Progressive Learning Machine: A New Approach for General Hybrid System Approximation

Yimin Yang, Member, IEEE, Yaonan Wang, Q. M. Jonathan Wu, Senior Member, IEEE, Xiaofeng Lin, and Min Liu

Abstract—As the most important property of neural networks (NNs), the universal approximation capability of NNs is widely used in many applications. However, this property is generally proven for continuous systems. Most industrial systems are hybrid systems (e.g., piecewise continuous), which is a significant limitation for real applications. Recently, many identification methods have been proposed for hybrid system approximation; however, these methods only operate in linear hybrid systems. In this paper, the progressive learning machine—a new learning algorithm based on multi-NNs—is proposed for general hybrid nonlinear/linear system approximation. This algorithm classifies hybrid systems into several continuous systems and can approximate any hybrid system with zero output error. The performance of the proposed learning method is demonstrated via numerical examples and with experimental data from real applications.

Index Terms—Bidirectional extreme learning machine (B-ELM), cluster growing algorithm, hybrid system approximation, neural networks (NNs).

I. INTRODUCTION

THE widespread popularity of neural networks (NNs) in many fields is mainly due to their ability to approximate complex nonlinear mappings from input samples. The NNs can provide models for a large class of natural and artificial phenomena that are difficult to handle using classical parametric techniques. Recently, Huang et al. [1], Yang et al. [2], Kasun et al. [3], and Luo et al. [4] have proven that single-hidden-layer feedforward networks (SLFNs) with random hidden nodes can approximate any continuous nonlinear/linear system. However, most real industrial systems are hybrid systems (piecewise nonlinear systems) for which there are few approximation results in [5]. Hybrid systems are models of processes governed by differential or difference equations that exhibit both continuous and discontinuous dynamic behaviors. Continuous dynamics are described by variables taking values from a continuous set, while discontinuous dynamics are described by variables taking values from a discrete, typically finite, set [6]. In some instances, discontinuous behavior arises genuinely from switching or interaction between separate physical systems. More often, hybrid systems are a mere modeling abstraction: the physical process is intrinsically continuous, but its behavior is more easily described by certain discrete events, transitions, phases, modes, or states [7].

As two special classes of hybrid systems, approximation of piecewise affine (PWA) models and switched linear models has received much attention from researchers, because they can be used to describe hybrid phenomena that are frequent in practical situations, e.g., when the system dynamics change due to physical limits, dead-zones, switches, and thresholds. The PWA models are systems in which the state-input domain is partitioned into a finite number of polyhedral clusters, with each submodel in the different clusters being linear or affine [8]. Switched linear models are defined as collections of linear models, connected by switches that are indexed by a discrete-valued additional variable (discrete state) [6]. If the submodels in each cluster have an autoregressive exogenous (ARX) input–output relationship, then the system is called a piecewise affine ARX (PW-ARX) or switched ARX (SARX) system. Another distinct advantage of the approximation of hybrid linear models is that they form an attractive model structure that can approximate nonlinear dynamical systems by switching among various linear or affine models [9, 10]. Consequently, they are useful for the controller design of nonlinear systems, as linear controllers for affine/linear subsystems can be first designed according to any of the well-known linear control synthesis methods, and then based on the operating cluster of the nonlinear system, the controller can switch from one to another.

Hybrid system approximation is a very challenging problem. Most existing methods are designed for the piecewise linear model or the switched linear model. Existing methods for hybrid system approximation include the following.

1) In the case of switched linear models, the algebraic-geometric method [7], [11], [12] and the product-of-errors-based method [13] are used to estimate the number of discrete state and model parameters. References [14] and [15] suggest sparse representation as a possible alternative for solving the approximation problem of switched linear systems.
2) In the case of PWA/linear models, the clustering-based strategies [16]–[18], the kernels method [19], the bounded-error approach [20], the Bayesian learning-based procedure [21], and the mixed integer programming approach [22] are applied to the identification problem of PWA models. The bounded-error approach [20] uses a greedy strategy to partition a set of unfeasible inequalities into a minimum number of feasible subsystems, and then iterates between assigning regressors to models and computing the model parameters. In [18], the PWARX system is considered and clustering, linear least squares, fuzzy clustering, and pattern recognition techniques are exploited to identify both the linear submodel and the polyhedral partition of the regressor set. In [22], the approximation problem is formulated for two subclasses of PWA models, namely, hinging hyperplane ARX (HHARX) and PWARX models, and solved via mixed-integer linear or quadratic programs.

Unfortunately, all the aforementioned hybrid system identification algorithms only operate in linear subsystems or ARX subsystems, i.e., the submodel dynamics is a linear or very simple nonlinear function (nonlinear ARX). This could be a significant limitation for practical applications.

This paper proposes solutions for the general hybrid system approximation problem: 1) approximation of nonlinear hybrid systems switching between complex continuous unknown nonlinear dynamics and 2) approximation of general piecewise nonlinear systems in which the submodel dynamics are any nonlinear function. Our solution is called progressive learning nonlinear systems in which the submodel dynamics are any linear dynamics and 2) approximation of general piecewise nonlinear systems, switched linear systems, SARX systems, and general switched nonlinear systems.

2) The proposed algorithm is developed from SLFNs. This algorithm differs from other NN learning methods that only have universal approximation capability for continuous systems. This paper manages to extend NNs to discontinuous, piecewise, and nonlinear/linear systems: it is shown that the proposed method can approximate any hybrid system.

II. PRELIMINARIES AND PROBLEM STATEMENT

A. Notations

The sets \( \{\Omega_i\} \) form a complete partition of the set \( \{\Omega\} \) if \( \bigcup_{i=1}^{n} \Omega_i = \Omega \) and \( \Omega_i \cap \Omega_j = \emptyset, \forall i \neq j \). The \( m \)-ary Cartesian product of a partition \( \Omega \) is denoted by \( \Omega^m \). The difference of two sets \( \Omega_i \) and \( \Omega_j \) is denoted by \( \Omega_i/\Omega_j \). The sets of real, integer, and positive integer numbers are denoted by \( \mathbb{R} \), \( \mathbb{Z} \), and \( \mathbb{Z}^+ \), respectively. The \( #A \) defines the number of samples in a finite cluster \( A \). For \( M \) training samples \( \{(y_i, x_i)\}_{i=1}^{M} \), \( x \) represents a regression input vector and \( y \) represents output data. For all \( j \in \mathbb{Z}^+ \), the notation \( \{(a_j, b_j, L_j)\}_{i=1}^{L_j} \) represents parameters partition of \( j \)th SLFNs NN\(_j\), \( L_j \) represents the hidden-node-numbers of NN\(_j\), \( a_j \) is the weight vector connecting the input layer to the \( l \)th hidden node, \( b_j \) is the bias of the \( l \)th hidden node, and \( \beta_j \) is the weight connecting the \( j \)th hidden node to the output node. The notation \( \hat{y}(j, L_j, x) \) represents the output of \( j \)th network with \( L_j \) hidden nodes, i.e., \( \hat{y}(j, L_j, x) = \sum_{l=1}^{L_j} b_{jl} H(a_{jl}, b_{jl}, x) \). The \( H \) is called the hidden layer output matrix of the SLFNs; the \( l \)th column of \( H(H) \) is the \( l \)th hidden node output with respect to inputs. \( s_1, s_2, s_3, (s_1, s_2, s_3) \in \mathbb{Z}^+ \) represent the loop-count-numbers (Section III-C).

