Function Approximation Using Generalized Adalines

Jiann-Ming Wu, Zheng-Han Lin, and Pei-Hsun Hsu

***Abstract*—This paper proposes neural organization of generalized adalines (gadalines) for data driven function approximation. By generalizing the threshold function of adalines, we achieve the K-state transfer function of gadalines which responds a unitary vector of K binary values to the projection of a predictor on a receptive field. A generative component that uses the K-state activation of a gadaline to trigger K posterior independent normal variables is employed to emulate stochastic predictor-oriented target generation. The fitness of a generative component to a set of paired data mathematically translates to a mixed integer and linear programming. Since consisting of continuous and discrete variables, the mathematical framework is resolved by a hybrid of the mean field annealing and gradient descent methods. Following the leave-one-out learning strategy, the obtained learning method is extended for optimizing multiple generative components. The learning result leads to parameters of a deterministic gadaline network for function approximation. Numerical simulations further test the proposed learning method with paired data oriented from a variety of target functions. The result shows that the proposed learning method outperforms the MLP and RBF learning methods for data driven function approximation.**

***Index Terms*—Adalines, generative models, mean field annealing, perceptron, postnonlinear projection, potts encoding, supervised learning.**

I. Introduction

ADALINES (adaptive linear elements) of Widrow [22] have been widely employed to develop neural networks [1], [8], [13] for solving tasks of classification, noise cancellation, system identification, and signal prediction [3], [17], [23], [10]. An adaline is composed of a receptive field and a threshold function. When stimulated by an input or a predictor, $\mathbf{x} \in R^d$, it apples its transfer function to the projection of the input on a receptive field to form a bipolar output. A typical adaline network is organized to sum up weighted activations of all its adalines, and is mathematically expressed by a network function in terms of a set of adaptable parameters, including the receptive fields and posterior weights. As the design cost is measured by the mean square error between the desired and estimated targets, learning an adaline network subject to a set of paired data translates to an unconstrained optimization, which aims to minimize the design cost by optimizing the network parameters. Since the transfer function of an adaline is nonlinear, the network function of adalines is beyond the scope spanned by linear regressions.

Manuscript received December 16, 2004; revised May 31, 2005.

J.-M. Wu is with the Department of Applied Mathematics, National Dong Hwa University, Hualien 941, Taiwan (e-mail: jmwu@mail.ndhu.edu.tw).

Z.-H. Lin is with the Department of Computer Science and Information Engineering, MingDao University, ChangHua 523, Taiwan.

P.-H. Hsu is with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 100, Taiwan.

Digital Object Identifier 10.1109/TNN.2006.873284

This paper explores data driven function approximation [5] based on generalized adalines. The novel neural organization is devised by generalizing the threshold function to K-state transfer function, which transforms its input to a K-state activation, represented by elements in $\Xi_K = \{\mathbf{e}_1^K, \ldots, \mathbf{e}_K^K\}$, where $\mathbf{e}_k^K$ is a unitary vector with the kth bit one and the others zero. A K-state transfer function uses K built-in knots to partition its domain to K nonoverlapping intervals so as to represent the exclusive membership of its input to K nonoverlapping intervals by a K-state activation. By replacing the threshold function of an adaline with a K-state transfer function, we have the generalized adaline (gadaline) for constructing novel neural networks.

Internal representations based on K-state activations impact on organizing and learning neural networks for data driven function approximation. Relevant issues are explored by addressing stochastic modeling of predictor-oriented target generation using gadalines. Following the idea, the K-state activation of a gadaline in response to a predictor is employed to trigger one of K independent normal variables or generators to produce an instance in approximating the desired target. The obtained generative component is organized to perform consecutive operations of projecting the predictor on a receptive field, encoding the projection to a K-state activation, and triggering a correspondent generator to produce an instance. In response to a predictor, the output of a generative component is mathematically expressed by the inner product of a K-state activation and a vector of instances produced by K generators. By tracking these primitive operations, we express the conditional probability density function (pdf) of the output of a generative component to the predictor in terms of a set of adaptable parameters, including the receptive field, the knot vector, and the mean vector of K posterior generators. Fitting a generative component to paired data induces a mixed integer and linear programming. Since consisting of continuous and discrete variables, the mathematical framework is resolved by a hybrid of the mean field annealing and gradient descent methods, which leads to an effective learning method for optimizing a generative component subject to paired data.

The model of emulating predictor-oriented target generation is extended to consist of multiple generative components, where the model output is set to the sum of outputs of multiple generative components. In this work, learning multiple generative components is decomposed to subtasks of simultaneously learning individual generative components following the leave-one-out learning strategy. It is suggested to sequentially optimize generative components one by one, and that whenever an individual component is updated, the parameters of the others are considered as constant. Following the strategy, a generative component is optimized to have an output that compensates for difference between the desired model output and the sum of outputs of the remaining components.

1045-9227/$20.00 © 2006 IEEE

A deterministic gadaline network is organized to sum up weighted activations of all its gadalines. Its output refers to the sum of output expectations of multiple generative components, as each of its posterior weight vectors corresponds to the mean vector of generators of a generative component. Calculating the output expectation of a generative component involves with consecutive operations of projecting a predictor on a receptive field, encoding the projection by a K-state transfer function and multiplying the K-state activation to a posterior weight vector. As the last two operations relates to realization of a postnonlinear (PNL) function, the three consecutive operations perform a cascaded composition of a linear projection and a PNL function, which is termed as a PNL projection here. By the terminology, a deterministic network of M gadalines is said to sum up M PNL projections.

There exist several remarkable properties that could be used to illustrate the difference between the gadaline network and the approximating networks based on multilayer perceptrons (MLP) and radial basis functions (RBF).

1) A gadaline network uses K-state activations as internal representations. Since the threshold function of an adaline is a special case of the K-state transfer function, the network function of multilayer adalines or perceptrons belongs to the function space spanned by gadaline networks.
2) Like a network of multilayer adalines or perceptrons, a gadaline network has a set of receptive fields that form a basis for linear transformation of dimensionality reduction, as the number of gadalines in a gadaline network is less than the dimension of predictors. This makes the gadaline network more suitable for function approximation with high dimensional predictors in comparisons with the RBF network and Gauss RBF network [19].
3) Based on the K-state transfer function instead of the threshold function, the PNL functions embedded within a gadaline network are more flexible and general for function approximation when compared with those within the MLP network.
4) The RBF network, e.g., the Gauss RBF network of Sanner and Slotine [19], is apparently different from the MLP network and the gadaline network proposed in this work.
   a) Unlike the MLP network, the Gauss RBF network sums up weighted radial basis functions instead of projective basis functions. In contrast, the proposed gadaline network performs a mapping that sums up multiple postnonlinear projections instead of weighted radial basis functions.
   b) In the Gauss RBF network [19], the regular partition to its input space R^d involves with a Cartesian product of d sets of equally spaced knots for determining centers of RBF functions. However, a gadaline network only addresses on an irregular partition to the domain of each of its K-state transfer functions; it does not make use of a Cartesian product of d sets of regular or irregular knots for internal representations.
   c) The parameters of a gadaline network are $Md + M(2K + 1)$ in number which linearly depends on the input dimension d, state number K and gadaline number M. But the parameter number in a Gauss RBF network is proportional to nonpolynomial K^d that increases intolerably for relatively large K and d.

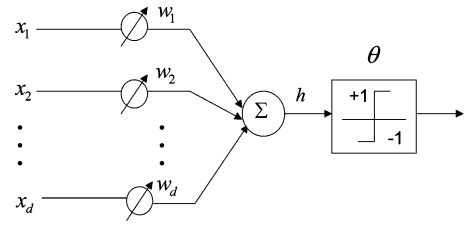

Fig. 1. Adaptive linear element (Adaline).

This paper is organized as follows. Section II introduces neural organization of gadalines based on the K-state transfer function generalized from the threshold function of adalines. In Section III, a generative component, which consists of a gadaline as well as K posterior generators, is organized to realize stochastic modeling of predictor-oriented target generation, and its learning to paired data is explored based on a hybrid of mean field annealing and gradient descent methods. In Section IV, the model for emulating predictor-oriented target formation is extended to comprise multiple generative components, and the relevant learning method is addressed following the leave-one-out learning strategy. Section V gives numerical studies on data driven function approximation by gadaline networks in comparison with MLP networks and RBF networks. The conclusions of this work are given in Section VI.

II. Neural Organization of Gadalines

A. Adalines

As shown in Fig. 1, an adaline is composed of a receptive field $\mathbf{w}$ and a threshold function $\theta(\cdot)$. When stimulated by a predictor, $\mathbf{x}[t] = (x_1[t], \ldots, x_d[t])^T$, it projects the input on its receptive field, $\mathbf{w} = (w_1, \ldots, w_d)^T$, to form an external field expressed by

$$h[t] = \mathbf{w}^T \mathbf{x}[t] \tag{1}$$

then employ the following threshold function to encode the external field:

$$\theta(h[t]) = \begin{cases} 1, & \text{if } h[t] \geq 0 \\ -1, & \text{otherwise.} \end{cases} \tag{2}$$

To facilitate our presentations, we set $x_d[t]$ to one for all t so as to represent an arbitrary hyperplane in R^{d-1} by (1) in the following contexts.

A stochastic threshold function is defined to have a stochastic output in response to an external field h. Let s denote a discrete random variable corresponding to the output of a stochastic threshold function. Since s depends on the external field, the conditional pdf of s to h is assumed proportional to $\exp(\beta h s)$, such as

$$\Pr(s|h) \propto \exp(\beta h s)$$

where β is a positive parameter for modulating randomness. Since the outcome of s is bipolar, by normalization, we have the following conditional pdf:

$$\Pr(s|h) = \frac{\exp(\beta h s)}{\exp(\beta h) + \exp(-\beta h)}.$$

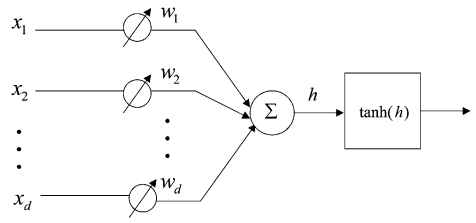

Fig. 2. Perceptron.

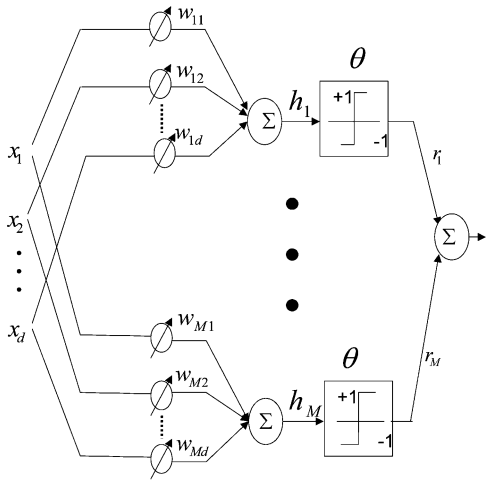

Fig. 3. Typical adaline network.

When $h = h[t]$, the expectation of s is expressed as follows:

$$\begin{aligned} g(h[t]) &\equiv \langle s | h = h[t] \rangle \\ &= \Pr(s = 1 | h = h[t]) - \Pr(s = -1 | h = h[t]) \\ &= \frac{\exp(\beta h[t]) - \exp(-\beta h[t])}{\exp(\beta h[t]) + \exp(-\beta h[t])} \\ &= \tanh(\beta h[t]). \end{aligned} \tag{3}$$

For sufficiently small β, the expectation approaches to zero. In the occasion, the output is generated with maximal randomness and its expectation tends to be independent of the external field. On the other hand, for sufficiently large β, the g function asymptotically approaches the threshold function of (2), where the output becomes a deterministic result of the external field. When $\beta = 1$, the output expectation is identical to the output of a perceptron [13], [18] shown in Fig. 2.

A typical adaline network synchronously transmits its signals feed-forward the multilayer structure shown in Fig. 3. A network of M adalines is mathematically expressed by

$$F(\mathbf{x}[t]) = \sum_{m=1}^{M} r_m \theta\left(\mathbf{w}_m^T \mathbf{x}[t]\right) \tag{4}$$

where r_m denotes the posterior weight, quantifying the connecting strength from an adaline to the output unit, and the network parameter contains all the receptive fields and posterior weights. By replacing the threshold function θ in (4) with the hypertangent function of (3), we have the network function for multilayer perceptrons [13], [21], [20].

