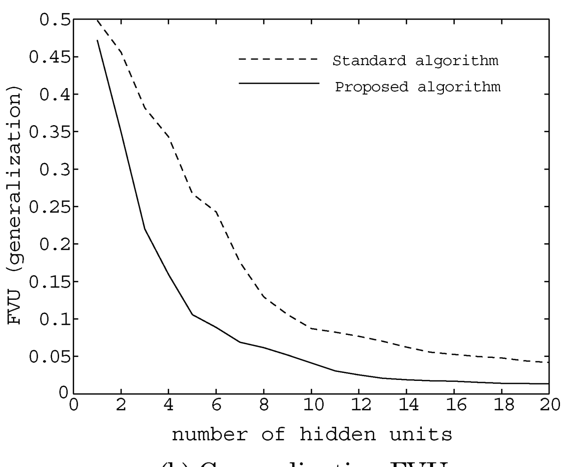
$$g(x_1,x_2)=1.3356\left\{1.5(1-x_1)+e^{2x_1-1}\sin\left(3\pi(x_1-0.6)^2\right)\right.$$
$$\left.+e^{3(x_2-0.5)}\sin\left(4\pi(x_2-0.9)^2\right)\right\}. \quad (24)$$

- *Radial function (RF)*

$$g(x_1,x_2)=24.234\left\{\left((x_1-0.5)^2+(x_2-0.5)^2\right)\right.$$
$$\left.\times\left(0.75-(x_1-0.5)^2-(x_2-0.5)^2\right)\right\}. \quad (25)$$



(a) Training FVUs



(b) Generalization FVUs

Fig. 7. (a) Training and (b) generalization FVUs of the proposed and the "standard" constructive OHL networks for the 3-D regression function of Example III.

- *Simple interaction function (SIF)*

$$g(x_1,x_2) = 10.391\left\{(x_1-0.4)(x_2-0.6)+0.36\right\}. \quad (26)$$

- *Complicated interaction function (CIF)*

$$g(x_1,x_2)=1.9\left\{1.35+e^{x_1-x_2}\sin\left(13(x_1-0.6)^2\right)\sin(7x_2)\right\}. \quad (27)$$

Simulations for the previous four functions are performed under the same settings as in the HF case. Simulated results for these five functions are summarized in Table I. Clearly, the generalization FVUs are significantly improved by utilizing our proposed algorithm. For SIF and AF functions both algorithms yield similar training FVUs as the number of hidden units increases, although our proposed algorithm results in smaller training FVUs for the other functions regardless of how large the number of hidden units become.

Example III: Consider now the three-dimensional (3-D) function given by

$$g(x_1,x_2,x_3)=\frac{1}{1+e^{-e^{x_1}+(x_2-0.5)^2+3\sin(\pi x_3)}}+2x_1\sin(2\pi x_2). \quad (28)$$

A thousand $(P = 1000)$ uniformly distributed random samples within the interval [0, 1] were used for network training. Also, 1000 different samples were uniformly generated for generalization performance evaluation. The SNR is chosen to be 10 [dB]. The results are depicted in Fig. 7. Obviously, in this case our proposed algorithm works much better in terms of the generalization FVU as compared to the standard algorithm.

To summarize the simulated results for the regression problem in Examples I–III, one can observe that our proposed constructive network having Hermite polynomials as activation functions learns from the training samples, at least as well as, and generally much better than the standard networks using sigmoidal activation functions. The proposed constructive network also produces smaller generalization FVUs. In other words, our proposed network architecture provides similar or even improved learning capabilities, and is more capable of representing the data and generalizing beyond the training samples as compared to a conventional technique.

Finally, to demonstrate that the concept presented in this paper can equally be applied to classification problems we apply the proposed algorithm to a two-category classification problem and a real-life benchmark problem known as "Cancer" [2], [33]. Note that in this problem the output node has no longer a linear activation function and is now changed to a sigmoidal function. The output-side training is carried out using the quasi-Newton algorithm. To make a fair comparison, the training parameters for the "standard" and our proposed algorithms have been selected independently such that each algorithm could yield approximately its best performance.