B. Problem Statement and Definition

Given a discrete-time nonlinear dynamical system with inputs \( u \in \mathbb{R}^n \), outputs \( y \in \mathbb{R} \), and possibly discontinuous dynamics, let \( u(k-1) \) and \( y(k-1) \) be past inputs and outputs, respectively, generated by the system up to time \( k-1 \). Formally, a hybrid model is defined as

\[
\begin{align*}
\begin{cases}
x(k+1) &= f_1(x(k)) + e_1, y(k+1) = g_1(x(k)) + \mu_1 \text{ if } [x(k), y(k)] \in \Omega_1 \\
\vdotso & \\
x(k+1) &= f_{c'}(x(k)) + e_{c'}, y(k+1) = g_{c'}(x(k)) + \mu_{c'} \text{ if } [x(k), y(k)] \in \Omega_{c'},
\end{cases}
\end{align*}
\]

\( x(k) = [y(k-1), \ldots, y(k-n_a), u^T(k-1), \ldots, u^T(k-n_b), w^T(k)] \)

(1)

where \( x(k) \) is the regression vector with fixed structure depending only on the past \( n_a+n_b \) outputs \( y(k), \ldots, y(k-n_a), \) past \( n_b \) inputs \( u(k), \ldots, u(k-n_b) \), and the state \( w(k) \). The \( c' \) is the number of submodels (or discrete modes), \( x(k+1) \in \mathbb{R}^m, f(\cdot)_i \) and \( g(\cdot)_i, i = 1, \ldots, c' \) are unknown, nonlinear, and continuous functions. The \( e_1, \mu_1, i = 1, \ldots, c' \) are noise.

Remark 1: Different from identification methods that only works for linear piecewise or switched models, such as PWARX, the switched linear model, SARX, and HHARX, this paper studies a more general hybrid system approximation problem in which the number of submodels and system orders \( (n_a, n_b) \) are unknown, and the unknown submodel function \( f(\cdot), g(\cdot) \) in (1) is complex nonlinear. Such a problem is very complex, but it is very meaningful for real applications.

Definition 1: Given a finite data set \( \mathcal{X} \) that is sampled from a hybrid model in (1), \( Q_l, i = 1, \ldots, c \) are called a maximum submodel block of \( \mathcal{X} \) and \( c \) is the number of maximum submodel blocks of \( \mathcal{X} \) according to the following.

1) \( Q_l, i = 1, \ldots, c \) form a complete partition of \( \mathcal{X} \).
2) There are no other continuous submodel partitions \( Q'_l \subseteq \mathcal{X}, i = 1, \ldots, c' \) and \( c' < c \). This implies \( c \) the smallest number of continuous submodels in \( \mathcal{X} \).
3) \( Q_l \) is sampled from a unique continuous unknown dynamic subsystem in (1).

In order for this identification problem to be well posed, \( c \) number of NNs, i.e., \( \{(a_j, b_j, L_j)\}_{i=1}^{L_j}, j = 1, \ldots, c \) must be uniquely defined, and the number \( c \) should be the smallest. This requires that dynamic submodels in a hybrid system differ from one other so that we can identify the discrete state.
Problem 1: Given a finite set of data points \( \mathcal{X} = \{y_k, x_k\} \), \( k = 1, \ldots, M \) sampled from a hybrid model (1), and any small positive value \( \epsilon > 0 \), estimate a number of maximum submodel blocks \( c \), maximum submodel blocks of \( \mathcal{X} \) \( Q_i = \{(x_{i_k}, y_{i_k})\}_{k=1}^{M_i}, i = 1, \ldots, c \), and obtain \( c \) unique NN parameter sets \( \text{NN}_j = \{(a_{i_j}, b_{i_j}, \beta_{i_j})\}_{i=1}^{L_j}, j = 1, \ldots, c \) such that the corresponding hybrid model (1) satisfies

\[
|\hat{y}(j, L, x_k) - y_{i_k}| \leq \epsilon, k = 1, \ldots, M_j, \quad j = 1, \ldots, c. \tag{3}
\]

Remark 2: The same hybrid model can generally be expressed in different ways. For example in \([23]\), the same piecewise linear affine system can be expressed by

\[
w(k + 1) = 0.8 \left[ \begin{array}{c}
\cos(a(k)) \\
\sin(a(k)) \\
\cos(a(k)) 
\end{array} \right] w^T(k) + \left[ \begin{array}{c}
0 \\
1 
\end{array} \right] u(k) \\
y(k) = [1, 0]^T w^T(k) \tag{4}
\]

\[a(k) = \begin{cases}
\frac{\pi}{2}, & \text{if } w_1(k) \geq 0 \text{ (mode 1)} \\
-\frac{\pi}{2}, & \text{if otherwise (mode 2)}
\end{cases}
\]

\[y(k) = \begin{cases}
0.8y(k - 1) - 0.64y(k - 2) - 0.4\sqrt{3}u(k - 2) & \text{if } y(k - 1) \geq 0, y(k - 2) \geq 0 \quad \text{model 1} \\
0.64y(k - 2) + 0.4\sqrt{3}u(k - 2) & \text{if } y(k - 1) < 0, y(k - 2) \geq 0 \quad \text{model 2} \\
0.8y(k - 1) - 0.64y(k - 2) + 0.4\sqrt{3}u(k - 2) & \text{if } y(k - 1) < 0, y(k - 2) < 0 \quad \text{model 3} \\
0.64y(k - 2) - 0.4\sqrt{3}u(k - 2) & \text{if } y(k - 1) \geq 0, y(k - 2) < 0 \quad \text{model 4}. 
\end{cases} \tag{5}
\]

If we use (5) to express this hybrid system, according to (1) we can easily get \( c' = 4 \) because there are four submodels in (5). However, if we use (4) to express the same hybrid system, we will get \( c' = 2 \) because there are only two submodes in (5). According to Definition 1.2, the number of maximum submodel blocks \( c \) is the smallest continuous submodel-number in the hybrid system. Thus, the number of maximum submodel blocks is 2 not 4 regardless of which equation is used to generate the training points \( \mathcal{X} \). By solving Problem 1, we can get the unique maximum submodel blocks of \( \mathcal{X} \): \( Q_1, Q_2 \), and \( c = 2 \) (it can also be indicated by experiment in Section VI). The aim of this paper is to find the smallest number of NNs to approach each corresponding maximum submodel block in \( \mathcal{X} \).

Remark 3: The focus here is on providing several particular NNs to approach \( c \) continuous dynamic subsystems (submodels) in hybrid systems and to approximate a hybrid nonlinear system model to find the smallest number of NN parameter sets satisfying (3).

The basic idea of the PLM is to find NNs that make the inequalities in (3) true for as many \( M_j (j = 1, 2, \ldots, c) \) as possible, and then to remove those satisfied data points and repeat the process for those that remain until all data points have been accounted for.

The proposed method is composed of two parts: 1) cluster growing for classifying training data and 2) cluster estimation for testing data. In Section III, we propose a cluster estimation algorithm for training data (the cluster growing algorithm). In Section IV, we propose a cluster estimation algorithm for testing data. The structure of proposed PLM is shown in Fig. 1.

### III. Cluster Growing and Optimal Offspring Cluster Selection

To solve Problem 1, we must first focus our attention on estimating a number of maximum submodels partitions, classifying the data points into several clusters, and obtaining these continuous submodels. The cluster-growing algorithm is shown in Fig. 2, and there are four steps in this proposed method (see Steps 1–4).

**Definition 2:** Given a training cluster \( \phi = \{(y_k, x_k)\}, k = 1, \ldots, M \). For all \( j \in \mathbb{Z}^+ \), the notation \( \text{NN}_\phi^j \) represents parameter set of \( j \)th SLFNs \( \text{NN}_\phi^j = \{(a_{i_j}, b_{i_j}, \beta_{i_j})\}_{i=1}^{L_j} \), which have already been trained by cluster \( \phi \) such that

\[
\text{NN}_\phi^j = \{(y_k, x_k) \mid |y_k - \hat{y}(j, L, x_k)| = 0 \} \quad \text{for } k = 1, \ldots, M. \tag{6}
\]

**Definition 3:** Given a finite set of data points \( \mathcal{X} = \{y_k, x_k\}, k = 1, \ldots, M \) that is sampled from a hybrid model in (1), we called the cluster \( \phi^i \) as a parent cluster, and called \( \phi^{i+1} \) as the offspring cluster of the cluster \( \phi^i \) according to the following.