Subject to a set of paired data, $\{(\mathbf{x}[t], y[t])\}_{t=1}^{N}$, function approximation using an adaline network specifies a task of

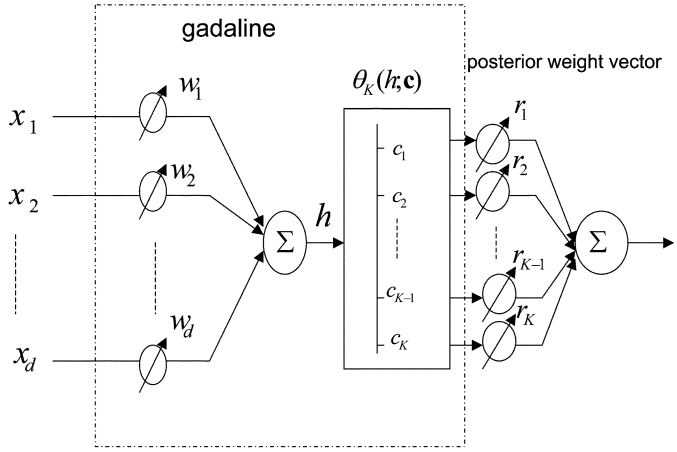

Fig. 4. Weighted gadaline.

optimizing the network parameter via minimizing the following design cost:

$$D(\{\mathbf{w}_m\}, \{r_m\}) = \frac{1}{N} \sum_{t=1}^{N} \|y[t] - F(\mathbf{x}[t])\|^2. \tag{5}$$

B. Gadalines

A gadaline is organized to consist of a receptive field and a K-state function as shown in Fig. 4. A K-state function uses K built-in knots, denoted by $\mathbf{c} = (c_1, c_2, \ldots, c_K)^T$, to partition its domain into K nonoverlapping intervals, each denoted by

$$I_k = \left(\frac{c_k + c_{k-1}}{2}, \frac{c_k + c_{k+1}}{2} \right], \qquad \text{for } k = 1, \ldots, K$$

where $c_0 = c_{\min}$ and $c_{K+1} = c_{\max}$ are two auxiliary knots for boundary conditions, and $c_k < c_{k+1}$ for all k. It follows:

$$\cup_{k=1}^{K} I_k = (c_{\min} c_{\max}]$$

and $I_k \cap I_l = \oslash$ for $k \neq l$. The K-state activation in response to an external field $h[t]$ is a unitary vector belonging the set Ξ_K, representing the exclusive membership of $h[t]$ to one of K nonoverlapping intervals. A K-state transfer function with knot vector $\mathbf{c}$ is expressed by

$$\theta_K(h[t]; \mathbf{c}) = \mathbf{e}_k^K, \qquad \text{if } h[t] \in I_k. \tag{6}$$

A gadaline employs adaptable knots to form an irregular partition to the domain of its K-state transfer function, equivalently the range of the projection expressed by $h[t] = \mathbf{w}^T \mathbf{x}[t]$. It is notable that the external field $h[t]$ is formed by projecting the input on a receptive field instead of measuring distance between $\mathbf{x}[t]$ and a center as in a RBF network. Therefore, the gadaline network presented in this work is different from the Gauss RBF network in [19], where each dimension of the input space is partitioned by a set of equally spaced knots and the Cartesian product of d sets of knots is employed to determine centers of K^d radial basis functions.

The threshold function of an adaline is a special case of the K-state function. By setting $K = 2$ and $\mathbf{c} = (-1 \;\; 1)^T$, we have

$$\theta(h[t]) = \mathbf{c}^T \theta_2(h[t]; \mathbf{c})$$

which implies that an adaline network can be emulated using gadalines.

A stochastic K-state transfer function is said to have a stochastic output in response to an external field. Let $\boldsymbol{\delta}$ denote a unitary random vector with possible outcomes belonging to Ξ_K. Assume that the conditional pdf of $\boldsymbol{\delta}$ to an external field h obeys the following expression:

$$\Pr(\boldsymbol{\delta} = \mathbf{e}_k^K | h) \propto \exp(-\beta \|h - c_k\|^2).$$

By normalization, the conditional pdf is rewritten as

$$\Pr(\boldsymbol{\delta} = \mathbf{e}_k^K | h) = \frac{\exp(-\beta \|h - c_k\|^2)}{\sum_l \exp(-\beta \|h - c_l\|^2)}.$$

Let $g_K(h[t]; \mathbf{c}, \beta)$ be a vector function whose output denotes the expectation of $\boldsymbol{\delta}$ conditional to $h = h[t]$. Then

$$\begin{aligned} g_K(h[t]; \mathbf{c}, \beta) &= \sum_{k=1}^{K} \mathbf{e}_k^K \Pr\left(\boldsymbol{\delta} = \mathbf{e}_k^K | h = h[t]\right) \\ &= \left(\frac{\exp(-\beta \|h[t] - c_1\|^2)}{\sum_{l=1}^{K} \exp(-\beta \|h[t] - c_l\|^2)}, \ldots, \right. \\ &\quad \times \frac{\exp(-\beta \|h[t] - c_k\|^2)}{\sum_{l=1}^{K} \exp(-\beta \|h[t] - c_l\|^2)}, \ldots, \\ &\quad \left. \times \frac{\exp(-\beta \|h - c_K\|^2)}{\sum_{l=1}^{K} \exp(-\beta \|h - c_l\|^2)} \right)^T \end{aligned} \tag{7}$$

which exactly coincides with the normalized response of a Gaussian array in the field of computational neuroscience [7], or the hidden function with one-dimensional inputs in [24]. Again, the external field to g_K in a gadaline network is the projection of $\mathbf{x}[t]$ on a receptive field instead of the distance between $\mathbf{x}[t]$ and a center, as in the work [24].

A Gaussian array consists of K Gaussian units, each having its own Gaussian-like tuning curve in response to the common feature h

$$f\left(h; c_k, \sigma_h^2\right) = \frac{1}{\sqrt{2\pi}\sigma_h} \exp\left(-\frac{\|h - c_k\|^2}{2\sigma_h^2}\right) \tag{8}$$

where the preference c_k denotes the center of the Gaussian window and σ_h^2 denotes the common width. As shown in Fig. 5, a Gaussian array has a selective response to the common feature; the most responsive Gaussian unit possesses a preference closest to the common feature. As the external field h to a stochastic K-state function relates to the common feature for a Gaussian array, the knot c_k corresponds to the preference of a Gaussian unit.

The β parameter in function g_K of (7) plays the same role as in function g of (3). For sufficiently large β, g_K approaches θ_K. The function g can be represented by g_K with $K = 2, \mathbf{c} = (-11)^T$ as follows:

$$g(h[t]) = \mathbf{c}^T g_2(h[t]; \mathbf{c}, \beta)$$

which implies that the transfer function of the perceptron can be emulated by a special case of g_K.

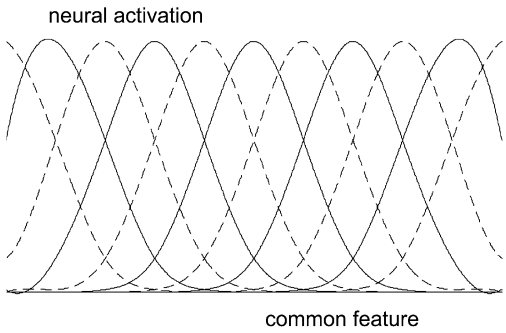

Fig. 5. Array of Gaussian tuning curves.

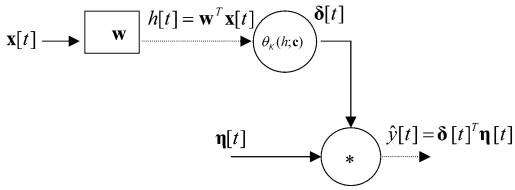

Fig. 6. Generative component.

TABLE I
FOUR CONSECUTIVE OPERATIONS WITHIN A GENERATIVE COMPONENT

	operation	transformation	parameter
1	project a predictor on a receptive field	$h[t] = \mathbf{w}^T \mathbf{x}[t]$	$\mathbf{w}$
2	encode an external field	$\boldsymbol{\delta}[t] = \theta_K(h[t]; \mathbf{c})$	$\mathbf{c}$
3	trigger K generators	$\eta_k[t] \sim N(r_k, \sigma_y^2)$	$\mathbf{r}$
4	select one instance	$\widehat{y}[t] = \boldsymbol{\delta}[t]^T \boldsymbol{\eta}[t]$	-

III. STOCHASTIC MAPPING BY A GENERATIVE COMPONENT

A. A Generative Component

The generative component shown in Fig. 6 consists of a gadaline as well as K posterior independent normal variables, where the K-state activation of a gadaline is employed to select a generator to produce an output instance. The consecutive operations of a generative component for emulating predictor-oriented target generation are shown in Table I. By tracking these primitive operations, we are able to figure out the conditional pdf of the output of a generative component to a predictor.

The first two operations in Table I deterministically translate a predictor $\mathbf{x}[t]$ to the K-state activation $\boldsymbol{\delta}[t]$, where $\boldsymbol{\delta}[t]$ is obtained by applying a K-state transfer function to the external field $h[t]$. The third operation uses K independent normal variables, each with its own mean r_k, to produce K instances, collectively denoted by vector $\boldsymbol{\eta}[t]$. According to the activation given by the second operation, the last operation selects one of K instances as the response to $\mathbf{x}[t]$. The selection is simply expressed by

$$\widehat{y}[t] = \boldsymbol{\delta}[t]^T \boldsymbol{\eta}[t] = \theta_K(\mathbf{w}^T \mathbf{x}[t]; \mathbf{c})^T \boldsymbol{\eta}[t].$$

The above expression relates the output of a generative component to a conditional mixture of K independent normal variables, as the K-state activation $\boldsymbol{\delta}[t]$ refers to a condition for

selection of K independent normal variables. The expression figures out the predictor-oriented target generation emulated by a generative component, which facilitates organizing and learning deterministic gadaline networks based on internal representations using K-state activations.

Since $\widehat{y}[t]$ is an instance of a normal variable determined by $\boldsymbol{\delta}[t]$, in case of $h[t] \in I_k$, equivalently $\boldsymbol{\delta}[t] = \mathbf{e}_k^K$, the conditional pdf of the component output to $h = h[t]$ is the same as the pdf of the selected normal variable. That is

$$q(y|h = h[t]) = f\left(y; r_k, \sigma_y^2\right), \text{ if } \boldsymbol{\delta}[t] = \mathbf{e}_k^K \text{ or } h[t] \in I_k \quad (9)$$

which takes the following algebraic form:

$$q(y|h = h[t]) = \sum_{k=1}^{K} \theta_K(h[t]; \mathbf{c})^T \mathbf{e}_k^K f\left(y; r_k, \sigma_y^2\right) \quad (10)$$

following the definition in (6).

B. Fitting and Learning

The parameter of a generative component consists of a receptive field, knot vector, mean vector, and variance, respectively represented by $\mathbf{w}, \mathbf{c}, \mathbf{r} = (r_1 r_2 \cdots r_K)^T$, and σ_y^2. For fixed $\mathbf{w}$, $\mathbf{x}[t]$ is deterministically transformed to $h[t]$ by the first operation. The pair, $(h[t], y[t])$, given by the first operation constrains that the desired output of the last three operations in response to $h[t]$ should be as close as possible to $y[t]$. Therefore, the fitness of the conditional pdf in (10) to $(h[t], y[t])$ for all t is measured by the following averaged log likelihood:

$$\begin{aligned} l &= \frac{1}{N} \log \prod_t q(y[t]|h = h[t]) \\ &= \frac{1}{N} \sum_t \log \sum_k \boldsymbol{\delta}[t]^T \mathbf{e}_k^K f\left(y[t]; r_k, \sigma_y^2\right) \\ &= \frac{1}{N} \sum_t \log \sum_k \delta_k[t] f\left(y[t]; r_k, \sigma_y^2\right) \\ &= \frac{1}{N} \sum_t \sum_k \delta_k[t] \log f\left(y[t]; r_k, \sigma_y^2\right) \end{aligned} \quad (11)$$

where $\delta_k[t]$ in the third line denotes the kth element of $\boldsymbol{\delta}[t]$. l serves as a criterion for optimizing parameters of a generative component.