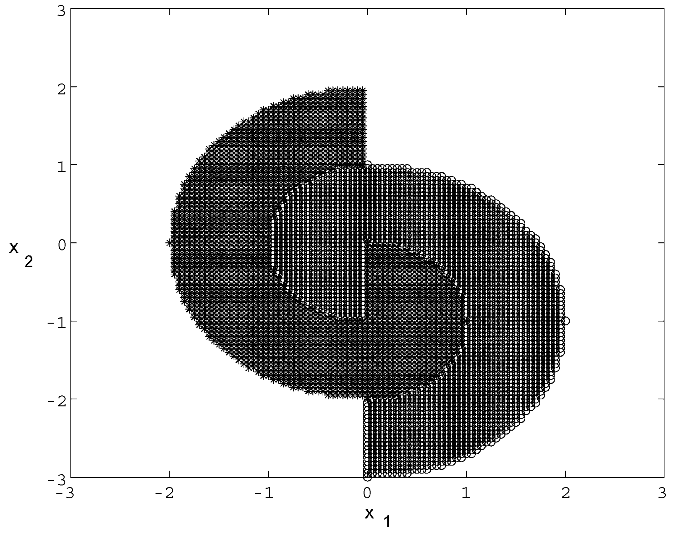


Fig. 8. Original two categories.

B. Classification Cases

Example IV: Consider the two-category classification problem

$$g(x_1,x_2) = \begin{cases} 0 & x_1<0, 1\le x_1^2+x_2^2\le 4 \\ & or\ x_1>0, x_1^2+(x_2+1)^2\le 1. \\ 1 & x_1<0, x_1^2+x_2^2<1 \\ & or\ x_1>0, 1<x_1^2+(x_2+1)^2\le 4. \end{cases} \quad (29)$$

The two categories are sampled at an 0.1 interval in both the horizontal (x_1) and the vertical (x_2) directions to obtain sufficient number of samples for network training. Sampling with the resolution of 0.05 is performed to collect data for network generalization. Fig. 8 shows the original two categories. The

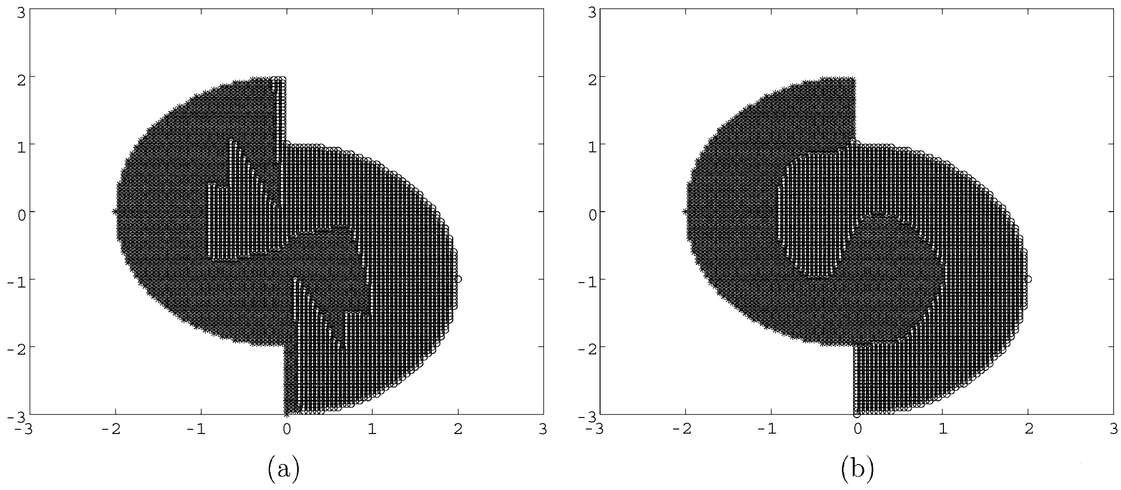


Fig. 9. Generalized two categories by the "standard" and the proposed constructive FNNs with 5 hidden units. (a) Standard constructive algorithm. (b) Proposed constructive algorithm.