1) \( \phi^i, \phi^{i+1} \subset \mathcal{X} \).
2) Given \( m \) SLFNs \( \text{NN}_\phi^i = \{(a_{i_j}, b_{i_j}, \beta_{i_j})\}_{i=1}^{L_j} \), \( i = 1, \ldots, m \) which have already been trained by cluster \( \phi^i \), the cluster \( \phi^{i+1} \) is obtained by

\[
E_i = \{(y_k, x_k) \mid |y_k - \hat{y}(i, L, x_k)| = 0, (y_k, x_k) \in \mathcal{X} \} \quad k = 1, \ldots, M, \quad j = 1, \ldots, m
\]

\[
\phi^{i+1} = \{\phi^{i+1} \mid \phi^{i+1} \in \{E_i\}_{i=1}^m \}
\]

**Definition 4:** Given a cluster \( \phi^i \) as a parent cluster and repeated (7) \( s \) times to generate \( s \) offspring clusters \( \phi^{i+1}, \ldots, \phi^{i+s} \). We called \( \phi^{i+s} \) as a \( s \)th-generation offspring cluster of \( \phi^i \).

**Step 1:** Given a finite training points \( \mathcal{X} \), which is sampled from a hybrid model (1), and in the training points \( \mathcal{X} \), there are \( c \) maximum submodel blocks: \( Q_i = \{(y_{i_k}, x_{i_k})\}_{k=1}^{M_i} \),...
The optimal offspring cluster $Y$ belongs to the same maximum submodel block. More importantly, the number of points in these $s_1$ offspring clusters will monotonically increase: $(#\phi_0^1 < #\phi_1^1 < \cdots < #\phi_{s_1}^1)$. On the other hand, according to Theorem 2, if points of data in a parent cluster $\phi_0^1$ or points of data in the offspring cluster $\phi_1^1$ of the parent cluster $\phi_0^1$ come from different continuous systems, the number of points in these offspring clusters will be fixed and not extend any more $#\phi_1^1 \ll M$. Thus, according to Steps 3 and 4, the optimal offspring cluster $Y$ belongs to the same maximum submodel block and the upper bound of $\#Y$ equals to $M$.

The above four steps constitute a cluster growing algorithm, as shown in Fig. 2. This cluster growing algorithm ensures the optimal offspring cluster $Y$ belongs to the same maximum submodel block. To prove that the optimal offspring cluster $Y$ belongs to the same maximum submodel block, we discuss two situations in the following sections. In Section III-A, we will prove that if a parent cluster $\phi_i^j$ belongs to the same maximum submodel block, its offspring cluster $\phi_{i}^{j+1}$ also belongs to the same maximum submodel block. More importantly, these offspring clusters will grow: $\phi_i^j \subseteq \phi_{i}^{j+1} \subseteq \cdots \subseteq \phi_{i}^{j+s_1} = Y = Q_i \subset \mathcal{X}$. These clusters will be fixed and not extend: $#\phi_i^j \approx #\phi_{i}^{j+1} \approx \cdots \approx #\phi_{i}^{j+s_1} \ll #Q_i$. In this paper, we prove that if a parent cluster and its offspring clusters belong to a hybrid system (discontinuous system), these clusters will be fixed and not extend: $#\phi_i^j \approx #\phi_{i}^{j+1} \approx \cdots \approx #\phi_{i}^{j+s_1} \ll #Q_i$.

---

1Here, we do not prove a parent cluster $\phi_0^1$, which belongs to a discontinuous system (hybrid system), cannot produce offspring clusters belonging to the same maximum submodel block. It is unnecessary to consider this situation: if an offspring cluster $\phi_1^j$, which is produced by a discontinuous parent cluster $\phi_0^1$ ($\phi_0^1 \subset \mathcal{X}, \phi_0^1 \not\subset Q_i, i = 1, \ldots, c$), belongs to the same maximum cluster, we can get this offspring cluster $\phi_1^j$ can be seen as a new parent cluster. Finally, the new parent cluster $\phi_1^j$ can produce its offspring clusters, which also belongs to the same maximum submodel block.
A. Parent Cluster \( \phi_i^j \) Belongs to the Same Maximum Submodel Block

If a parent cluster \( \phi_i^j, 1 \leq i \leq n \) belongs to a maximum submodel block \( \mathcal{Q}_i \), we can get the following result: its \( s \)th-generation offspring cluster \( \phi_i^{j+1}, \) \( j < s < 1 \), satisfies: \( \#\phi_i^j < \#\phi_i^{j+1} < \ldots < \#\phi_i^1 = \#\mathcal{Q}_i, \phi_i^1 = \mathcal{Q}_i \), if \( s1 \to +\infty \).

To prove this result, we first prove Lemmas 1-3 and then prove Theorem 1.

**Lemma 1:** Given \( c \) maximum submodel blocks: \( \{(y_i^j, x_i^j)\}_{i=1}^N \subseteq \mathcal{X} \times \mathcal{R}^m, \{(y_i^j, x_i^j)\}_{i=1}^N = \mathcal{Q}_i, i = 1, \ldots, c, \) and \( N, m \in \mathbb{Z}^+ \). Assuming \( q, q \in \mathbb{Z}^+ \), samples are selected randomly from \( \mathcal{X} \), repeated \( n \) times, \( n \) \( \to +\infty \), there exists one group of \( q \) samples in these \( n \) groups such that all the points in this group belong to the same maximum submodel block.

**Proof:** According to the condition, we get \( \#\mathcal{X} = c \times N \). The probability of \( q \) samples belonging to the same maximum submodel block is \( P = (C_1)^q/C_{cn}^q < 1 \) if these \( q \) samples are selected at random from \( \mathcal{X} \) once. Therefore, repeated \( n \) times, at least once the probability of \( q \) samples belonging to the same maximum submodel block is \( 1 - (1 - P)^n \). Because \( (1 - P) < 1 \), the probability \( (1 - P)^n \to 0 \) and \( 1 - (1 - P)^n \to 1 \) if \( n \to +\infty \).

**Lemma 2:** Given \( c \) training partitions: \( \{(y_i^j, x_i^j)\}_{i=1}^N \subseteq \mathcal{R} \times \mathcal{R}^m, i = 1, \ldots, c, \) and \( N, m \in \mathbb{Z}^+ \). Each training data set belongs to a maximum submodel block \( \mathcal{Q}_i \), i.e., \( \{(y_i^j, x_i^j)\}_{i=1}^N \subseteq \mathcal{Q}_i, i = 1, \ldots, c, \) \( j \)th NN (\( \mathbb{N}_j^y \) = \( \{a_i^j, b_i^j, \beta_i^j\}_{i=1}^L, L \gg N \)) with \( L \) hidden-node numbers that has already been trained by training set \( \mathbb{Q}_j^y \) = \( \{y_i^j, x_i^j\}_{i=1}^N \subseteq \mathbb{Q}_j^y \subseteq \mathbb{Q}_i \), where \( \frac{1}{\phi_i^j} \leq \frac{1}{\phi_i^{j+1}} \). The set \( \phi_i^j \) both come from the same continuous function. In theory, NNs can approximate any continuous function. Thus, if \( m \to +\infty \), there exists an NN (\( \mathbb{N}_j^y \)) in these \( m \) NNs, such that \( \{y_i, x_i\} | y_i = \hat{y}_j(g, L, x_i) = 0, (y_i, x_i) \in \phi_i^j/\phi_i^{j+1} \). (13)

Equation (13) implies \( \#\mathcal{Q} \to +\infty, \#\mathcal{Q} \neq N \).

If we combine the condition Lemma 2 with the condition of Lemma 3, then we can obtain more generally result that is very important for solving Problem 1.