Consider encoding $h[t]$ to $\boldsymbol{\delta}[t]$ by the second operation. Since the K-state activation $\boldsymbol{\delta}[t]$ is a deterministic result of $h[t]$

$$H_k = \left\{h[t] | \boldsymbol{\delta}[t] = \mathbf{e}_k^K\right\}$$

can be uniquely obtained by collecting all $h[t]$ that are encoded to the same activation, e.g., $\mathbf{e}_k^K$. Following the Gaussian-like tuning curve assumption in (8), it is assumed that the pdf underlying H_k is a normal pdf with mean c_k and variance σ_h^2. The assumption leads to the following averaged log likelihood which measures the fitness of a normal pdf to elements in H_k:

$$l'_k = \frac{1}{N_k} \log \prod_{t:\boldsymbol{\delta}[t] = \mathbf{e}_k^K} f\left(h[t]; c_k, \sigma_h^2\right)$$

where N_k denotes the size of H_k. The weighting sum of l'_k over all k becomes

$$\begin{aligned} l' &= \sum_k \frac{N_k}{N} l'_k \\ &= \frac{1}{N} \sum_k \sum_{t:\boldsymbol{\delta}[t] = \mathbf{e}_k^K} \log f\left(h[t]; c_k, \sigma_h^2\right) \\ &= \frac{1}{N} \sum_k \sum_t \boldsymbol{\delta}[t]^T \mathbf{e}_k^K \log f\left(h[t]; c_k, \sigma_h^2\right) \\ &= \frac{1}{N} \sum_t \sum_k \delta_k[t] \log f\left(h[t]; c_k, \sigma_h^2\right). \end{aligned} \quad (12)$$

The best fitness of a generative component to paired data, $(\mathbf{x}[t], y[t])$ for all t, summarizes to minimize

$$\begin{aligned} &E(\{\boldsymbol{\delta}[t]\}, \mathbf{w}, \mathbf{c}, \mathbf{r}, \boldsymbol{\sigma}) \\ &\quad = -\frac{1}{2}(l + l') \\ &\quad = \frac{1}{2N\sigma_h^2} \sum_t \sum_k \delta_k[t] \|\mathbf{w}^T \mathbf{x}[t] - c_k\|^2 \\ &\qquad + \frac{1}{2N\sigma_y^2} \sum_t \sum_k \delta_k[t] \|y[t] - r_k\|^2 \end{aligned} \quad (13)$$

subject to

$$\begin{aligned} &\sum_k \delta_k[t] = 1, \qquad \text{for all } t \\ &\delta_k[t] \in \{0, 1\}, \qquad \text{for all } t, k \end{aligned} \quad (14)$$

where the constant in (13) has been neglected and $\boldsymbol{\sigma}$ denotes collection of σ_h and σ_y.

Since the mathematical framework contains both discrete and continuous variables, the objective function E is not differentiable with respect to the discrete variable $\delta_k[t]$. It is resolved by a hybrid of the mean field annealing (MFA) and gradient descent methods.

Relating vector $\boldsymbol{\delta}[t]$ to a Potts variable [15] automatically releases the unitary constraint in (14). By doing this, we have a Potts system, $\Lambda = \{\boldsymbol{\delta}[t]\}_t$. Under the Boltzmann assumption [15], it follows:

$$\Pr(\Lambda) \propto \exp(-\beta E(\Lambda; \mathbf{w}, \mathbf{c}, \mathbf{r}, \boldsymbol{\sigma}))$$

where β denotes the inverse of a temperature like parameter, and the notation $E(\Lambda; \mathbf{w}, \mathbf{c}, \mathbf{r}, \boldsymbol{\sigma})$ means that the continuous variables, $\mathbf{w}, \mathbf{c}, \mathbf{r}$, and $\boldsymbol{\sigma}$ in (13) are fixed. The optimal configuration of such Potts system dominates the Boltzmann distribution as β increases asymptotically. It follows:

$$\lim_{\beta \to \infty} \Pr(\Lambda^*) = 1$$

where

$$E(\Lambda^*; \mathbf{w}, \mathbf{c}, \mathbf{r}, \boldsymbol{\sigma}) = \min_{\Lambda} E(\Lambda; \mathbf{w}, \mathbf{c}, \mathbf{r}, \boldsymbol{\sigma}).$$

The MFA method tracks the mean configuration of a Potts system under an annealing process, where the β parameter is gradually increased from a sufficiently low value to large one.

For each β value, the MFA method estimates the mean configuration by iteratively executing mean field equations toward a stationary point. The mean configuration obtained at a β value is set to be the initial mean configuration at the subsequent β value. The mean field equations can be derived from the following free energy [15]:

$$\psi(\mathbf{v},\mathbf{u}) = E(\mathbf{v};\mathbf{w},\mathbf{c},\mathbf{r},\boldsymbol{\sigma}) + \sum_t\sum_k v_k[t]u_k[t] - \frac{1}{\beta}\sum_t \log[\sum_k \exp(\beta u_k[t])] \tag{15}$$

where $v_k[t]$ denotes the mean of $\delta_k[t]$, $u_k[t]$ is an auxiliary variable, and $\mathbf{v}$ and $\mathbf{u}$, respectively, denote the collection of all $v_k[t]$ and $u_k[t]$.

By setting

$$\begin{cases} \frac{\partial\psi(\mathbf{v},\mathbf{u})}{\partial v_k[t]} = 0 \\ \frac{\partial\psi(\mathbf{v},\mathbf{u})}{\partial u_k[t]} = 0 \end{cases} \quad \text{for all } t \text{ and } k$$

we have the following mean field equation which characterizes the saddle point of the free energy (15):

$$\begin{aligned} u_k[t] &= \frac{\partial E(\mathbf{v};\mathbf{w},\mathbf{c},\mathbf{r},\boldsymbol{\sigma})}{\partial v_k[t]} \\ &= \frac{1}{2N\sigma_h^2}\|\mathbf{w}^T\mathbf{x}[t] - c_k\|^2 \\ &\quad + \frac{1}{2N\sigma_y^2}\|y[t] - r_k\|^2 \end{aligned} \tag{16}$$

$$v_k[t] = \frac{\exp(-\beta u_k[t])}{\sum_{l=1}^{K}\exp(-\beta u_l[t])}. \tag{17}$$

It is suggested by a hybrid of the mean field annealing and gradient descent methods [24] to seek for $\mathbf{w},\mathbf{c},\mathbf{r}$, and $\boldsymbol{\sigma}$ by minimizing the quantity $E(\mathbf{v},\mathbf{w},\mathbf{c},\mathbf{r},\boldsymbol{\sigma})$ directly, where $E(\mathbf{v},\mathbf{w},\mathbf{c},\mathbf{r},\boldsymbol{\sigma})$ is obtained by replacing each discrete $\delta_k[t]$ in (13) with continuous $v_k[t]$ for each β value. By setting

$$\begin{cases} \frac{\partial E(\mathbf{v},\mathbf{w},\mathbf{c},\mathbf{r},\boldsymbol{\sigma})}{\partial c_k} = 0 \\ \frac{\partial E(\mathbf{v},\mathbf{w},\mathbf{c},\mathbf{r},\boldsymbol{\sigma})}{\partial r_k} = 0 \\ \frac{\partial E(\mathbf{v},\mathbf{w},\mathbf{c},\mathbf{r},\boldsymbol{\sigma})}{\partial \sigma_h} = 0 \\ \frac{\partial E(\mathbf{v},\mathbf{w},\mathbf{c},\mathbf{r},\boldsymbol{\sigma})}{\partial \sigma_y} = 0 \end{cases}$$

we have the following rules for determining elements of $\mathbf{c},\mathbf{r}$, and $\boldsymbol{\sigma}$, respectively:

$$c_k = \frac{\sum_t v_k[t]h[t]}{\sum_t v_k[t]} \tag{18}$$

$$r_k = \frac{\sum_t v_k[t]y[t]}{\sum_t v_k[t]} \tag{19}$$

$$\sigma_h = \left(\frac{1}{N}\sum_t\sum_k \delta_k[t]\|\mathbf{w}^T\mathbf{x}[t] - c_k\|^2\right)^{1/2} \tag{20}$$

$$\sigma_y = \left(\frac{1}{N}\sum_t\sum_k \delta_k[t]\|y[t] - r_k\|^2\right)^{1/2}. \tag{21}$$

The two variances σ_h^2 and σ_y^2 are also treated as adaptive parameters instead of predefined constants. Setting

$$\frac{\partial E(\mathbf{v},\mathbf{c},\mathbf{r},\mathbf{w},\boldsymbol{\sigma})}{\partial \mathbf{w}} = 0$$

leads to

$$\sum_k\sum_t v_k[t](\mathbf{w}^T\mathbf{x}[t] - c_k)\mathbf{x}[t] = 0$$

which takes the following vector form:

$$\mathbf{A}\mathbf{w} = \mathbf{b}$$

with elements given by

$$\begin{aligned} A_{ji} &= \sum_t x_j[t]x_i[t], \\ b_j &= \sum_k c_k\left(\sum_t v_k[t]x_j[t]\right). \end{aligned}$$

Let $\mathbf{A}^+$ denote the pseudo inverse of $\mathbf{A}$. In general, the linear system can be solved by

$$\mathbf{w} = \mathbf{A}^+\mathbf{b}. \tag{22}$$

Learning a generative component subject to a set of paired data is summarized by the following procedure:

1) input all $(\mathbf{x}[t], y[t])$. Set β to a sufficiently low value. Randomly initialize $\mathbf{w}$ as a unitary vector and elements in $\mathbf{c}$ and $\mathbf{r}$ uniformly distributed within $[-1\ 1]$;
2) rescale all $y[t]$ within $[-1\ 1]$; rescale all $x_i[t]$ within $[-1\ 1]$, separately for each i; set $\sigma_h = \sigma_y = 1$;
3) calculate $h[t] = \mathbf{w}^T\mathbf{x}[t]$ for all t;
4) determine all $v_k[t]$ by (16)–(17);
5) determine $\{c_k\}, \{r_k\}, \sigma_h$, and σ_y by (18)–(21);
6) determine $\mathbf{w}$ by (22) and normalize $\mathbf{w}$ to a unitary vector;
7) increase β by an annealing schedule. If the halting condition holds, end the procedure; otherwise, go to step 3).

As the annealing process gradually increases the β parameter from a sufficiently low value to large one, each $v_k[t]$ changes from near $1/K$ to a binary value. Under the annealing process, the stability of vector $\mathbf{v}[t]$ measured by $\sum_k v_k^2[t]$ changes from near $(1/K)$ to one for each t. Therefore, the averaged stability for all $\mathbf{v}[t]$ approaches to one for a sufficiently large β. In the occasion, the procedure exits at step 7).

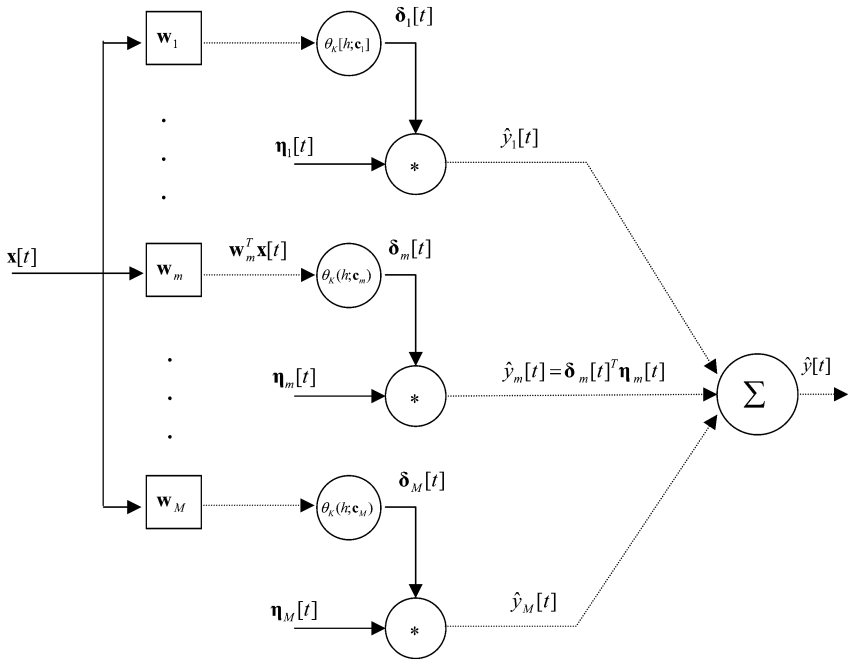

Fig. 7. Generative model with M generative components.