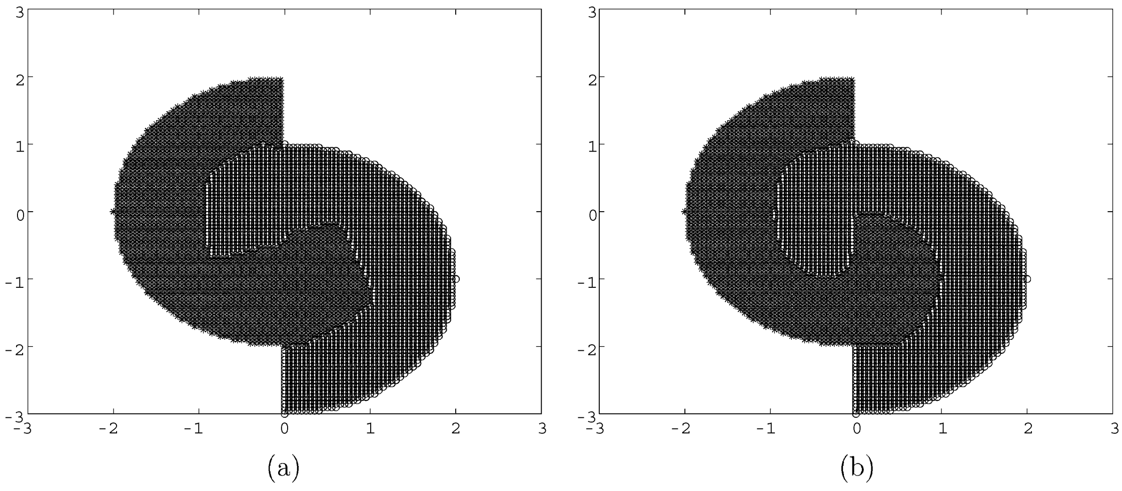


Fig. 10. Generalized two categories by the "standard" and the proposed constructive FNNs with 10 hidden units. (a) Standard constructive algorithm. (b) Proposed constructive algorithm.

generalized two categories are depicted in Figs. 9 and 10, that are obtained from two networks with 5 and 10 hidden units trained by the "standard" and our proposed constructive algorithms, respectively. Comparisons of the training and the generalization errors (FVUs) for both constructive algorithms are shown in Fig. 11. From Figs. 9–11, it follows very clearly that our proposed constructive FNN using Hermite polynomials as its activation functions yields better performance as compared to the standard constructive FNN that utilizes identical sigmoidal functions as its activation functions.

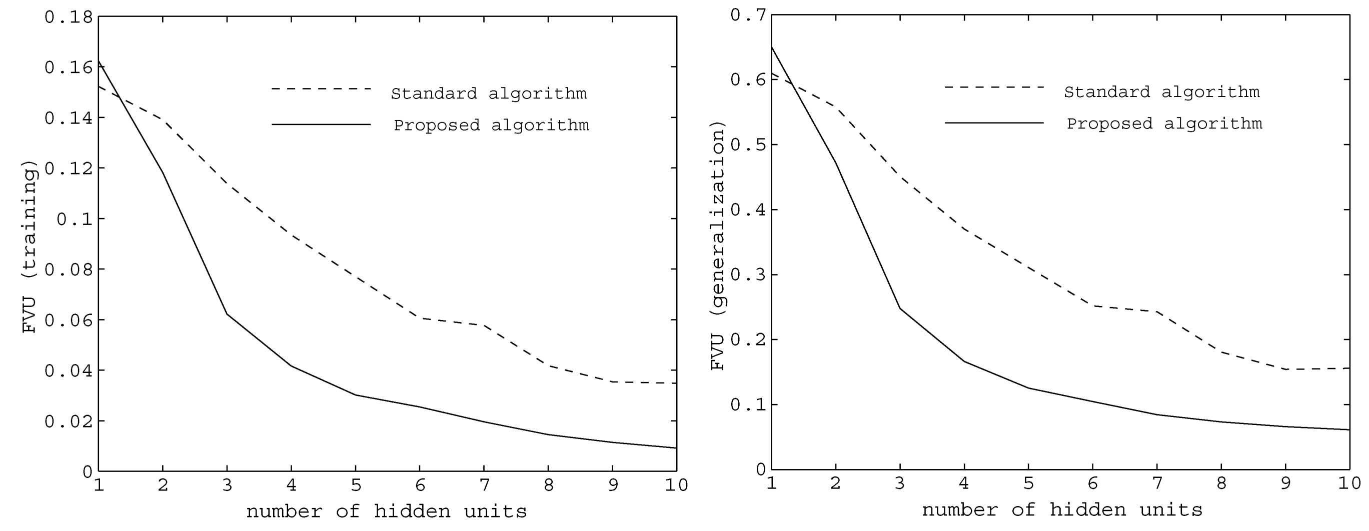
Example V: Benchmark classification problem "Cancer" [2], [33]:

The database in this benchmark case contains 699 samples of "real" data. The data was originally obtained from the University of Wisconsin Hospitals, Madison, from Dr. W. H. Wolberg. For further details regarding the specifications of the data refer to [2] and [33]. We have performed a 4-fold cross-validation tests and simulations. Training samples, validation samples and testing samples contained 350, 175, and 174 data points, respectively. The training samples are used for constructive network learning, the validation samples are used for stopping the constructive algorithm, and the testing samples are used for evaluating the performance of the network. There are nine input nodes and two output nodes. The maximum number of hidden units was set to 20 in order to observe the performance of the designed networks. Forty (40) independent runs are conducted. Simulation results are shown in Figs. 12–14. From the simulation results, the following remarks are concluded.