**Theorem 1:** Given \( c \) maximum submodel blocks of training set \( \mathcal{X} \), \( \mathbb{Q}_j^y = \{y_i^j, x_i^j\}_{i=1}^N \subseteq \mathbb{Q}_j^y \subseteq \mathbb{Q}_i \), \( g \) \( \in \mathbb{Z}^+ \). Given training set: \( \phi_i^j = \{y_i^j, x_i^j\}_{i=1}^N \subseteq \mathcal{R} \times \mathcal{R}^m, i = 1, \ldots, c, \) and \( g \) \( \in \mathbb{Z}^+ \). If \( m \) SLFNs (\( \mathbb{N}_j^y \) = \( \{a_i^j, b_i^j, \beta_i^j\}_{i=1}^L, i = 1, \ldots, m \)) that has already been trained by \( \mathcal{Q}_j^y \): \( \sqrt{\sum_{i=1}^L b_i^j H(a_i^j, b_i^j, x_i^j)} = 0 \) for all \( k = 1, \ldots, N, i = 1, \ldots, m \), then: \( N < \#\mathcal{Q} \leq \#\phi_i^j \leq M \) and \( 2) \phi_i^j < \phi_i^{j+1} \subseteq \mathcal{Q}_i \), where \( \phi_i^{j+1} = \{\phi_i^j \cap \phi_i^{j+1}\} \in (\{E_i\}, \phi_i^{j+1} = \operatorname{argmax}(\#E_i)) \).

Because network \( \{a_i^j, b_i^j, \beta_i^j\}_{i=1}^L \) is approximate to set \( \mathcal{Q} \) and \( \phi_i^{j+1} \) with 0 output error according to (15) and (16).

In theory, one network can only approximate one unique continuous function and one NN cannot approximate two different continuous functions (if these two continuous functions consist of one hybrid system). Therefore, we can get the result \( \mathcal{Q} \) and \( \phi_i^{j+1} \) from the same continuous function (\( \mathcal{Q} \subseteq \phi_i^{j+1} \)).

If \( \mathcal{Q} \subseteq \phi_i^{j+1} \), we can define \( Y' = \mathcal{Q}/\phi_i^{j+1} = Y' \). According to the definition of \( \phi_i^{j+1} \) and \( \mathcal{Q} \), we can get: \( Y' \subseteq \mathcal{X}, Y' \subseteq \mathcal{Q}_i \). However, because \( \phi_i^{j+1} \subseteq \mathcal{Q}_i \), we can get \( Y' = \emptyset \) and \( \phi_i^{j+1} \).

**Remark 5:** Theorem 1 points out a way to identify unknown piecewisely or switched nonlinear systems and indicates how to design a method to estimate continuous clusters, classify the training data points into several continuous clusters, and approximate hybrid nonlinear/linear systems via NNs. According to Theorem 1, if a parent cluster \( \phi_i^j \) is selected from the same maximum submodel block

\[ (y_i, x_i) \in \phi_i^j/\phi_i^{j+1} \]
\((\phi_{g}^{j})_{j=1}^{N} = \{x_{k}, y_{k}\}_{k=1}^{N} \subset Q_{g}\), we can obtain offspring cluster \(\phi_{g}^{j+1}\) that belongs to the same maximum submodel block \((\phi_{g}^{j+1})_{j=1}^{N} \subset Q_{g}\) and \(\#\phi_{g}^{j+1} > \#\phi_{g}^{j}\). Therefore, if repeated \(s_{1}\) times, we can get \(\phi_{g}^{s_{1}} \subset \phi_{g}^{s_{1}+1} \subset \cdots \subset \phi_{g}^{j+1} = Y = Q_{g}\).

B. Parent Cluster \(\phi_{1}^{j}\) Belongs to a Hybrid System

Theorem 2: Given maximum submodel blocks of training data \(\mathcal{X}: Q_{1} = \{(x_{i}, y_{i})\}_{i=1}^{M}, i = 1, \ldots, c\) and \(M \in \mathbb{Z}^{+}\). Given a parent cluster \(\phi_{n}^{0} \equiv \phi_{1}^{0} = N, N \ll M\) and its \(s_{1}\) offspring clusters \(\phi_{1}^{1}, \ldots, \phi_{1}^{s_{1}}\) belong to a hybrid system: \(\phi_{1}^{0}, \ldots, \phi_{1}^{s_{1}} \subset Q_{g}, \phi_{1}^{1}, \ldots, \phi_{1}^{s_{1}} \subset \mathcal{X}, g = 1, \ldots, c\). Then: 

1) \(\#\phi_{1}^{0} \leq \#\phi_{1}^{1} \leq \ldots \leq \#\phi_{1}^{s_{1}}\) and 2) the upper bound of \(\#\phi_{1}^{j} \ll M\).

Proof: According to the definition of offspring cluster, it is obvious that \(\#\phi_{1}^{0} \leq \#\phi_{1}^{1} \leq \ldots \leq \#\phi_{1}^{s_{1}}\). Thus, we only prove result (2).

Because \(\phi_{1}^{0}\) is a parent cluster, we can get \(\#\phi_{1}^{0} = N = L_{i}\). If \(\#\phi_{1}^{j+1} = M\), we can get \(\#\phi_{1}^{j+1} = M \gg L_{i}\). In theory, NNs with \(L_{i}\) hidden-node-numbers can fit any data points that are from continuous systems or NNs with \(L_{i}\) hidden-node numbers can fit any \(L_{i}\) data points regardless of whether the data points are from the same continuous system or not. According to the definition of offspring cluster, an SLFNs with \(L_{i}\) hidden-node-numbers can approximate the cluster \(\phi_{1}^{j}\) with zero error. Because \(\#\phi_{1}^{s_{1}} = M \gg L_{i}\), we can get the result that \(\phi_{1}^{s_{1}}\) is from the same maximum submodel block. However, according to the condition of Theorem 2, the cluster \(\phi_{1}^{s_{1}}\) belongs to a hybrid system. We get the following result: the upper bound of \(\#\phi_{1}^{j} \ll M\).

Remark 6: According to Theorems 1 and 2, the optimal offspring cluster \(Y\) belongs to the same maximum submodel block if \(s_{1} \rightarrow +\infty\).

C. Simplified Cluster Growing Algorithm

In Sections III-A and III-B, we have proven that based on the cluster growing algorithm, the optimal offspring cluster \(Y\) belongs to the same maximum submodel block. However, there are two problems that make the algorithm hard to use in real applications.

1) Too many loops. If we want to classify and estimate these \(c\) maximum submodel blocks in \(\mathcal{X}\), we have to train \(c \times n \times m \times s_{1}\) times (\(s_{1}\) should be a large value). This loop count is so large that the computation complexity of this algorithm could be too high.

2) According to Theorem 1, when \(m\) NNs are requested to train \(\phi\), the network output error \(E\) should be equal to 0. In practice, this is impossible.

In this section, we modify the original cluster growing algorithm and propose a simplified cluster growing algorithm for real applications. The modifications are as follows.

1) To reduce the loop count of the cluster growing algorithm, we simplify some learning steps and train SLFNs \(c \times s_{2} \times m \times n + c \times s_{3} \times m\) times where \(s_{2}, s_{3} \ll s_{1}\). The learning steps and structure of the simplified cluster growing algorithm are shown in Fig. 3.

2) In real applications, one cannot request that the network output error equal to 0. Here, we modify (7) by adding small positive \(\varepsilon\) as

\[
\phi_{1}^{j+1} = \{\phi_{1}^{j+1} = \{x_{k}, y_{k}\} \in \{E_{i}\}, \phi_{1}^{j+1} = \text{argmax}(\#E_{i})
\]

\[
i = 1, \ldots, n\}
\]

3) As mentioned in [24], in theory, a bidirectional extreme learning machine (B-ELM) with several hidden nodes can reduce network output errors to zero. Thus, to reduce output errors of SLFNs to \(\varepsilon\) as quickly as possible, in PLM we use B-ELM to train each SLFN.