IV. Stochastic Mapping by Multiple Generative Components

A. A Model With Multiple Generative Components

As shown in Fig. 7, stochastic modeling of predictor-oriented target generation is extended to consist of M generative components. The model output $\widehat{y}[t]$ is expressed by

$$\widehat{y}[t] = \sum_m \widehat{y}_m[t]$$

where m indexes M generative components and $\widehat{y}_m[t]$ denotes the output of the mth generative component in response to $\mathbf{x}[t]$. The receptive field, knot vector, mean vector, and variances within the mth generative component are represented by $\mathbf{w}_m, \mathbf{c}_m, \mathbf{r}_m$, and $\sigma_h^2(m)$ and $\sigma_y^2(m)$, respectively.

Let $\mathbf{W}$ be a matrix with M rows of $\mathbf{w}_m^T, m = 1, \ldots, M$. Then

$$\mathbf{h}[t] = \mathbf{W}\mathbf{x}[t]$$

denotes a vector of external fields caused by $\mathbf{x}[t]$, with elements represented by

$$h_m[t] = \mathbf{w}_m^T\mathbf{x}[t].$$

The conditional pdf of (10) is rewritten as follows:

$$\begin{aligned} &q_m(y_m|h_m = h_m[t]) \\ &\quad = \sum_{k=1}^{K} \theta_K(h_m[t];\mathbf{c}_m)^T \mathbf{e}_k^K f\left(y_m; r_{mk}, \sigma_y^2(m)\right) \\ &\quad = f(y_m; \theta_K(h_m[t];\mathbf{c}_m)^T\mathbf{r}_m, \sigma_y^2(m)) \end{aligned} \tag{23}$$

where m runs from 1 to M. Since there is only one active bit in a K-state activation, the second line translates to that the conditional pdf of y_m to $h_m = h_m[t]$ is a normal pdf, whose mean is further expressed by $\theta_K(h_m[t];\mathbf{c}_m)^T\mathbf{r}_m$ in the third line.

Since the model output is the sum of outputs of M generative components, in case of $\mathbf{h} = \mathbf{h}[t]$, equivalently $h_m = h_m[t]$ for all m, it coincides with the sum of M normal variables, exactly a normal variable with mean $\sum_m \theta_K(h_m[t];\mathbf{c}_m)^T\mathbf{r}_m$ and variance $\sum_m \sigma_y^2(m)$. The argument leads to the following conditional pdf:

$$p(y|\mathbf{h} = \mathbf{h}[t]) = f\left(y; \sum_m \theta_K(h_m[t];\mathbf{c}_m\right)^T \mathbf{r}_m, \sum_m \sigma_y^2(m)) \tag{24}$$

which represents the stochastic mapping carried out by a model consisting of M generative components.

B. A Deterministic Gadaline Network

A typical gadaline network is organized to sum up output expectations of multiple generative components, and is mathematically expressed by

$$\begin{aligned} F_{K,M}(\mathbf{x}[t]) &\equiv \langle y|\mathbf{h} = \mathbf{W}\mathbf{x}[t]\rangle \\ &= \sum_{m=1}^{M} \theta_K\left(\mathbf{w}_m^T\mathbf{x}[t];\mathbf{c}_m\right)^T \mathbf{r}_m \end{aligned} \tag{25}$$

where the subindex indicates that the network contains M gadalines and uses K-state transfer functions. The network function sums up M weighted activations, each being the inner product of the K-state activation and posterior weight vector of a gadaline.

Since the threshold function of adalines can be realized by a two-state function, an adaline network can be mathematically expressed by a gadaline network function in (4). By setting $K = 2, \mathbf{c}_m = (-1\ 1)^T$, and $\mathbf{r}_m = (-r_m\ r_m)^T$, we have

$$\begin{aligned} F(\mathbf{x}[t]) &= \sum_{m=1}^{M} r_m \theta\left(\mathbf{w}_m^T \mathbf{x}\right) \\ &= \sum_{m=1}^{M} (-r_m r_m) \theta_2\left(\mathbf{w}_m^T \mathbf{x}; (-1\ 1)^T\right) \\ &= \sum_{m=1}^{M} \mathbf{r}_m^T \theta_2\left(\mathbf{w}_m^T \mathbf{x}; \mathbf{c}_m\right) \end{aligned}$$

which is identical to $F_{2,M}(\mathbf{x}[t])$ of (4).

Further replacing θ_K in $F_{K,M}$ with g_K of (7) leads to the following network function:

$$G_{K,M}(\mathbf{x}[t]) = \sum_{m=1}^{M} \mathbf{r}_m^T g_K(\mathbf{w}_m^T \mathbf{x}[t]; \mathbf{c}_m, \beta_m) \tag{26}$$

where the β_m parameter can be replaced with the ratio between a global β and the variance $\sigma_h^2(m)$ for each gadaline. Both the network functions $G_{K,M}$ and $F_{K,M}$ share the same network parameters. By using the g_K function, $G_{K,M}$ always behaves smoother than $F_{K,M}$.

C. Learning by Leave-One-Out Approximation

Based on the leave-one-out learning strategy [11], we present the learning method for a model consisting of multiple generative components in this subsection.

Since the model output is the sum of outputs of M generative components, the desired output $y_m[t]$ of the mth generative component in response to $\mathbf{x}[t]$ is set to compensate for the error between the desired model output $y[t]$ and the sum of outputs of the other $M-1$ generative components. That is

$$y_m[t] = y[t] - \sum_{n \neq m} \hat{y}_n[t] \tag{27}$$

where the summation term measured the model output as if the mth generative component offered no contribution to the model output. The pairs $(\mathbf{x}[t], y_m[t])$ for all t form an intermediate training set for optimizing the mth generative component based on the updating rules (16)–(22).

Under the annealing process, the desired output of the mth generative component in response to $\mathbf{x}[t]$ can be expressed in terms of mean activations, such as

$$y_m[t] = y[t] - \sum_{n \neq m} \mathbf{v}_n[t]^T \mathbf{r}_n \tag{28}$$

where the mean activation is the result tracked by the MFA method.

The learning method for M generative components is stepwise described as follows:

1) input all $(\mathbf{x}[t], y[t])$; set β to a sufficiently low value; initialize $\mathbf{W}$ at random and elements in each $\mathbf{c}_m$ and $\mathbf{r}_m$ equally spaced within $[-1\ 1]$;
2) rescale all $y[t]$ within $[-1\ 1]$; rescale all $x_i[t]$ within $[-1 1]$, separately for each i; set $\sigma_h(m) = \sigma_y(m) = 1$ for all m;
3) initialize elements of all $\mathbf{v}_m[t]$ near $1/K$ at random;
4) for each m, execute the following steps:
   a) calculate $y_m[t]$ by (28) for all t;
   b) fit the mth generative component to all $(\mathbf{x}[t], y_m[t])$ as follows:
      i) calculate $h_m[t] = \mathbf{w}_m^T \mathbf{x}[t]$ for all t;
      ii) determine $\mathbf{v}_m[t]$ by (16)–(17) for all t;
      iii) determine $\mathbf{c}_m, \mathbf{r}_m, \sigma_h(m)$, and $\sigma_y(m)$ by (18)–(21);
      iv) determine $\mathbf{w}_m$ by (22) and normalize $\mathbf{w}_m$ to a unitary vector;
5) increase β by an annealing schedule. If the halting condition holds, end the procedure; otherwise, go to step 4).

Step 4) sequentially updates multiple generative components. Whenever a generative component is updated, the parameters of the other $M-1$ generative components are considered as constant. For the currently updated generative component, the desired output in response to $\mathbf{x}[t]$ is calculated at step 4a) to compensate for the difference between the desired model output $y[t]$ and the sum of outputs of the other $M-1$ generative components. By step 4a), all $(\mathbf{x}[t], y_m[t])$ initiate a stand-alone subtask for fitting the mth generative component, constraining optimization of parameters $\{\mathbf{v}_m[t]\}, \mathbf{c}_m, \sigma_h(m), \sigma_y(m), \mathbf{r}_m$, and $\mathbf{w}_m$ by rules (16)–(22) at step 4b).

The above learning method possesses several remarkable features relative to existing MLP and RBF learning methods.

1) The mathematical framework shown in (13)–(14) formulates a novel mixed integer and linear programming for fitting the conditional pdf to paired data.
2) Since the energy function E in (13), consisting of continuous and discrete variables, is not differentiable with respect to discrete variables, the mathematical framework is resolved by a hybrid of the mean field annealing and gradient descent methods.
3) Like previous works [15], [24], the annealing process can help evolution of network parameters escaping from the trap of tremendous local minima within the energy function E.
4) Under the annealing process, the calculation of matrix inversion (22) takes part in iterative update of network parameters by interactive dynamics of (16)–(22), whereas similar matrix inversion that has been employed to determine posterior weights in a typical RBF learning method invokes no iterative updates under the mean-field-annealing process.
5) The mean field (16)–(17) employs global β and local variance $\sigma_h^2(m)$ and $\sigma_y^2(m)$ for modulating determination of the mean configuration of a Potts system, where the local variance for each m is dynamically updated under the annealing process.
6) Learning multiple generative components is decomposed to subtasks of simultaneously learning individual generative components following the leave-one-out learning strategy.

D. Approximating a Variety of Target Functions

Learning multiple generative components subject to paired data achieves network parameters for functions $F_{K,M}$ and

TABLE II
VARIETY OF TARGET FUNCTIONS

Target function	Description	Category
$y(z)=\varphi(z)$	One-dimensional functions such as tanh, cos, ln, sin, exp and polynomial functions	(d=1,M=1)
$y(\mathbf{z})=\mathbf{a}^T\mathbf{z}$	A linear projection	(d>1,M=1)
$y(\mathbf{z})=\varphi(\mathbf{a}^T\mathbf{z})$	A post-nonlinear projection	(d>1,M=1)
$y(\mathbf{z})=\sum_{m=1}^{M} b_m\varphi_m(\mathbf{a}_m^T\mathbf{z})$	A weighting sum of post-nonlinear projections	(d>1,M>1)
$y(\mathbf{z})=\prod_{m=1}^{M}\varphi_m(\mathbf{a}_m^T\mathbf{z})$	A product of post-nonlinear projections	(d>1,M>1)

$G_{K,M}$ of (25)–(26). The two network functions share the same network parameters but use different transfer functions, θ_K and g_K. With a proper β value, $G_{K,M}$ behaves smoother than $F_{K,M}$, where the suitable β value for the $G_{K,M}$ function can be determined by minimizing the design cost based on the determined network parameters.

Table II lists three categories of target functions, respectively represented by tuples of $(d=1, M=1), (d>1, M=1)$, and $(d>1, M>1)$, with discrimination according to the dimension d of predictors and the number M of PNL functions embedded within the target function.

The first type of the target function in Table II consists of a variety of differentiable one-dimensional nonlinear functions. Each of them can be approximated by the network functions, $F_{K,1}(z)$ and $G_{K,1}(z)$, derived from learning a single generative component with $d=1$ and input $z \in R$. The network function $F_{K,1}(z)$ is a spline function that sums up K piece-wise constant functions, and $G_{K,1}(z)$ represents a parametric smooth function beyond the scope of traditional quadratic or cubic spline functions.

In Table II, the target function with $d>1$ and $M=1$ is expressed by a linear or postnonlinear projection from R^d to R, which can be approximated by the network function corresponding to a generative component with receptive field $\mathbf{w}$ belonging R^d. The target function with $d>1$ and $M>1$ in Table II is more complicated; each takes form of a sum or product of multiple postnonlinear projections. The former can be directly approximated by $F_{K,M}$ or $G_{K,M}$ through learning multiple generative components, but the latter, a product expression, needs proper processes for its approximation by learning multiple generative components.