C1) When the number of hidden units becomes larger than 3, the training FVU of our proposed algorithm indicates a significant decrease when compared to FVU of the standard algorithm (Fig. 12).

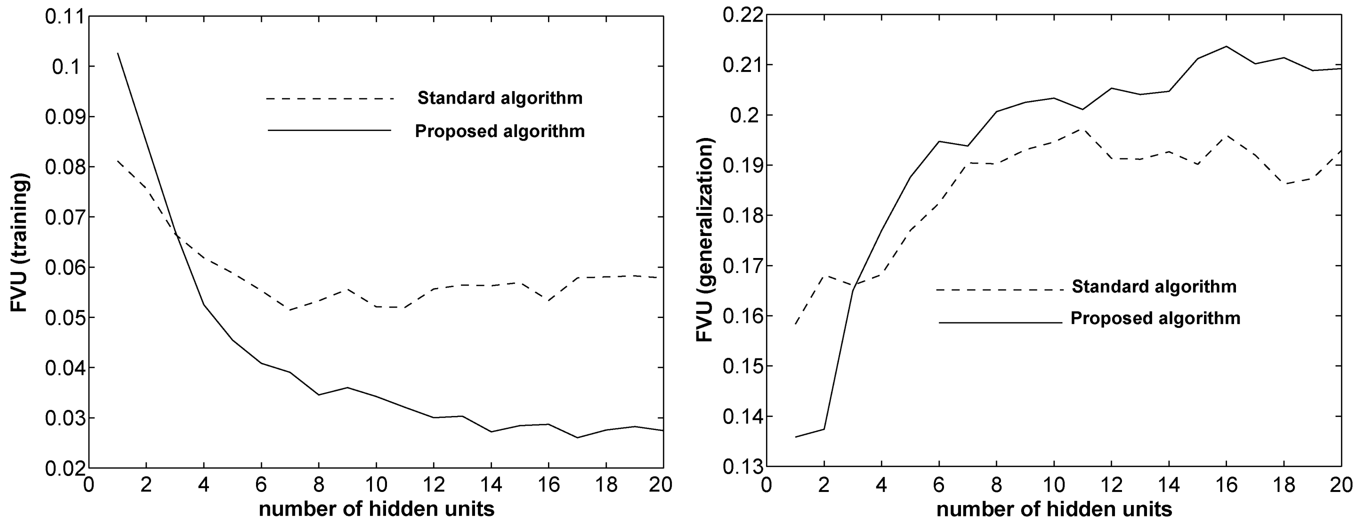
C2) It is interesting to note that the generalization FVUs of both algorithms increase as more new hidden units are added. Our proposed algorithm shows smaller FVU if the number of hidden units is less than 3, but indicates larger FVU if the number of hidden units becomes larger than 3 (Fig. 13).

C3) In both training and generalization, if the FVU is smaller (larger) then the mean recognition rate will be larger (smaller). From Fig. 13(b), we observe that the mean recognition rates of our proposed algorithm are higher than the standard algorithm when the number of hidden units is less than 3.



(a) Training FVUs (b) Generalization FVUs

Fig. 11. (a) Training and (b) generalization FVUs of the proposed and the "standard" constructive OHL networks for a two-category classification problem.



(a) Training FVUs (b) Generalization FVUs

Fig. 12. (a) Training and (b) generalization FVUs of the proposed and the "standard" constructive OHL networks for the benchmark "Cancer" of Example V.

C4) From Fig. 14, we observe that even though the maximum recognition rate for training of both standard and our proposed methods can reach 100%, only our proposed algorithm can achieve a maximum recognition rate of 100% in testing.