IV. CLUSTER ESTIMATION FOR TESTING DATA POINTS

The classification and estimation steps described in Section III returns an estimate \(c\) of the number of submodels, and the partitions of indices \(\mathcal{K}_{i}, i = 1, \ldots, c\), characterizing the \(c\) feasible submodels \(Q_{1}, \ldots, Q_{c}\) extracted from (1) and (2). These provide the classifications of the \(N\) training data points regardless of whether the data points are from the same continuous system or not. Therefore, if repeated \(s_{1}\) times, we can get \(\phi_{g}^{s_{1}} \subset \phi_{g}^{s_{1}+1} \subset \cdots \subset \phi_{g}^{j+1} = Y = Q_{g}\).

**Algorithm 1** Pseudocode for PLM

Given: Training data \(\mathcal{X}, \gamma, s_{2}, s_{3} \in \mathbb{Z}^{+}\) let \(n_{4} = 1, \mathcal{X}_{n_{4}} = \mathcal{X}\).

while \(#\mathcal{X}_{n_{4}} > (\#\mathcal{X}))/10\) do

for \(n_{2} = 1\) to \(n\) do

Select \(q\) samples from \(\mathcal{X}_{n_{4}}\) randomly and set this cluster as \(\phi_{n_{2}}^{0}\) according to step 1.

Obtain \(\phi_{n_{2}}^{1}\) according to step 2.

for \(n_{1} = 2\) to \(s_{3}\) do

Obtain \(\phi_{n_{2}}^{n_{1}}\) according to (7).

end for

end for

Obtain \(n\) clusters \(\phi_{1}^{1}, \ldots, \phi_{n}^{2}\) and compute \(Y\) according to definition 3.

Set \(D_{n_{4}}^{1} = Y\) as a parent cluster.

for \(n_{3} = 2\) to \(s_{3}\) do

Obtain offspring cluster \(D_{n_{4}}^{n_{3}}\) of \(D_{n_{4}}^{1}\).

end for

let \(Q_{n_{4}} = D_{n_{4}}^{n_{3}}\)

update \(\mathcal{X}_{n_{4}} = \mathcal{X}_{n_{4}} / Q_{n_{4}}\)

let \(n_{4} = n_{4} + 1\)

end while

The classification and estimation steps described in Section III returns an estimate \(c\) of the number of submodels, and the partitions of indices \(\mathcal{K}_{i}, i = 1, \ldots, c\), characterizing the \(c\) feasible submodels \(Q_{1}, \ldots, Q_{c}\) extracted from (1) and (2). These provide the classifications of the \(N\) training data points regardless of whether the data points are from the same continuous system or not. Therefore, if repeated \(s_{1}\) times, we can get \(\phi_{g}^{s_{1}} \subset \phi_{g}^{s_{1}+1} \subset \cdots \subset \phi_{g}^{j+1} = Y = Q_{g}\).
Fig. 3. Structure of proposed progressive learning method.

data points into the $c$ clusters $Q_i = \{ (y_k, x_k) : k \in K_i \}$, $i = 1, \ldots, c$. In this paper, M-SVM [25] is used to train these $c$ clusters $Q_1, \ldots, Q_c$. Then, all testing data points can be classified by this trained M-SVM. Finally, we can use $c$ NNs obtained in Section III to generate each output of the testing data point. Fig. 4 shows the structure of the testing data cluster estimation.

V. PERFORMANCE VERIFICATION

To examine whether our proposed PLM can be used both in linear and nonlinear hybrid system approximation, we test it on numerical linear hybrid systems and real nonlinear hybrid systems. In Section V-A, we test PLM on PWA linear systems and switched linear systems. In Sections V-B and V-C, we test PLM on a numerical nonlinear hybrid example and on real nonlinear hybrid systems. In Section V-E, we discuss the sensitivity of some parameters for PLM. In Section V-F, we discuss the generalization performance of PLM via other learning methods.

Furthermore, a performance comparison has also been conducted in this section. In Section V-A3, we compare the generalization performance of PLM with several linear hybrid system identification methods. In Section V-D, we compare the generalization performance of PLM with two NNs including I-ELM and B-ELM. In Section V-G, we carry out some compared experiment based on held-out set test. To test whether there are performance differences between the PLM and other recent methods, one of the statistical significance tests named $t$-test are used in the previous sections.

The $t$-test is a statistical method used to evaluate the significant difference between two algorithms. The $t$-value will be positive if the first algorithm is better than the second, and it is negative if it is poorer. The $t$-value is defined as follows:

$$ t = \frac{\hat{\varsigma}_2 - \hat{\varsigma}_1}{\sqrt{\frac{\hat{\psi}_1^2}{w+1} + \frac{\hat{\psi}_2^2}{w+1}}} $$

Fig. 4. Structure of testing data cluster estimation.
where \( \zeta_1 \) and \( \zeta_2 \) are the mean values of the first and second methods, respectively; \( \vartheta_1 \) and \( \vartheta_2 \) are the standard deviations of the first and second methods, respectively; and \( \psi \) is the value of the degrees of freedom and in our test, \( \psi = 49 \). When the \( t \)-value is higher than 1.645 (\( \psi = 40 \)), there is a significant difference between the two algorithms with a 95\% confidence level.

The \( u \) for all the testing systems except Sections V-B and V-C in this section are generated randomly from standard norm distribution and \([-4, 4]\) range is used, i.e., we set \( u = 4 \times 2 \times (\text{rand} - 0.5) \) in MATLAB code. While \( u \) in Sections V-B and V-C is generated from standard norm distribution and \([-1, 1]\) range is used, i.e., we set \( u = 2 \times (\text{rand} - 0.5) \) in MATLAB code. The Gaussian noise \( e \) for all the testing systems except (29) is also generated from standard norm distribution \( u(k) \sim N(0, 1) \) [we set \( e = 2 \times (\text{rand} - 0.5) \) in MATLAB code]. While \( e \) in (29) is generated standard normal with truncated at \([0, -1]\) [we set \( e = 2 \times (\text{rand} - 1) \) in MATLAB code]. All data sets have been preprocessed in the following way.

1) To indicate the detail situation of solution by figures, the experimental results in Sections V-A1 and V-A2 are obtained by one trial.
2) In Sections V-A3 and V-B–V-F, all data sets have been preprocessed in the same way. In each trail, we first generate \( u(k) \) randomly and obtain a data set, then some of the data set are used to create the training set and the remaining is used for the test set.
3) In Section V-G, we consider of testing on an independent held-out set. In each trail, we first generate \( u(k) \) randomly and obtain a data set, then 3/5 of the data set are used to create the training set, then other 1/5 is used for the validation set to select parameter, and remaining 1/5 is used for the test set.

The input data are normalized into \([-1, 1]\), while the output data for regression are normalized into the range \([0, 1]\). For all the sections in Section V except Section V-D, 10 sigmoid hidden nodes are used in PLM, while in Section V-D, four sigmoid hidden nodes are used.

The experiments of different algorithms on all the results are conducted in MATLAB 2009a running on the same Windows 7 machine with at 16 GB of memory and E3-1230 processor. The codes used for LSSVM\(^3\) and ELM\(^4\) are downloaded from the Internet.

\(^3\)http://www.esat.kuleuven.be/cova/lssvm/  
\(^4\)http://www.extreme-learning-machines.org/

A. Approximating General Linear Hybrid Systems: Hinging Hyperplane ARX, Linear PWA System, and Switched Linear System

This simulation study is carried out to demonstrate the efficacy of our proposed algorithm in approximating general linear hybrid systems. The generalization performances obtained by the proposed PLM are given in Sections V-A1 and V-A2, and the compared generalization performance results are shown in Section V-A3.

1) Hinging Hyperplane ARX: The HHARX models are a kind of PWA model [22]. In this simulation, we consider the following HHARX model [22]:

\[
y(k) = 0.8y(k - 1) + 0.4u(k - 1) - 0.1 + \max\{-0.3y(k - 1) + 0.6u(k - 1) + 0.3, 0\}. \quad (21)
\]

This equation can be rewritten as a PWARX model in (22), as shown at the bottom of this page.