Since taking log to a product of positive postnonlinear projections induces an expression that sums up multiple postnonlinear projections, it is suggested to approximate $\ln \mathbf{y}(z)$ by the network function $F_{K,M}$ or $G_{K,M}$, then $y(\mathbf{z})$ based on the obtained network functions. The log function must have a positive domain. Taking this into account, one can represent the target function by the following combination of two rectified parts:

$$y(\mathbf{z}) \approx \mathrm{Rec}(y(\mathbf{z})) - \mathrm{Rec}(-y(\mathbf{z})) \tag{29}$$

where the rectifying function is defined by

$$\begin{aligned}\mathrm{Rec}(z) &= z, \qquad \text{if } z>0\\ &= 0, \qquad \text{otherwise.}\end{aligned}$$

TABLE III
TARGET FUNCTIONS

Target function	Description
$y_1(z)=\psi(z)+n$	One-dimensional function with uniform noise $n \in [-0.5, 0.5]$
$y_2(\mathbf{z})=0.3z_1+z_2-0.5z_3+2z_4-0.7z_5$	Single linear projection
$y_3(\mathbf{z})=\tanh(0.3z_1+z_2-0.5z_3+2z_4-0.7z_5)$	Single post-nonlinear projection
$y_4(\mathbf{z})=\tanh(0.8z_1+0.2z_2)+\tanh(0.3z_1-0.9z_2)$	Sum of two PNL projections
$y_5(\mathbf{z})=\tanh(0.8z_1+0.2z_2+0z_3+0z_4+0z_5)$ $+\tanh(0.3z_1-0.9z_2+0z_3+0z_4+0z_5)$	Sum of two PNL projections with ineffective attributes
$y_6(\mathbf{z})=0.5z_1^2-0.9z_2^2$	Quadratic function
$y_7(\mathbf{z})=\exp(-\frac{z_1^2}{\sigma_1}-\frac{z_2^2}{\sigma_2})\cos(\kappa z_1+\phi)$	Gabor function

Now both $\ln \mathrm{Rec}(y(\mathbf{z}))$ and $\ln \mathrm{Rec}(-y(\mathbf{z}))$ take form of summing up multiple postnonlinear projections, and can be approximated by learning multiple generative components separately subject to training sets of

$$\begin{aligned}S^+ &= \{(\mathbf{x}[t], \ln y[t]) | y[t]>0\}\\ S^- &= \{(\mathbf{x}[t], \ln(-y[t])) | y[t]<0\}.\end{aligned} \tag{30}$$

Let $G^+_{K,M}(\mathrm{z})$ and $G^-_{K,M}(\mathbf{z})$ denote two approximating functions respectively obtained by learning to S^+ and S^-. The final approximation to the original target function $y(\mathbf{z})$ is given by

$$y(\mathbf{z}) \approx \exp(G^+_{K,M}(\mathbf{z})) - \exp(G^-_{K,M}(\mathbf{z})) \tag{31}$$

where the two exponential terms in the right-hand side, respectively, approximate $\mathrm{Rec}(y(\mathbf{z}))$ and $\mathrm{Rec}(-y(\mathbf{z}))$ in (29).

V. NUMERICAL SIMULATIONS

Numerical simulations of learning multiple generative components are explored in this section. The proposed learning method is employed to approximate a variety of target functions in Table III in the first subsection and is compared with the relevant MLP and RBF learning methods in the second subsection.

A. Performance Evaluation

The proposed learning method is implemented by the MATLAB language on Pentium III personal computers. For each target function in Table III, the learning method is employed to estimate the network parameter subject to a training set; then, the obtained network function $G_{K,M}$ is verified by a testing set. Within the training and testing sets correspondent to each target function y_i in Table III, all $\mathbf{x}[t]$ form a uniform sample from a bounded domain of y_i and each pair $(\mathbf{x}[t], y[t])$ is obtained by setting $y[t]$ to $y_i(\mathbf{x}[t])$.

For each target function, the design and generalization costs of an approximating function are measured by the mean square errors over the correspondent training and testing sets. Table IV shows summary statistics of the mean square error and execution time derived from twenty executions of the proposed learning method for each target function.

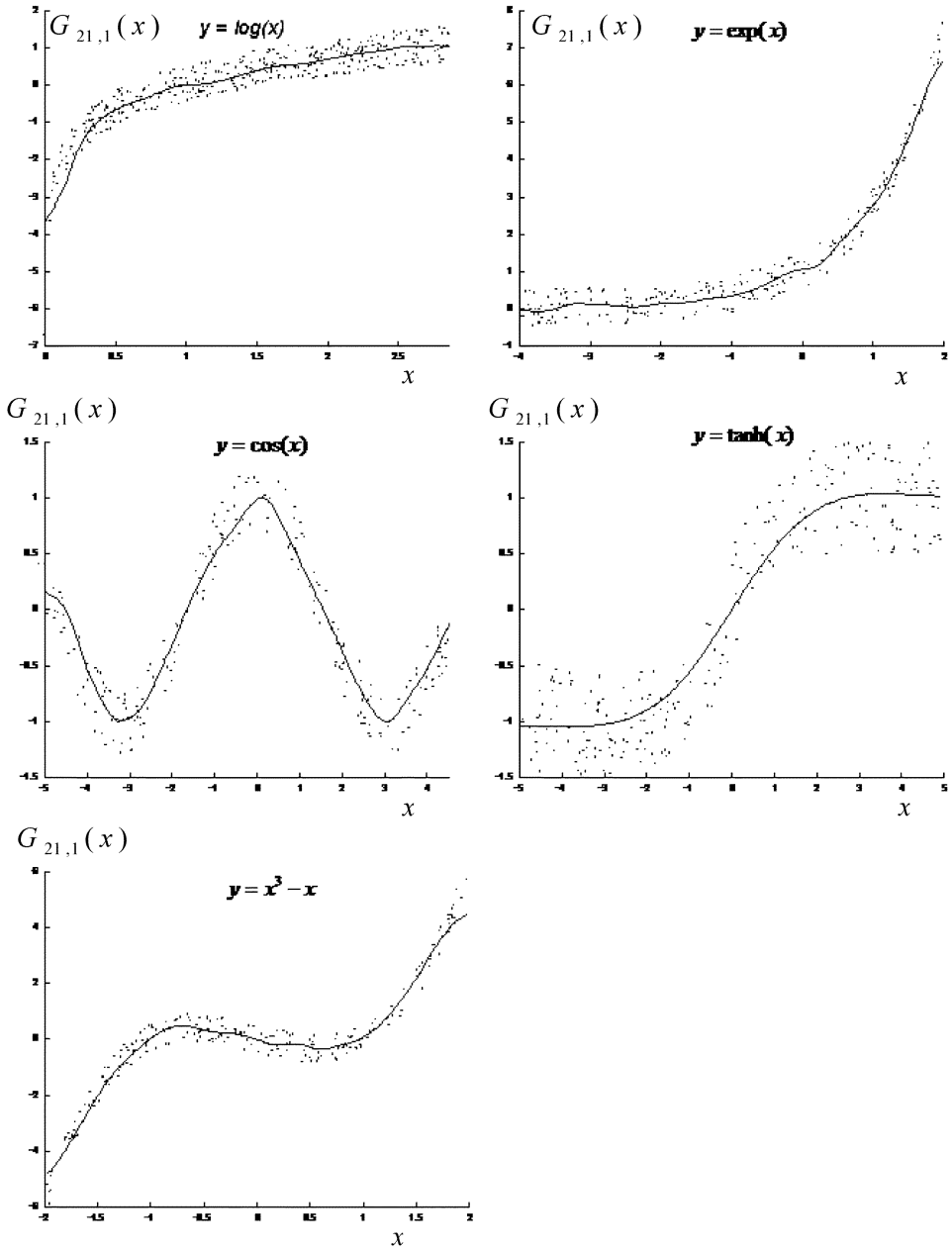

Fig. 8. Approximating functions $G_{21,1}$ for the noisy logarithm, exponential, cosine, hypertangent, and polynomial functions.

In the first example, the proposed learning method is tested with one-dimensional noisy target functions given at the first row of Table III. The training and testing sets for each target function respectively contain 400 and 200 pairs of data. Each plot in Fig. 8 shows the noisy paired data oriented from specific nonlinearity and the approximating function $G_{21,1}$ obtained by learning a generative component with $K = 21$, where the horizontal and vertical coordinates respectively measure the one-dimensional predictor and the function value. The mean squared error for each nonlinearity is shown at the first five rows of Table IV. The proposed learning method well deals with noisy targets for these examples. The derived approximating function effectively captures nonlinearity underlying each training set as shown in each plot of Fig. 8.

In the remaining experiments of this subsection, the training and testing sets for each target function respectively contain 1000 and 500 pairs of data. The target functions $y_2(\mathbf{z})$ and $y_3(\mathbf{z})$ in Table III are separately approximated by a network function $G_{21,1}$ with multidimensional predictors, since they take form of single linear and postnonlinear projections, respectively. Figs. 9(a) and 9(b) show the PNL functions embedded within the obtained approximating functions for $y_2(\mathbf{z})$ and $y_3(\mathbf{z})$, where the horizontal coordinate measures the projection on a receptive field and the vertical coordinate measures the output of the PNL function calculated by $g_K(h; \mathbf{c})^T \mathbf{r}$. The PNL function in Fig. 9(a) tends to be linear, and that in Fig. 9(b) is likely to be a hypertangent function in shape. In the approximating function, the only receptive field listed below each horizontal coordinate can be verified for its linear dependence on the projective

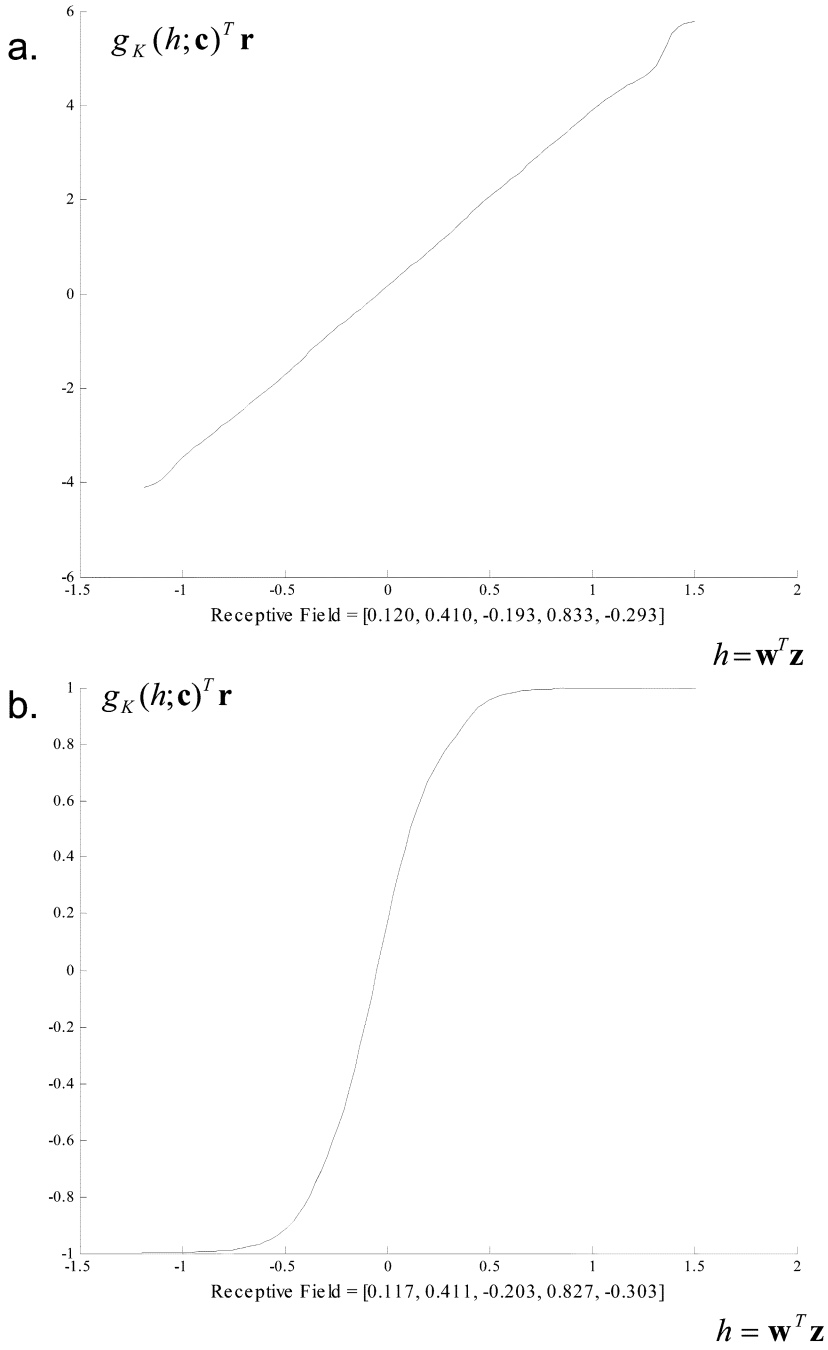

Fig. 9. Component functions for linear and nonlinear projection function.

vector in $y_2(\mathbf{z})$ or $y_3(\mathbf{z})$. Since $y_2(\mathbf{z})$ and $y_3(\mathbf{z})$ are not noisy, as shown in Table IV, the testing cost reduces to a scale below 10^{-3}. In Table IV, the standard deviation below 10^{-10} has been neglected.