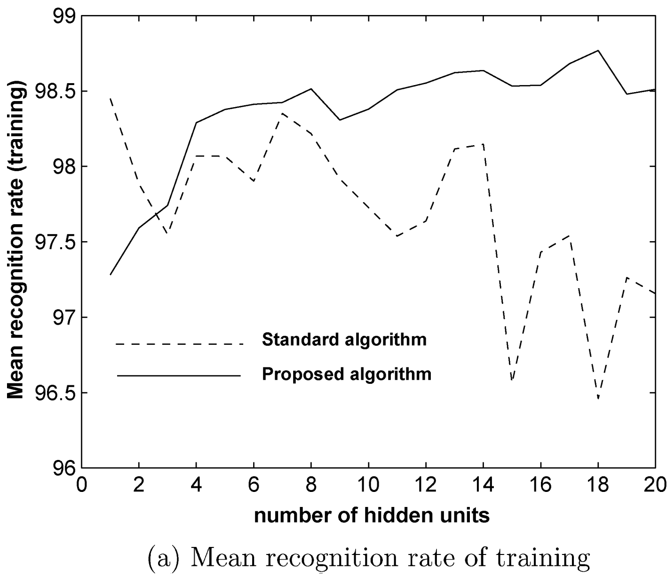
V. Conclusion

In this paper, we have proposed a new type of a constructive OHL-FNN that adaptively assigns appropriate orthonormal Hermite polynomials to its generated neurons. The network generally learns as effectively as, but generalizes much better than the conventional constructive OHL networks [18] in the regression and classification cases studied in the paper. Several simulations for the regression problem are carried out to confirm the effectiveness and superiority of our proposed new algorithm. Applications to a simple two-category problem and a "Cancer" benchmark problem are also included to demonstrate the potential utility of our proposed algorithm to the classification problem.

Appendix

"Quickprop" Algorithm

The *quickprop* algorithm developed by Fahlman [7] has played an important role in the input-side training of our constructive OHL-FNNs. In the Appendix, a brief introduction to this algorithm is provided in the context of correlation-based input-side training. Note that *quickprop* is a second-order optimization method based on Newton's method. As shown in the following, it is simple in form, but has been found to be surprisingly effective when used iteratively [8].



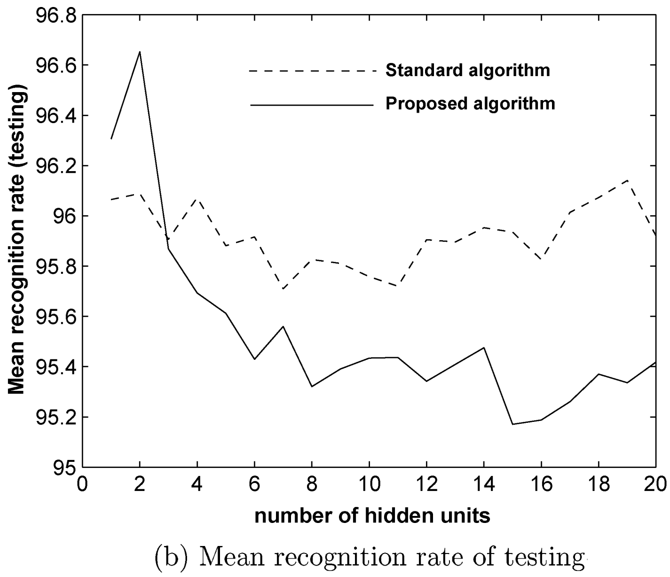
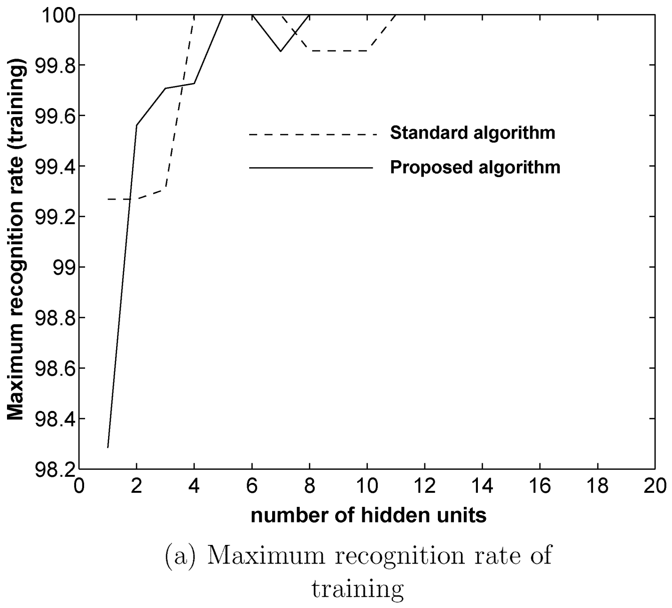


Fig. 13. (a) Recognition rates of training and (b) of testing FVUs of the proposed and the "standard" constructive OHL networks for the benchmark "Cancer" of Example V.