In this paper, system (21) is simulated with initial regression vector \( x(0) = [y(0), u(0)] = [0, 0]^T \) and \( u(k) \) uniformly distributed in the interval \([-4, 4]\). The regression vector \( x(k) = [y(k - 1), u(k - 1)] \) of training data set contains \( N = 600 \) data points, of which 279 and 321 are generated by each of the two models, respectively. The bound \( \varepsilon \) chosen is equal to 0.01. The learning process with \( s_2, s_3 = 10 \) provides the correct number \( c = 2 \) of submodels and clusters containing 263 and 307 data points, respectively. Only 30 data points are nonclassified. In these 570 classified data points, only 19 data points are misclassified. The 96.67\% of the data points are correctly classified.

The model is validated by computing the residuals on \( N = 399 \) validation data of 175 and 224, which are used to test our proposed method. At this stage, the classification of the data points consists of clusters with 163 and 236 data points (Figs. 5 and 6). Only 14 data points have been misclassified, and 96.49\% of testing data points are classified correctly. The plot of the residuals, desired output, and simulated output are shown in Figs. 7–9.

2) General Linear PWA System: Similar to Remark 2, this example also shows that no matter which expression equation is used to generate training set from the same PWA model, the maximum submodel blocks of this PWA model is unique. In this example, two different expressions (4) and (5) are used to illustrate and clarify that the number of maximum submodel blocks \( c \) is unique and \( c = 2 \) not 4 in this PWA model (4) and (5).

\[
y(k) = \begin{cases} 
0.8y(k-1)+0.4u(k-1)-0.1+\{0.3y(k-1)+0.6u(k-1)+0.3\} & \text{if } -0.3y(k-1)+0.6u(k-1)+0.3 > 0 \\
0.8y(k-1)+0.4u(k-1)-0.1 & \text{if } -0.3y(k-1)+0.6u(k-1)+0.3 \leq 0 
\end{cases} 
\quad \text{model 1}
\]

\[
y(k) = \begin{cases} 
-0.4y(k-1)+u(k-1)+1.5 & \text{if } 4y(k-1)-u(k-1)+10 > 0 \\
0.5y(k-1)-u(k-1)-0.5 & \text{if } 4y(k-1)-u(k-1)+10 \geq 0 \text{ and } 5y(k-1)+u(k-1)-6 \leq 0 \\
-0.3y(k-1)+0.5u(k-1)-1.7 & \text{if } 5y(k-1)+u(k-1)-6 > 0 
\end{cases} 
\quad \text{model 2}
\]

\[
y(k) = \begin{cases} 
0.5y(k-1)-u(k-1)-0.5 & \text{if } 4y(k-1)-u(k-1)+10 \geq 0 \\
-0.3y(k-1)+0.5u(k-1)-1.7 & \text{if } 5y(k-1)+u(k-1)-6 > 0 
\end{cases} 
\quad \text{model 3}
\]

\[
y(k) = \begin{cases} 
-0.4y(k-1)+u(k-1)+1.5 & \text{if } 4y(k-1)-u(k-1)+10 > 0 \\
0.5y(k-1)-u(k-1)-0.5 & \text{if } 4y(k-1)-u(k-1)+10 \geq 0 \text{ and } 5y(k-1)+u(k-1)-6 \leq 0 \\
-0.3y(k-1)+0.5u(k-1)-1.7 & \text{if } 5y(k-1)+u(k-1)-6 > 0 
\end{cases} 
\quad \text{model 4}
\]

\[
y(k) = \begin{cases} 
0.5y(k-1)-u(k-1)-0.5 & \text{if } 4y(k-1)-u(k-1)+10 \geq 0 \\
-0.3y(k-1)+0.5u(k-1)-1.7 & \text{if } 5y(k-1)+u(k-1)-6 > 0 
\end{cases} 
\quad \text{model 5}
\]

\[
y(k) = \begin{cases} 
0.5y(k-1)-u(k-1)-0.5 & \text{if } 4y(k-1)-u(k-1)+10 \geq 0 \\
-0.3y(k-1)+0.5u(k-1)-1.7 & \text{if } 5y(k-1)+u(k-1)-6 > 0 
\end{cases} 
\quad \text{model 6}
\]

\[
y(k) = \begin{cases} 
0.5y(k-1)-u(k-1)-0.5 & \text{if } 4y(k-1)-u(k-1)+10 \geq 0 \\
-0.3y(k-1)+0.5u(k-1)-1.7 & \text{if } 5y(k-1)+u(k-1)-6 > 0 
\end{cases} 
\quad \text{model 7}
\]
This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

YANG et al.: NEW APPROACH FOR GENERAL HYBRID SYSTEM APPROXIMATION

Fig. 5. True partition of the testing data set generated from (21).

Fig. 6. Estimated partition of the testing data set generated from (21).

Fig. 7. Plot of the residuals on testing data generated from (21).

Fig. 8. Partial data set (1st–50th sample) generated from (21) used for estimation and validation. Red and blue lines represent the desired system output and the simulated system output.

Fig. 9. Full data set (1st–400th sample) generated from (21) used for estimation and validation. Red and blue lines represent the desired system output and the simulated system output.

Fig. 10. True partition of the training data points generated from (4).

Fig. 11. True partition of the training data points generated from (4).

Fig. 12. Plot of the residuals, desired output, and simulated output are shown in Figs. 12–14.

This plant, taken from [23] proves the equivalence of (4) and (5). The PWA state space equation (4) is simulated with initial state $w(0) = [w_1, w_2] = [0.5, 0.7]^T$ and $u(0)$ uniformly distributed in the interval $[-4, 4]$. The regression vector $x(k)$ of the training data set, which is shown in Fig. 10, contains $N = 600$ data points of 327 and 273, generated by each of the two subsystems, respectively. The bound $\varepsilon$ chosen is equal to 0.01. The learning process with $s_2 = 10$, $s_3 = 10$, and $q = 10$ provides the correct number $c = 2$ of submodels and clusters containing 325 and 275 data points, respectively. Only eight data points are misclassified.

The model is validated by computing the residuals on $N = 399$ data points, of which 212 and 187 are used to test our proposed method. At this stage, the classification of the data points consists of clusters with 205 and 195 data points, respectively (Fig. 11). Only 11 data points have been misclassified, and 97.24% of data points are classified correctly. The plot of the residuals, desired output, and simulated output are shown in Figs. 12–14.

For (5), it can be noted that the PWARX model has four submodels. However, based on Remark 2, we illustrate that the number of maximum submodel partitions is 2 not 4, because (5) is, in fact, the same as (4) based on the analysis of [23]. In this test, the PWARX model is simulated with the same input, using the first two outputs, $y(0)$ and $y(1)$, of the PWA state space model to form the initial regression vector $x(0) = [y(1), y(0), u(1), u(0)]$. The regression vector can be defined as $x(k) = [y(k - 1), y(k - 2), u(k - 1), u(k - 2)]$ that contains $N = 600$ data points (Fig. 15), of 107, 179, 132, and 182 are generated by each of the four models, respectively. The bound $\varepsilon$ is equal to 0.02. The learning process with $s_2 = 10$, $s_3 = 10$, and $q = 10$ provides the correct number $c = 2$ of submodels and clusters containing 267 and 254 data points, respectively. Interestingly, in these two maximum submodel blocks we found that all data points in maximum submodel block 1 come from models 1 and 4 in (5), and all data points in maximum submodel block 2 come from models 2 and 3 in (5). The model is validated by computing the residuals on $N = 399$ data points of which 83, 119, 77, and 120 are used to test our proposed method. At this stage,
the classification of the data points consists of clusters with 216 and 183 data points, respectively (Figs. 15 and 16). The plot of the residuals, desired output, and simulated output are shown in Figs. 17–19.