The target function $y_4(\mathbf{z})$ in Table III sums up two PNL projections, so it is approximated by a two-gadaline network. The two plots in Fig. 10(a) respectively show $y_4(\mathbf{z})$ and the obtained approximating function $G_{21,2}$. The two PNL functions as well as their correspondent receptive fields in the obtained $G_{21,2}$ function are shown in Fig. 10(b). It is shown that the proposed learning method succeeds in extracting the two PNL projections of $y_4(\mathbf{z})$.

The input to function $y_5(\mathbf{z})$ contains three ineffective redundant attributes which are not causes to the function output. Fig. 11 shows the obtained $G_{21,2}$ function, in which the two receptive fields

$$\left\{\begin{array}{l}\mathbf{w}_1=[0.974,0.228,0.014,0.015,-0.006]\\ \mathbf{w}_2=[0.313,-0.949,0.008,0.001,0.032]\end{array}\right\}$$

have relatively small weights for the ineffective attributes, z_3, z_4, and z_5. The receptive fields themselves carry with informations on extracting causes of the target function. The mean squared error in approximating the above target functions has ignorable variation among twenty executions, which shows high reliability of the proposed learning method.

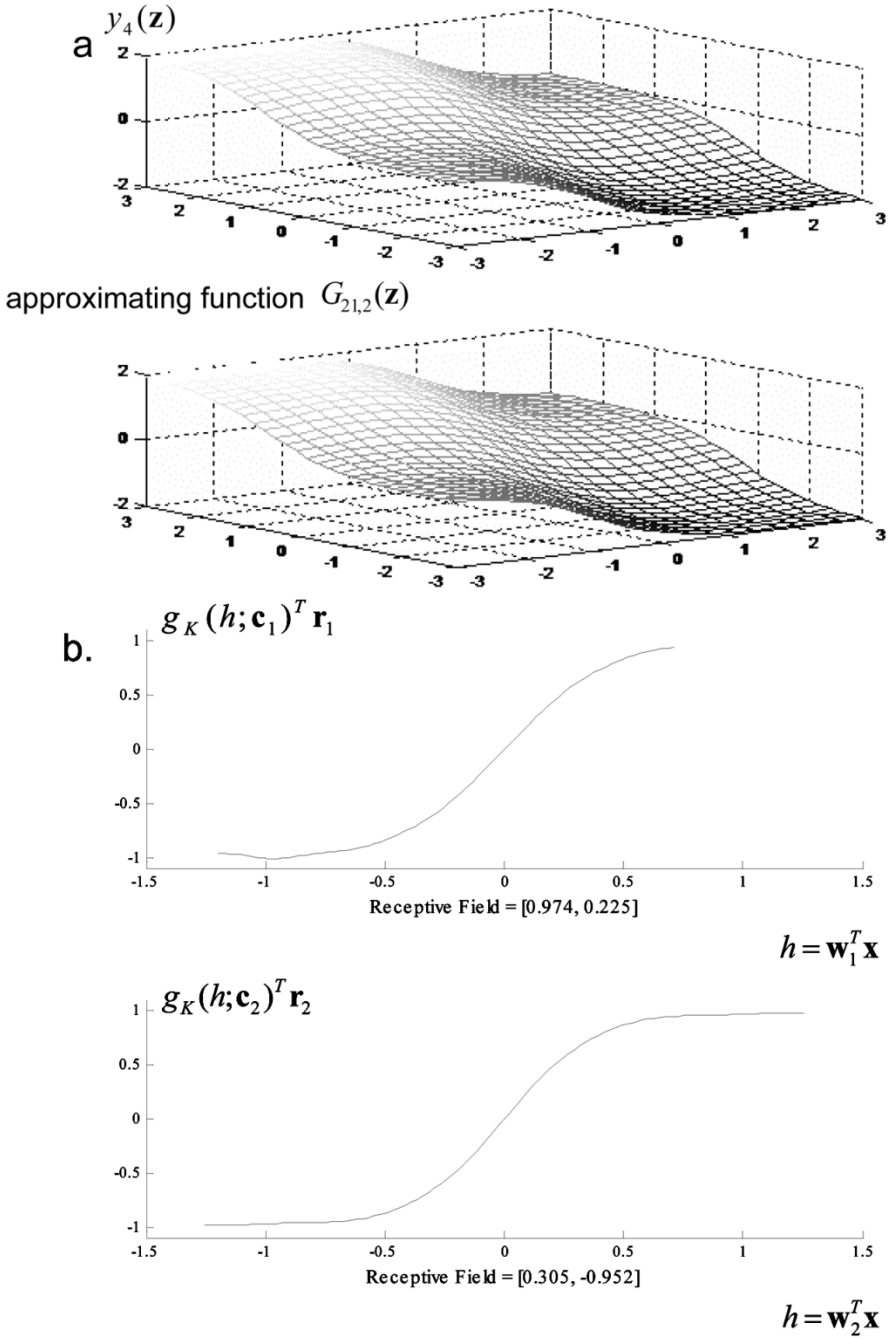

Fig. 10. Approximating function $G_{21,2}$ and the component functions for the summation of two nonlinear projection functions.

The PNL functions in a trained $G_{K,M}$ function are adaptable to the training set. As shown in Fig. 12(b), the two PNL functions embedded within the trained $G_{21,2}$ function for $y_6(\mathbf{z})$ tend to be quadratic ones as in $y_6(\mathbf{z})$.

The Gabor function $y_7(\mathbf{z})$ is thought of as a product of two postnonlinear projections. As argued in Section IV-D, it can be expressed in terms of the following rectified parts:

$$y_7^+(\mathbf{z}) = \mathrm{Rec}(y_7(\mathbf{z}))$$
$$y_7^-(\mathbf{z}) = \mathrm{Rec}(-y_7(\mathbf{z})).$$

The intermediate target functions, $\ln y_7^+(\mathbf{z})$ and $\ln y_7^-(\mathbf{z})$, can be approximated by the trained approximating functions separately subject to two training sets, S^+ and S^-, specified in (30). Let $G_{21,2}^+(\mathbf{z})$ and $G_{21,2}^-(\mathbf{z})$ denote the two approximating functions respectively derived from S^+ and S^-. As shown in (31), the final approximating function is expressed as a combination of $\exp(G_{21,2}^+(\mathbf{z}))$ and $\exp(G_{21,2}^-(\mathbf{z}))$. Fig. 13(a) shows the target function $y_7(\mathbf{z})$ and the final approximating function for $y_7(\mathbf{z})$. The two rectified functions, $y_7^+(\mathbf{z})$ and $y_7^-(\mathbf{z})$, and their approximating functions are respectively shown in the left and right columns of Fig. 13(b). The mean square error from twenty executions listed in the Table IV shows that the Gabor function can is well approximated.

B. Comparisons With Relevant Learning Methods

The proposed learning method is compared with the relevant learning methods for MLP [9] and RBF [14], [16] networks in

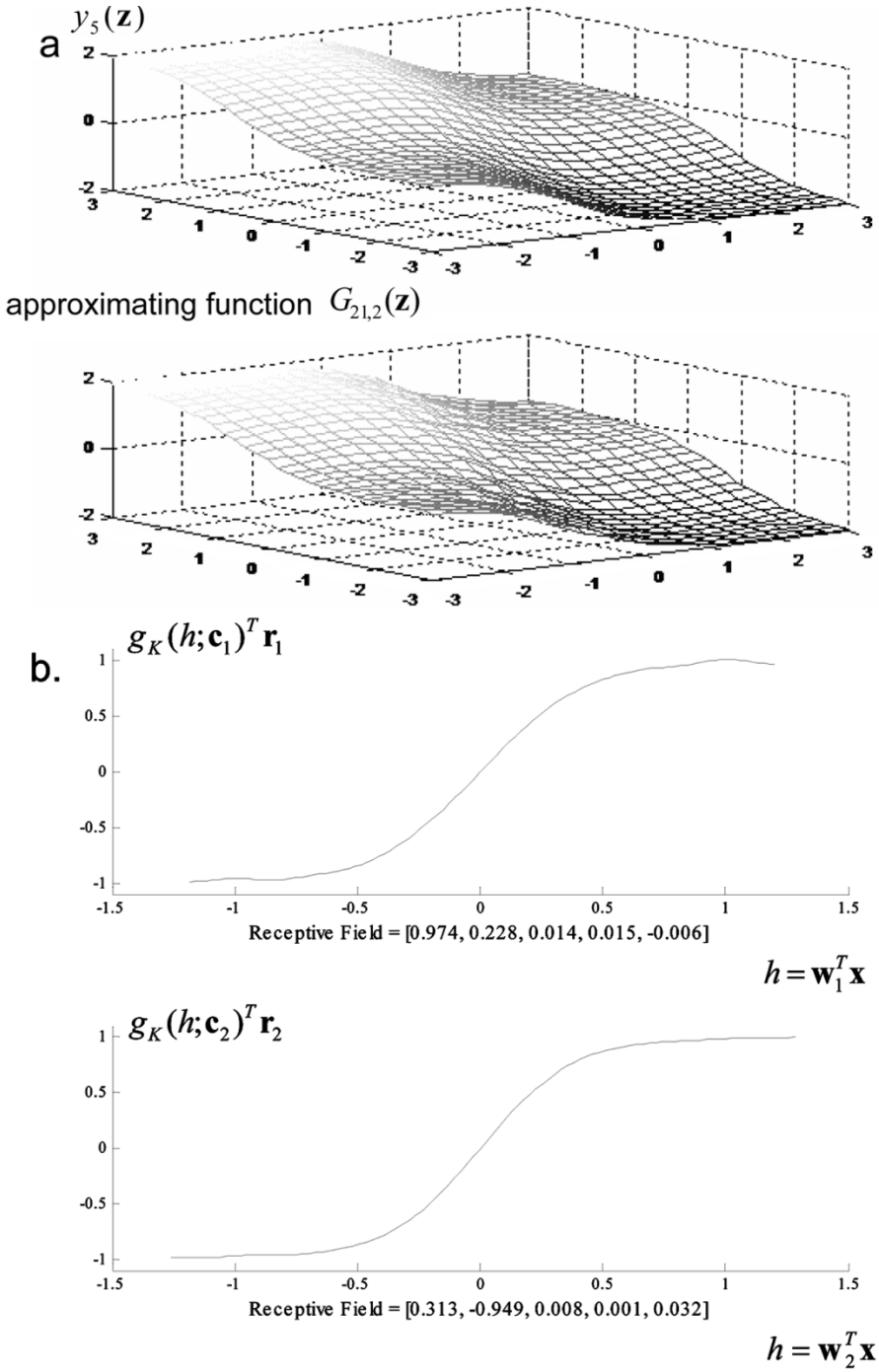

Fig. 11. Approximating function $G_{21,2}$ and the component functions for the summation of two nonlinear projection functions with three dummy variables.

performance and computation efficiency. The used MATLAB program for the RBF learning method was provided by the first author of the paper [16], in which the pseudocode description for the used program appeared on page 310. The MATLAB programs for the MLP leaning methods [4], [13], [12] were provided by the author of the work [13]. The training and testing sets used in the next example respectively contain 1000 and 500 pairs of data.