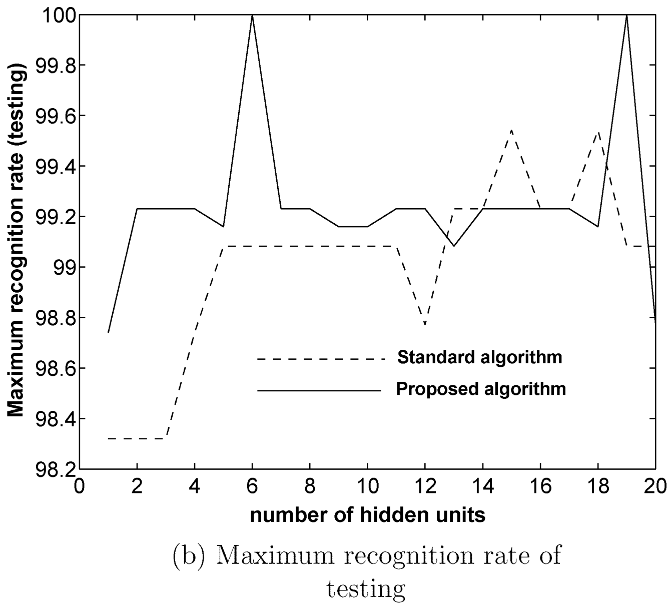


Fig. 14. Maximum recognition rates of (a) training and (b) testing of the proposed and the "standard" constructive OHL networks for the benchmark "Cancer" of Example V.

The correlation-based objective function for input-side training is reproduced here

$$J_{\text{input}} = \left| \sum_{j=1}^{P} \left(e_{n-1}^{j} - \bar{e}_{n-1} \right) \left(f_n \left(s_n^j \right) - \bar{f}_n \right) \right| \tag{30}$$

$$\bar{e}_{n-1} = \frac{1}{P} \sum_{j=1}^{P} e_{n-1}^{j} \tag{31}$$

$$\bar{f}_n = \frac{1}{P} \sum_{j=1}^{P} f_n \left(s_n^j \right) \tag{32}$$

$$s_n^j = \sum_{i=0}^{M} w_{n,i} x_i^j \tag{33}$$

where it is assumed that there are already $n-1$ hidden units in the network. The previous objective function is used to train the input-side weight $\{w\}$ of the nth hidden unit, and $f_n(\cdot)$ is the activation function of the nth hidden unit, x_i^j is the ith element of the input vector of dimension $M \times 1$, e_{n-1}^j is the output node error, and $f_n(s_n^j)$ is the output of the nth hidden unit, all defined for the training sample j. The derivative of J_{input} with respect to an input-side weight is given by

$$\begin{aligned} \frac{\partial J_{\text{input}}}{\partial w_{n,i}} &= \sum_{j=1}^{P} \delta_j x_i^j \\ \delta_j &= \operatorname{sgn}(C_0) \left(e_{n-1}^j - \bar{e}_{n-1} \right) \frac{d f_n \left(s_n^j \right)}{d s_n^j} \\ C_0 &= \sum_{j=1}^{P} \left(e_{n-1}^j - \bar{e}_{n-1} \right) \left(f_n \left(s_n^j \right) - \bar{f}_n \right) \end{aligned} \tag{34}$$

where for simplicity, $\bar{f}_n$ is treated as a constant in the previous calculation, even though $\bar{f}_n$ is actually a function of the input-side weights. Let us define

$$S_{n,i}(t) = -\frac{\partial J_{\text{input}}}{\partial w_{n,i}}, \quad i = 0, 1, \cdots, M \tag{35}$$

where t is the iteration step. The *quickprop* algorithm maximizing (30) may now be expressed by

$$\Delta w_{n,i}(t) = \begin{cases} \varepsilon S_{n,i}(t), & if\ \Delta w_{n,i}(t-1) = 0, \\ & (i = 0, 1, \cdots, M) \\ \frac{S_{n,i}(t)\Delta w_{n,i}(t-1)}{S_{n,i}(t-1) - S_{n,i}(t)}, & if\ \Delta w_{n,i}(t-1) \neq 0 \text{ and} \\ & \frac{S_{n,i}(t)}{S_{n,i}(t-1) - S_{n,i}(t)} < \mu \\ \mu \Delta w_{n,i}(t-1), & \text{otherwise} \end{cases} \tag{36}$$

where $\Delta w_{n,i}(t-1) = w_{n,i}(t) - w_{n,i}(t-1)$, and ε and μ are positive user-specified parameters. Note that in the simulation results presented in this work for the Examples *I–V* using the "logsig" activation function μ is selected as 1.75 and ε is selected as 0.35. Furthermore, in these examples with the Hermite polynomial activation functions we have selected $\mu = 0.25$ and $\varepsilon = 0.01$.