3) Generalization Performance Comparison of Linear Hybrid System Approximation Methods: The performances comparison between PLM and other linear hybrid approximation methods (bounded-error approach (BEA) [20], Identification method via mixed-integer linear programs (IMLP) [22], K-linreg [26], ME-MCS [27], PE-MCS [27], Algebraic method [7], and two sparse optimization-based methods [14], [15]) on two PWA linear systems (21) and (23) and three switched linear systems (24), (26), and (27) are given in Tables I–IV. In these tables, the close results obtained by different algorithms are underlined and the apparent better results are shown in boldface. All these results are obtained over 20 trials for all cases.

The first PWA system is the HHARX model defined in (21) and the second is taken from [20] and defined in (23). The regression vector \( x(k) = [y(k-1), u(k-1)] \) of the training data set contains \( N = 600 \) data points, and the model is validated by computing the residuals on \( N = 399 \) validation data.

We consider the first switched linear system taken from [27] and also considered in [7]. The aim is to recover, from \( N = 1000 \) samples, consisting 600 samples in training data and 399 samples in testing data, the parameters \( \theta_1 = [-0.9, 0.7]^T \) and \( \theta_2 = [1, -1]^T \) of a dynamical system that can switch between two models regularly [we set \( m = (\text{mod}(1:1000,60)>30)+1 \) in MATLAB code], with a continuous state \( [y(k), u(k)] \) and a zero-mean Gaussian input with unit variance \( u(k) = [u_1(k), u_2(k)], u(k) \in [0, 1] \). This switched linear system is defined as

\[
y(k) = \theta_1(m)u_1(k) + \theta_2(m)u_2(k) + a \times e(k)
\]
regularly, we set this switched system such that \( m \) is arbitrarily switching between two models. The regression vector \( \mathbf{x}(k) = [y(k - 1), y(k - 2), u(k - 1)] \) of the training data set contains \( N = 600 \) data points, and the model is validated by computing the residuals on \( N = 400 \) validation data. The third switched linear system is defined as

\[
y(k) = \begin{cases} 
0.2y(k - 1) + 0.24y(k - 2) + 2u(k - 1) + a \times e(k), & \text{model 1} \\
-1.4y(k - 1) - 0.53y(k - 2) + u(k - 1) + a \times e(k), & \text{model 2}
\end{cases}
\]  

(27)

Because IMLP and BEA only work for PWA linear system identification, in this section, we test these two methods on these two PWA linear systems (21) and (23). As in [20], for the two PWA linear systems we select the same parameters in a bound-error algorithm as: \( C = 10, T_0 = 100, \rho = 0.7, \) and \( \delta = 0.05. \) For the IMLP method, we directly use the performance result in [22] because according to [22], the estimated model obtained by IMLP is

\[
y(k) = 0.83y(k - 1) + 0.34u(k - 1) - 0.20
\]

(28)

Thus, we can easily achieve the RMSE between desired output \( y(k) \) and simulated output \( \hat{y}(k). \) In [22], IMLP is used for HHARX identification; thus, in this section, we only discuss the performance of IMLP for HHARX systems.

Tables I–IV show the performance comparison of BEA, IMLP, K-linreg, ME-MCS, PE-MCS, Algebraic method, and sparse optimization-based methods. Compared with other linear hybrid system approximation methods, the proposed PLM has two significant advantages.

1) Different from other linear hybrid system approximation methods, which are only suitable for switched systems or for PWA systems, the proposed method can approximate any linear hybrid system, such as PWAARX systems, general PWA systems, and switched linear systems. Furthermore, different from methods that require true submodel numbers, the proposed method can provide similar or better generalization performance than other methods without needing to know true submodel numbers, which can be seen in Table IV.

2) According to our results, although the computational time obtained by PLM is generally greater than that of other methods, PLM always achieves better performance than other methods when high noise levels are added to these hybrid systems. All compared methods can approximate this system from the low noise data, but only the proposed method can achieve acceptable errors for all noise levels. Take (27) for example: when \( a = 0.2, \) the training RMSE of PLM, ME-MCS, and PE-MCS are similar, but when \( a = 1, \) the training RMSE of ME-MCS, PE-MCS, K-linreg, and method [11] are about 28 times, 20 times, 9 times, and 30 times larger than that of PLM, respectively. When the \( t \)-value is higher than 1.645 (\( \nu = 40 \)), there is a significant difference between the two algorithms with a 95% confidence level. The mean cost values, standard
deviation values and t-value between the PLM and other compared methods are shown in Table III. We see that most t-values in this table are higher than 1.645 especially for high-noise data set. Thus, PLM has better robustness to noise.

### B. Approximating General Hybrid Nonlinear Systems

To give another example of the described proposed algorithm, we consider the problem of approximating general hybrid nonlinear systems. Consider the general nonlinear hybrid state space model, as shown in (29), at the bottom of the page, with $s = 2$, $\lambda_1 = 0.4$, $\lambda_2 = 1$, $h = [h_1 h_2] = [1, 2]$, $x(k) = [x_1(k), x_2(k)]$, $a = 0.2$, and $v = 0.3$.

This hybrid nonlinear model is simulated with initial state $w(0) = [w_1(0), w_2(0)] = [0.5 – 0.5]^T$. The regression vector $x = w$ of the training data set contains $N = 600$ data points, of which 243 and 357 are generated by each of the two models, respectively. The bound $\epsilon$ chosen is equal to 0.01. The learning process with $s_2 = 10$, $s_3 = 10$, and $n = 10$ provides the correct number $c = 2$ of submodels, and clusters containing 247 and 327 data points, respectively.

The model is validated by computing the residuals on $N = 399$ data points, of which 157 and 242 are used to test our proposed method. At this stage, the classification of the data points consists of clusters with 167 and 232 data points, respectively. Only 20 data points have been misclassified, and 94.99% of data points are classified correctly. The plot of the residuals, desired output, and simulated output are shown in Figs. 20–22.

### C. Experiments on Real Data

To test the performance of our approach on real data, we apply PLM to benchmark regression problems from the UCI database [28] (Table V). In Experiment 1, we consider two benchmark regression problems: the bank problem and the abalone problem, because these two problems have the same dimensions and approximate number of samples (4499 and 4177) (Table V). We generate a high-dimension piecewise nonlinear system by mixing the bank problem with the abalone problem. In Experiment 2, we consider three benchmark regression problems: bank, abalone, and census (house8L) because these three problems also have the same
dimensions. Then, a 9-dimension piecewise nonlinear system is generated by mixing house8l and bank with abalone. Classifying, estimating, and obtaining a complete description of such complex hybrid systems is rather challenging.

1) Experiment 1: This 9-dimension hybrid system consists of two regression problems (the bank problem and the abalone problem), and the high-dimension piecewise nonlinear system is generated by mixing the bank problem with the abalone problem. The mixed steps are shown as follows.

Step 1: Obtaining two B-ELM NNs \( \{X_i^1, b^1_i, \alpha_i^1, \beta_i^1, \rho_i^1\}_{i=1}^{40} \), \( \{X_i^2, b^2_i, \alpha_i^2, \beta_i^2, \rho_i^2\}_{i=1}^{40} \) with 40 hidden-node numbers, which have already been trained by these two real regression problems (bank and abalone).\(^5\)

Step 2: Generating inputs \( u(k) \) randomly from \([-1, 1]\).

Step 3: Establishing an estimated piecewise continuous

\(^5\)For all UCI data sets, because we do not know the precise mathematical expression. Thus, we have to first approximate each data set by NN to establish a nonlinear hybrid system.
points, and the training data set contains \( N \) points. The bounds \( \varepsilon \) normalized into \([-\infty, 0] \) distributed on \([-\infty, 0] \).

The output data were normalized into \([-1, 1]\), the noise signal \( e(k) \) is uniformly distributed on \([-0.1, 0.1]\), and average results were obtained over 20 trials,\(^6\) the testing data set contains \( N = 200 \) data points, and the training data set contains \( N = 1000 \) data points. The bounds \( \varepsilon \) chosen equal 0.03 and \( s_2 = 30, s_3 = 10, q = 4 \), respectively.