The sinusoidal grating function shown in Fig. 14 serves as the target function for the first example. Since it coincides with the postsinusoid projection, it is approximated by a gadaline network with $M = 1$. The proposed learning method is expected to extract the most appropriate orientation as well as the sinusoidal nonlinearity from the training set. Table V lists the mean square error and CPU time in seconds summarized from twenty executions of each of the relevant learning methods. The testing error of the MLP learning methods and the RBF learning method using 60 hidden units reduces to 0.0037 and 0.0085, respectively, and the proposed learning method achieves a $G_{60,1}$ function whose testing error reduces to 0.0004. The improvement shows that the most appropriate orientation as well as sinusoidal nonlinearity have been captured and represented by the only receptive field and PNL function in the obtained gadaline network. The averaged CPU time for each of the relevant

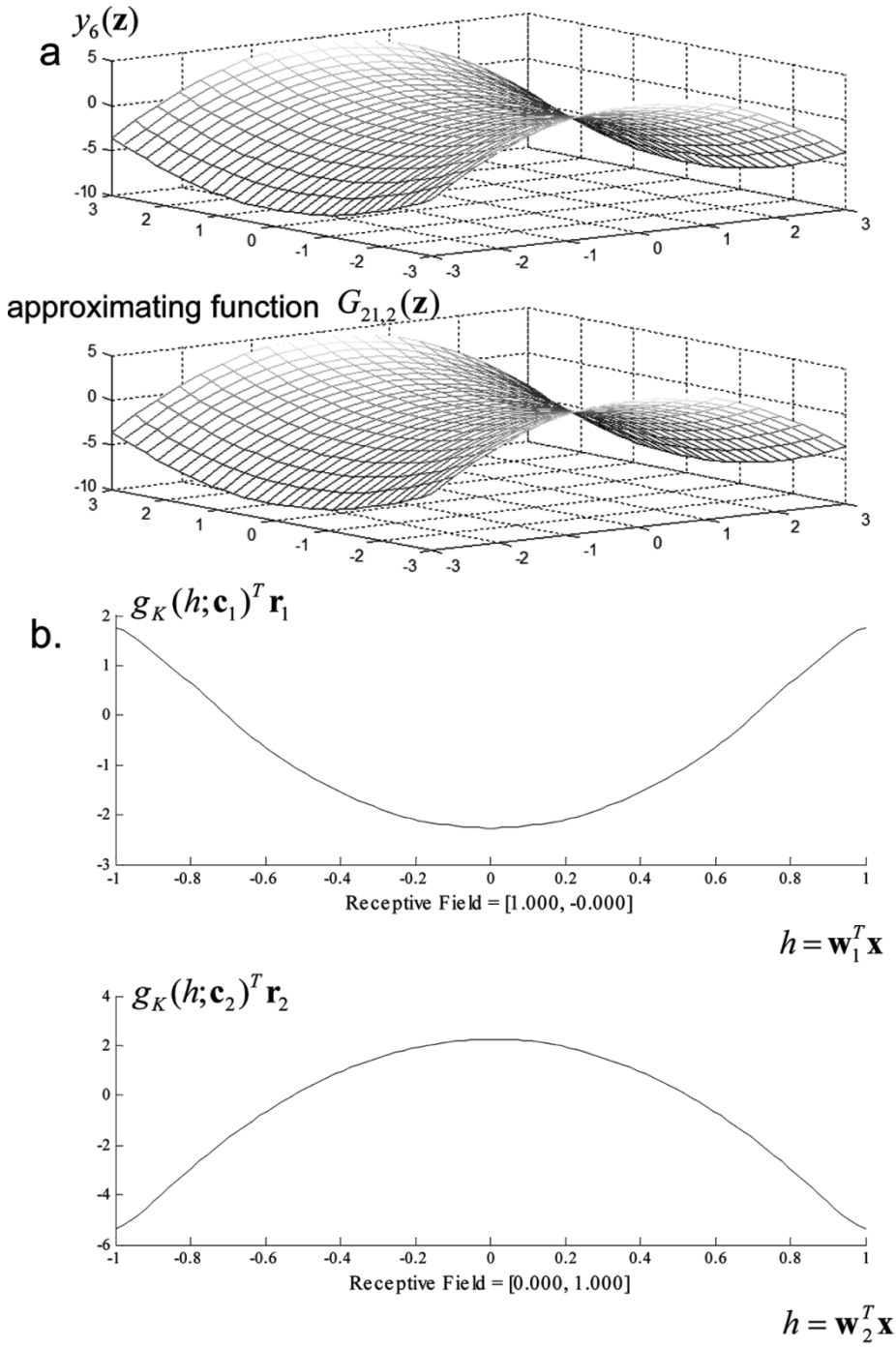

Fig. 12. Approximating function $G_{21,2}$ and the component functions for the quadratic function.

learning methods is listed in the last column of Table V. It is notable that the proposed learning method significantly improves the MLP learning methods in both performance and computational efficiency, and that the RBF learning method has larger variation relative to the proposed learning method in performance. In the first column of Table V, numerics in parentheses refer to the number of hidden units used in an MLP or RBF network.

The next example revisits the target function $y_5(\mathbf{z})$ in Table VI for testing the proposed learning method and the RBF method. The training and testing sets here respectively contain 1000 and 500 pairs of data, where the predictor in each pair contains five elements, each belonging to the interval $[-5, 5]$. The mean square error is listed in Table VI, which shows improvement by the proposed learning method. Since the RBF network performs a mapping based on the Euclidean distance, its learning method cannot brook interference caused by ineffective redundant attributes of the predictor.

The last example tests the relevant learning methods with the following target function:

$$y_8(\mathbf{z}) = \sin(0.4z_1 + 0.1z_2 + 0z_3 + 0z_4 + 0z_5 + 0z_6 + 0z_7) + \sin(0.15z_1 - 0.45z_2 + 0z_3 + 0z_4 + 0z_5 + 0z_6 + 0z_7)$$

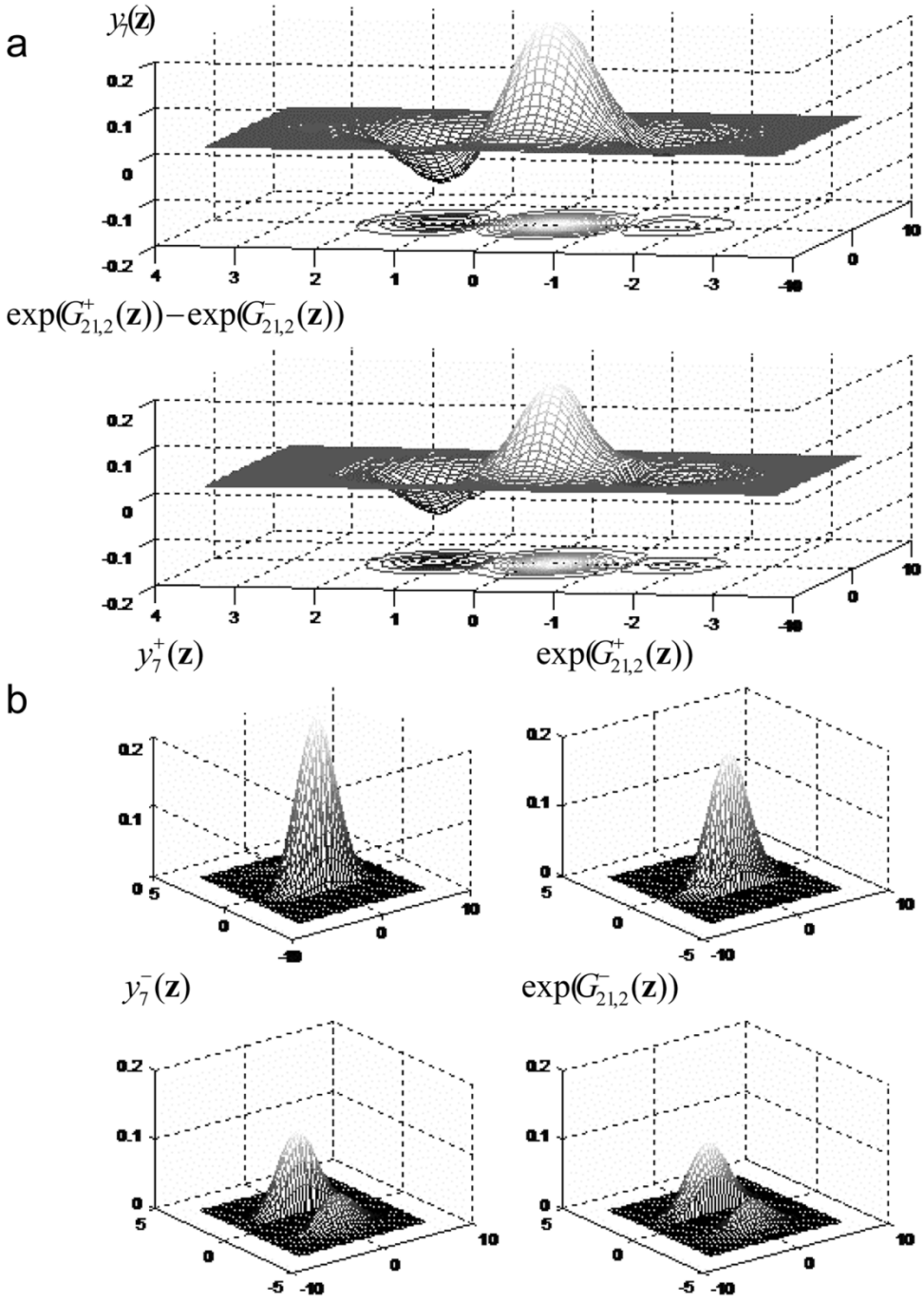

Fig. 13. Approximating function for the Gabor function and its two rectified functions.

a sum of two postsinusoid projections with multiple ineffective redundant attributes. The training and testing sets respectively contain 400 and 200 pairs of data. Table VII lists summary statistics of the design and generalization costs from twenty executions of each of the relevant learning methods. The data shows that the proposed learning method significantly improves the MLP and RBF learning methods in reducing both the training and testing errors.

It is notable that the Levenberg–Marquardt (LM) learning method has resulted in the overfitting situation as reflected by a satisfied training error against an extremely high testing error shown in row MLP-LM(60) of Table VII. The overfitting is primarily caused by the inadequacy of sigmoid-like functions for representing a postsinusoid projection. When the hidden units in an MLP network increased to 60, even though the LM learning method had significantly reduced the training error, it failed to approximate the sum of two postsinusoid projections in terms of 60 post-hypertangent projections as reflected by an intolerable generalization cost. An MLP network with overestimated hidden units failed to represent the function structure of $y_8(\mathbf{z})$. The overfitting situation has been effectively improved by the use of K-state transfer functions, which made the PNL functions embedded within a gadaline network more adaptable and general for representing the sinusoid nonlinearity and others.

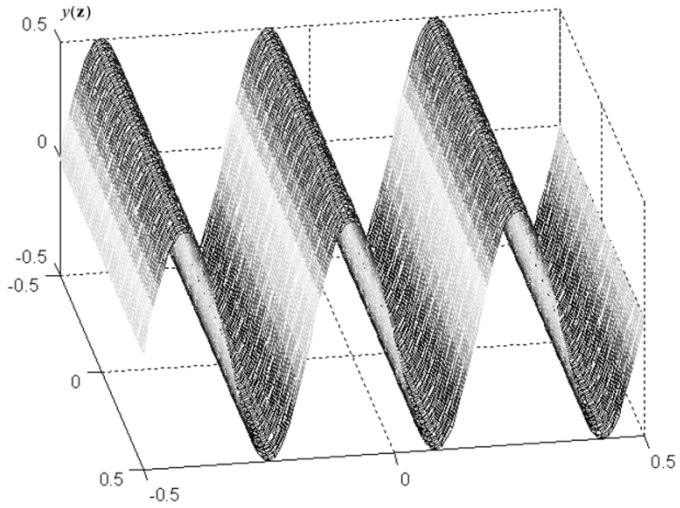

Fig. 14. Sinusoidal grating.