References

[1] T. Ash, "Dynamic node creation in backpropagation networks," *Connect. Sci.*, vol. 1, no. 4, pp. 365–375, 1989.

[2] C. L. Blake and C. J. Merz. (1998) UCI Repository of Machine Learning Databases. Dept. Inf. Comput. Sci., Univ. California, Irvine, CA. [Online]http://www.ics.uci.edu/~mlearn/mlrepository.html

[3] T. Chen and H. Chen, "Universal approximation to nonlinear operators by neural networks with arbitrary activation function and its application to dynamical systems," *Neural Netw.*, vol. 6, pp. 911–917, 1995.

[4] G. Cybenko, "Approximation by superposition of sigmoidal functions," *Math. Control, Signals Syst.*, vol. 2, no. 4, pp. 303–314, 1989.

[5] T. Draelos and D. Hush, "A constructive neural network algorithm for function approximation," in *Proc. IEEE Int. Conf. Neural Network*, vol. 1, Washington, DC, 1996, pp. 50–55.

[6] S. H. Ling, F. H. F. Leung, H. K. Lam, and P. K. S. Tam, "Tuning of the structure and parameters of a neural network using an improved genetic algorithm," *IEEE Trans. Neural Netw.*, vol. 14, no. 1, pp. 79–88, Jan. 2003.

[7] S. E. Fahlman, "An empirical study of learning speed in back-propagation networks," Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep., CMU-CS-88-162, 1988.

[8] S. E. Fahlman and C. Lebiere, "The Cascade-Correlation Learning Architecture," Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep., CMU-CS-90-100, 1991.

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

[10] S.-U. Guan and S. Li, "An approach to parallel growing and training of neural networks," in *Proc. 2000 IEEE Int. Symp. Intelligent Signal Processing Communication Systems (ISPACS'00)*, Honolulu, HI, pp. 1101–1104.

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

[12] S. Hosseini and C. Jutten, "Maximum likelihood neural approximation in presence of additive colored noise," *IEEE Trans. Neural Netw.*, vol. 13, no. 1, pp. 117–131, Jan. 2002.

[13] D. Husmeier, *Neural Networks for Conditional Probability Estimation, Perspectives in Neural Computation*. New York: Springer-Verlag, 1999.

[14] 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 Netw.*, vol. 5, no. 3, pp. 342–353, May 1994.

[15] G. A. Korn and T. M. Korn, *Mathematical Handbook for Scientists and Engineers*, 2nd ed. New York: McGraw-Hill, 1968.

[16] T. Y. Kwok and D. Y. Yeung, "Bayesian regularization in constructive neural networks," in *Proc. Int. Conf. Artificial Neural Networks*, Bochum, Germany, 1996, pp. 557–562.

[17] ——, "Constructive algorithms for structure learning in feedforward neural networks for regression problems," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 630–645, May 1997.

[18] ——, "Objective functions for training new hidden units in constructive neural networks," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 1131–1148, Sep. 1997.

[19] B. Lau and T. H. Chao, "Aided target recognition processing of mudss sonar data," in *Proc. SPIE*, vol. 3392, Orlando, FL, 1998, pp. 234–242.

[20] S. R. Lay, J. N. Hwang, and S. S. You, "Extensions to projection pursuit learning networks with parametric smoothers," in *Proc. IEEE Int. Conf. Neural Networks (ICNN'94)*, vol. 3, Orlando, FL, 1994, pp. 1325–1330.

[21] T. C. Lee, *Structure Level Adaptation for Artificial Neural Networks*. Norwell, MA : Kluwer, 1991.

[22] M. Lehtokangas, "Modeling with constructive backpropagation," *Neural Netw.*, vol. 12, pp. 707–716, 1999.