TABLE IV
PERFORMANCE OF APPROXIMATING A VARIETY OF TARGET FUNCTIONS

Target function	Mean±std(training)	Mean±std(testing)	CPU time in seconds
$\cos(z)$	3.06e-2±0	3.24e-2±0	2.3±0.0
$exp(z)$	8.04e-2±0	9.89e-2±0	6.1±0.1
$\ln(z)$	8.52e-2±0	9.07e-2±0	6.8±0.1
$z^3 - z$	9.66e-2±0	9.13e-2±0	5.5±0.1
$\tanh(z)$	8.27e-2±0	9.38e-2±0	7.1±0.1
$y_2(\mathbf{z})$	8.60e-4±0	9.50e-4±0	13.1±0.1
$y_3(\mathbf{z})$	1.90e-4±4.1e-5	1.80e-4±4.7e-5	15.8±0.7
$y_4(\mathbf{z})$	9.50e-4±0	8.70e-4±0	55.7±0.3
$y_5(\mathbf{z})$	1.30e-3±0	1.30e-3±0	48.3±0.2
$y_6(\mathbf{z})$	2.40e-4±1.3e-5	3.00e-4±1.5e-5	16.7±0.5
$y_7(\mathbf{z})$	1.90e-4±1.7e-5	1.70e-4±1.6e-5	21.0±0.4

VI. CONCLUSION

Organizing and learning gadaline networks based on K-state activations have been extensively explored based on stochastic modeling of predictor-oriented target generation using generative components.

The adaline of Widrow is a special case of a gadaline and the network function of adalines is within the function space spanned by gadaline networks. The universal approximation property that has been verified for adaline or perceptron networks is inherent in gadaline networks. In comparison with the adaline or MLP network, the PNL functions embedded within a gadaline network are more flexible and general for function approximation.

The fitness of a generative component to given paired data is characterized by a mixed integer and linear programming for data driven function approximation, which is resolved by a hybrid of the mean field annealing and gradient descent methods. Following the leave-one-out learning strategy, learning multiple generative components is realized by decomposition to subtasks of learning individual generative components.

Numerical simulations that test the proposed learning method with a variety of target functions have shown very encouraging results. The proposed learning method improves the MLP

TABLE V
PERFORMANCE OF THE RELEVANT LEARNING METHODS FOR APPROXIMATING THE SINUSOIDAL GRATING FUNCTION

	Mean±std(training)	Mean±std(testing)	CPU time
MLP-Batch(25)	4.70e-1±0	4.73e-1±0	21.2±0.1
MLP-Batch(60)	4.68e-1±0	4.71e-1±0	33.0±0.1
MLP-Recursive(25)	2.51e-1±0	2.59e-1±0	155.3±0.5
MLP-Recursive(60)	3.60e-3±0	3.70e-3±0	186.0±0.5
RBF(25)	7.98e-2±2.02e-2	8.68e-2±2.15e-2	0.6±0.1
RBF(60)	7.90e-3±7.00e-4	8.50e-3±1.10e-3	1.4±0.1
$G_{25,1}$	6.60e-3±0	6.70e-3±0	10.1±1.7
$G_{60,1}$	5.40e-4±0	6.10e-4±0	16.5±0.5

TABLE VI
PERFORMANCE OF THE TWO METHODS FOR APPROXIMATING FUNCTION $y_5(\mathbf{z})$

	Mean±std(training)	Mean±std(testing)	CPU time
RBF(25)	5.63e-2±4.50e-3	6.37e-2±5.4e-3	0.5±0.1
RBF(60)	3.07e-2±1.20e-3	4.36e-2±1.5e-3	1.6±0.1
$G_{21,2}$	1.30e-3±0	1.30e-3±0	48.3±0.2
$G_{25,2}$	7.80e-4±0	8.80e-4±0	64.3±8.9
$G_{60,2}$	8.10e-5±0	9.70e-5±0	102.9±0.4

TABLE VII
PERFORMANCE OF THE RELEVANT LEARNING METHODS FOR APPROXIMATING $y_8(\mathbf{z})$

Mean Square Error	training set	testing set	Times
MLP-Batch(25)	4.54e-1±0	9.12e-1±0	13.4±0.1
MLP-Batch(60)	2.04e-1±0	8.79e-1±0	20.9±0.1
MLP-Recursive(25)	1.51e-1±0	9.08e-1±0	65.3±0.3
MLP-Recursive(60)	1.17e-1±0	2.29±0	86.6±0.4
MLP-LM(25)	1.42e-1±0	1.00±0	9.6±0.1
MLP-LM(60)	4.00e-6±0	6.71±0	44.3±0.2
RBF(25)	4.72e-1±5.88e-2	6.01e-1±6.98e-2	0.3±0.2
RBF(60)	1.45e-1±1.26e-2	4.09e-1±2.33e-2	0.7±0.1
$G_{25,1}$	2.10e-3±0	2.40e-3±0	15.3±0.9
$G_{60,1}$	5.00e-4±0	1.00e-3±0	40.7±1.2

and RBF learning methods in effectiveness and reliability for approximating the target functions shown in Section V-B, and its efficiency in computation is compatible to that of the MLP learning methods as shown in Tables V and VII. By learning multiple generative components, the obtained approximating function $G_{K,M}$ succeeds in extracting the PNL functions embedded within the target function and is able to detect ineffective redundant attributes within the predictor. The proposed learning method has demonstrated its potential approximation to the target function which takes form of a sum or product of multiple postnonlinear projections.

Like multilayer adalines or perceptrons, the receptive fields within a gadaline network form a basis for a linear transformation. As the number of gadalines in a gadaline network is less than the dimension of predictors, the linear transformation maps from R^d to a reduced space R^M. After the dimensionality–reduction transformation, M PNL functions embedded within a gadaline network nonlinearly translate an image in R^M to an approximating value to the desired target. Since the number of PNL functions in a gadaline network equals to the number of receptive fields, the network parameter number expressed by $Md + M(2K + 1)$ linearly depends on the input dimension d, state number K, and gadaline number M. Therefore, for relatively large d and K, the gadaline network still possesses economic internal representations. The situation is totally different in a Gauss network [19]. Since using a Cartesian product of d sets of regular knots for internal representations of centers of radial basis functions, the parameter number in a Gauss network is proportional to K^d, which increases intolerably for relatively large K and d. This is the reason why the gadaline network is more suitable for function approximation with high dimensional predictors in comparisons with the RBF network and Gauss network.

It has been shown that the proposed learning method is effective for detecting ineffective redundant attributes of the predictor via seeking for essential receptive fields as well as PNL functions. That indicates applicability of the proposed learning method to the task of supervised dimensionality–reduction. The predictor for $y_8(\mathbf{z})$ contains eight attributes, however, only two of them cause the function output. Numerical simulations show well approximation to $y_8(\mathbf{z})$ by learning a gadaline network with $M = 2$, of which internal representations of receptive fields and PNL functions structure well such a target function, avoiding the interference caused by ineffective redundant attributes. The proposed learning method has been shown to significantly improve the MLP and RBF learning methods in approximating $y_8(\mathbf{z})$. Supervised dimensionality–reduction by the proposed learning method will be further explored in the near future with applications to classification, signal prediction, and system identification for medical signals [1], [3], [8], [13], [17], [23].

REFERENCES

[1] C. M. Bishop, *Neural Networks for Pattern Recognition*. London, U.K.: Oxford Univ. Press, 1995.

[2] G. Bugmann, "Normalized Gaussian radial basis function networks," *Neurocomput.*, vol. 20, no. 1–3, pp. 97–110, Aug. 1998.

[3] D. Erdogmus and J. C. Principe, "Convergence properties and data efficiency of the minimum error entropy criterion in adaline training," *IEEE Trans. Signal Process.*, vol. 51, no. 7, pp. 1966–1978, Jul. 2003.

[4] R. Fletcher, *Practical Methods of Optimization*. New York: Wiley, 1987.

[5] S. Ferrari and R. F. Stengel, "Smooth function approximation using neural networks," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 24–38, Jan. 2005.

[6] G. Pajares and J. M. de la Cruz, "Local stereovision matching through the ADALINE neural network," *Pattern Recognit. Lett.*, vol. 22, no. 14, pp. 1457–1473, Dec. 2001.

[7] P. Dayan and L. F. Abbott, "Theoretical neuroscience: Computational and mathematical modeling of neural systems," *J. Roy Stat. Soc. B. Met.*, vol. 58, no. 1, pp. 155–176, 2001. analysis by Gaussian mixtures.

[8] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ: Prentice Hall, 1999.

[9] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. Reading, MA: Addison-Wesley, 1991.

[10] T. Hayakawa, W. M. Haddad, N. Hovakimyan, and V. Chellaboina, "Neural network adaptive control for nonlinear nonnegative dynamical systems," *IEEE Trans. Neural Netw.*, vol. 16, no. 2, pp. 399–413, Mar. 2005.

[11] M. M. S. Lee, S. S. Keerthi, C. J. Ong, and D. DeCoste, "An efficient method for computing leave-one-out error in support vector machines with Gaussian kernels," *IEEE Trans. Neural Netw.*, vol. 15, no. 3, pp. 750–757, May 2004.

[12] L. Ljung, *System Identification—Theory for the User*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[13] M. NØrgaard, O. Ravn, N. K. Poulsen, and L. K. Hansen, *Neural Networks for Modeling and Control of Dynamic Systems*. London, U.K.: Springer-Verlag, 2000.

[14] K.-R. Muller, A. J. Smola, G. Ratsch, B. Scholkopf, J. Kohlmorgen, and V. Vapnik, "Using support vector machines for time series prediction, in advances in kernel methods—Support vector learning," *Proc. NIPS'97 Support Vector Workshop*, pp. 243–254, 1997.

[15] C. Peterson and B. Södergerbg, "A new method for mapping optimization problems onto neural network," *Int. J. Neural Syst.*, vol. 1, no. 3, 1989.

[16] G. Ratsch, T. Onoda, and K. R. Muller, "Soft margins for AdaBoost," *Mach. Learn.*, vol. 42, no. 3, pp. 287–320, Mar. 2001.

[17] B. Mitchinson and R. F. Harrison, "Digital communications channel equalization using the kernel adaline," *IEEE Trans. Commun.*, vol. 50, no. 4, pp. 571–576, Apr. 2002.

[18] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Washington, DC: Spartan, 1962.

[19] R. M. Sanner and J. J. E. Slotine, "Gaussian Networks for direct adaptive-control," *IEEE Trans. Neural Netw.*, vol. 3, no. 6, pp. 837–863, Nov. 1992.

[20] Z. Uykan, "Clustering-based algorithms for single-hidden-layer sigmoid perceptron," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 708–715, May 2003.

[21] P. J. Werbos, "Backpropagation: Past and the future," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 1, Jul. 1988, pp. 343–353.

[22] B. Widrow, *Generalization and Information Storage in Networks of Adaline Neuron in Self-Organizing Systems*, M. Yovitz, G. Jacobi, and G. Goldstein, Eds. Washington, DC: Spartan, 1962, pp. 435–461.

[23] B. Widrow, "30 years of adaptive neural networks: Perceptron, madaline, and backpropagation," *Proc. IEEE*, vol. 78, no. 9, Sep. 1990.

[24] J. M. Wu, "Natural discriminant analysis using interactive potts models," *Neural Comput.*, vol. 14, no. 3, pp. 689–713, Mar. 2002.

Jiann-Ming Wu was born in Taiwan, on November 22, 1966. He received the B.S. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 1988, the M.S. and Ph.D. degrees in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 1990 and 1994, respectively.

In 1996, he joined the Faculty as an Assistant Professor at the Department of Applied Mathematics in National Dong Hwa University, Hualien, Taiwan, where he is currently an Associate Professor. His current research interests include neural networks and signal processing.

Zheng-Han Lin was born in Taiwan, R.O.C., on August 31, 1973. He received the B.S. degree in applied mathematics from Providence University, Taichung, Taiwan, in 1996, the M.S. and Ph.D. degrees in applied mathematics from National Dong Hwa University, Hualien, Taiwan, in 1998 and in 2005, respectively.

In 2005, he joined the faculty of the MingDao University, ChangHua, Taiwan, as an Assistant Professor at the Department of Computer Science and Information Engineering. His present research interests include signal and image processing, and design and analysis of generative model for constrained optimization and their applications.

Pei-Hsun Hsu was born in Taiwan in 1977. She received the B.S. degree in mathematics from National Central University, Taoyuan, Taiwan, in 1996, and the M.S. degree in applied mathematics from National Dong Hwa University, Hualien, Taiwan, in 2003. She is currently working toward the Ph.D. degree in computer science and information engineering, National Taiwan University, Taipei, Taiwan.