[23] T. H. Linh, S. Osowski, and M. Stodolski, "On-line heart beat recognition using hermite polynomials and neuro-fuzzy network," in *Proc. 19th IEEE Instrumentation and Measurement Technology Conf. (IMTC'02)*, vol. 1, Anchorage, AK, pp. 165–170.

[24] X. Yao, M. Islam, and K. Murase, "A constructive algorithm for training cooperative neural network ensembles," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 820–834, Jul. 2003.

[25] L. Ma and K. Khorasani, "Adaptive structure feed-forward neural networks using polynomial activation functions," in *Proc. SPIE*, vol. 4055, 2000, pp. 120–129.

[26] ——, "Constructive hermite polynomial feedforward neural networks with application to facial expression recognition," in *Proc. SPIE*, vol. 4520, 2001, pp. 31–42.

[27] ——, "Application of adaptive constructive neural networks to image compression," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1112–1126, Sep. 2002.

[28] ——, "A new strategy for adaptively constructing multilayer feedforward neural networks," *Neurocomput.*, vol. 51, pp. 361–385, 2003.

[29] ——, "Facial expression recognition using constructive feedforward neural networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 3, pp. 1588–1595, Jun. 2004.

[30] ——, "New training strategies for constructive neural networks with application to regression problems," *Neural Netw.*, vol. 17, no. 4, pp. 589–609, 2004.

[31] L. Ma, K. Khorasani, and M. R. Azimi-Sadjadi, "Adaptive constructive neural networks using hermite polynomials for compression of still and moving images," in *Proc. SPIE*, vol. 4739, 2002, pp. 108–119.

[32] G. Pilato, F. Sorbello, and G. Vassallo, "Using the hermite regression algorithm to improve the generalization capability of a neural network," in *Proc. 11th Italian Workshop on Neural Nets*, Salerno, Italy, 1999, pp. 296–301.

[33] L. Prechelt, "PROBEN1-A set of neural network benchmark problems and benchmarking rules," Electrotech. Lab., Dept. Informatics, Univ. Karlsruhe, Karlsruhe, Germany, Tech. Rep. 21/94, Sep. 1994.

[34] ——, "Investigation of the cascor family of learning algorithms," *Neural Netw.*, vol. 10, no. 5, pp. 885–896, 1997.

[35] A. I. Rasiah, R. Togneri, and Y. Attikiouzel, "Modeling 1-d signals using hermite basis functions," in *Proc. Inst. Elect Eng. Vis. Image Signal Process*, vol. 144, 1997, pp. 345–354.

[36] R. Setiono and L. C. K. Hui, "Use of a quasi-Newton method in a feedforward neural network construction algorithm," *IEEE Trans. Neural Netw.*, vol. 6, no. 1, pp. 273–277, Jan. 1995.

[37] W. Weng and K. Khorasani, "An adaptive structural neural network with application to eeg automatic seizure detection," *Neural Netw.*, vol. 9, no. 7, pp. 1223–1240, 1996.

Liying Ma received the B.S. degree from Northeastern University, Shengyang, China, in 1984, the M.S. degree from Hiroshima University, Higashi-Hiroshima, Japan, in 1991, and the Ph.D. degree from Concordia University, Montreal, QC, Canada, in 2001.

She was with the Automation Research Institute of Ministry of Metallurgical Industry, Beijing, China, as a Research Electrical Engineer, from 1984 to 1988. From 1991 to 1992, she worked as a Systems Engineer at Yokogawa Engineering Service Co. Tokyo, Japan. In 2002, she was a Scientific Engineer at the University of British Columbia, Canada. She has been a Postdoctoral Fellow at Concordia University since 2003. Her main areas of research interests are in neural networks and their applications to image processing and pattern recognition, bioinformatics, optimization and intelligent systems, and digital signal processing and applications.

K. Khorasani (M'85) received the B.S., M.S., and Ph.D. degrees in electrical and computer engineering from the University of Illinois, Urbana-Champaign, in 1981, 1982, and 1985, respectively.

From 1985 to 1988, he was an Assistant Professor at the University of Michigan, Dearborn, and since 1988, he has been at Concordia University, Montreal, Canada, where he is currently a Professor in the Department of Electrical and Computer Engineering. His main areas of research are in nonlinear and adaptive control, modeling and control of flexible link/joint manipulators, intelligent and autonomous systems, neural network applications to pattern recognition, robotics and control, adaptive structure neural networks, fault diagnosis, isolation and recovery, and distributed and collaborative force feedback of haptic interfaces in virtual environments. He has authored/co-authored over 200 publications in these areas.