Table VI shows the generalization performance, classification accuracy, and misclassified data numbers of proposed PLM with four sigmoid hidden nodes. As observed from this table, we find that the number of submodels in the training process is not the same as the number of submodels in the testing process. In theory, if the learning accuracy of the network is reduced to 0 (the bound \( \varepsilon \) can also be set to 0), any hybrid nonlinear/linear system can be classified into \( c \) continuous nonlinear systems, and the number of submodels in the training process is the same as the number of submodels in the testing process. In other words, if SLFNs reduce residual error to 0 with finite hidden nodes, the obtained submodel numbers, both in training process and in testing process, is the same and is equal to the number of maximum submodel blocks. However, for this complex hybrid system, one cannot request that the network output error equal 0. Even for continuous nonlinear systems, such as the abalone problem or other real data problems, one cannot request the network output error to equal 0.

The number of misclassified data points is obtained by summing the number of misclassified data points in each submodel during the testing process. Based on this definition, in Table VI, the proposed method classifies bank data points and abalone data points, respectively, from the hybrid system (33) with nearly 96% classification accuracy. We also find that our proposed method can solve this challenging task and the testing RMSE be reach a small value.

2) Experiment 2: To test the performance of our proposed method on a large data set, in this section, we test a hybrid nonlinear system, which consists of three regression problems (house8L, bank, and abalone). All data sets from these three regression problems have been preprocessed as the same as Experiment 2, and an estimated nonlinear hybrid system is established as follows:

\[
y(k) = \begin{cases} 
\sum_{i=1}^{40} \beta_i^1 H(\alpha_i^1, b_i^1, u(k)) + e(k) & \text{if } y(k-1) > 0.5 \\
\sum_{i=1}^{40} \beta_i^2 H(\alpha_i^2, b_i^2, u(k)) + e(k) & \text{if } (k-1) \leq 0.5 
\end{cases}
\]

(30)

All data sets from these two regression problems have been preprocessed in the same way. The output data were normalized into \([0, 1]\), the noise signal \( e(k) \) is uniformly distributed on \([-0.1, 0.1]\), and average results were obtained over 20 trials,\(^6\) the testing data set contains \( N = 2000 \) data points, while the training data set contains \( N = 400 \) data points. The bound \( \varepsilon \) is chosen as equal to 0.03 and \( s_2, s_3 \) is chosen as equal to 30 and 10, respectively. Table VII shows the generalization performance and classification accuracy of the proposed method, with four sigmoid hidden nodes. As observed from this table, we find that the proposed method estimates and classifies house8L data points, bank data points, and abalone data points from hybrid

\[
\text{Table V}
\begin{tabular}{|c|c|c|}
\hline
Datasets & #Attri & #data points \\
\hline
bank & 8 & 4499 \\
Abalone & 8 & 4177 \\
house8L & 8 & 27784 \\
\hline
\end{tabular}
\]

\(^6\)In each trial, we first generated \( y(k) \) and \( u(k) \) by Steps 1–3 and then used PLM to approximate a nonlinear hybrid system.
systems (31) with nearly 93% classification accuracy. We also find that our proposed method can solve this challenging task and the testing RMSE of the proposed method can be reached to a small value.

### D. Generalization Performance Comparison of Nonlinear Hybrid System Approximation Methods

The performance comparison between PLM and ELM methods (I-ELM [29], B-ELM [24]) on five nonlinear hybrid systems is given in Table VIII. In this table, the close results obtained by different algorithms are underlined and the apparent better results are shown in boldface. All these results are obtained over 10 trials for all cases. These five nonlinear hybrid systems consist of one nonlinear hybrid state space system (29) and four real nonlinear hybrid systems (30)–(33).

#### TABLE VI

**Generalization Performance and Classification Accuracy of Proposed Method in Experiment 1**

<table>
<thead>
<tr>
<th>nth trials</th>
<th>number of sub-model in training process</th>
<th>number of sub-model in testing process</th>
<th>testing RMSE of sub-model 1 (number of data)</th>
<th>testing RMSE of sub-model 2 (number of data)</th>
<th>testing RMSE of sub-model 3 (number of data)</th>
<th>testing RMSE of sub-model 4 (number of data)</th>
<th>classification accuracy</th>
<th>number of misclassified data points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>5</td>
<td>3</td>
<td>0.0168 (114)</td>
<td>0.0212 (85)</td>
<td>0.0255 (3)</td>
<td>N/A</td>
<td>0.0191</td>
<td>96.50%</td>
</tr>
<tr>
<td>2nd</td>
<td>6</td>
<td>3</td>
<td>0.0177 (99)</td>
<td>0.0281 (89)</td>
<td>0.0366 (12)</td>
<td>N/A</td>
<td>0.0266</td>
<td>95.50%</td>
</tr>
<tr>
<td>3rd</td>
<td>6</td>
<td>4</td>
<td>0.0163 (84)</td>
<td>0.0243 (95)</td>
<td>0.0424 (16)</td>
<td>N/A</td>
<td>0.0244</td>
<td>96.00%</td>
</tr>
<tr>
<td>4th</td>
<td>6</td>
<td>3</td>
<td>0.0160 (105)</td>
<td>0.0310 (76)</td>
<td>0.0544 (19)</td>
<td>N/A</td>
<td>0.0245</td>
<td>96.50%</td>
</tr>
<tr>
<td>5th</td>
<td>4</td>
<td>3</td>
<td>0.0245 (85)</td>
<td>0.0247 (101)</td>
<td>0.0382 (10)</td>
<td>N/A</td>
<td>0.0246</td>
<td>98.00%</td>
</tr>
<tr>
<td>6th</td>
<td>4</td>
<td>3</td>
<td>0.0184 (115)</td>
<td>0.0206 (81)</td>
<td>0.0228 (4)</td>
<td>N/A</td>
<td>0.0208</td>
<td>93.50%</td>
</tr>
<tr>
<td>7th</td>
<td>6</td>
<td>4</td>
<td>0.0233 (98)</td>
<td>0.0306 (95)</td>
<td>0.0262 (15)</td>
<td>N/A</td>
<td>0.0191</td>
<td>96.00%</td>
</tr>
<tr>
<td>8th</td>
<td>6</td>
<td>4</td>
<td>0.0191 (107)</td>
<td>0.0271 (82)</td>
<td>0.0392 (7)</td>
<td>N/A</td>
<td>0.0215</td>
<td>97.00%</td>
</tr>
<tr>
<td>9th</td>
<td>6</td>
<td>4</td>
<td>0.0170 (99)</td>
<td>0.0287 (79)</td>
<td>0.0331 (13)</td>
<td>N/A</td>
<td>0.0378</td>
<td>95.00%</td>
</tr>
<tr>
<td>10th</td>
<td>6</td>
<td>4</td>
<td>0.0151 (109)</td>
<td>0.0315 (60)</td>
<td>0.0374 (23)</td>
<td>N/A</td>
<td>0.0208</td>
<td>96.50%</td>
</tr>
</tbody>
</table>

#### TABLE VII

**Generalization Performance and Classification Accuracy of Proposed Method in Experiment 2**

<table>
<thead>
<tr>
<th>nth trials</th>
<th>number of sub-model in training process</th>
<th>number of sub-model in testing process</th>
<th>testing RMSE of sub-model 1 (number of data)</th>
<th>testing RMSE of sub-model 2 (number of data)</th>
<th>testing RMSE of sub-model 3 (number of data)</th>
<th>testing RMSE of sub-model 4 (number of data)</th>
<th>classification accuracy</th>
<th>number of misclassified data points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>8</td>
<td>5</td>
<td>0.0297 (146)</td>
<td>0.0209 (145)</td>
<td>0.0392 (71)</td>
<td>N/A</td>
<td>0.0402</td>
<td>91.75%</td>
</tr>
<tr>
<td>2nd</td>
<td>6</td>
<td>4</td>
<td>0.0166 (163)</td>
<td>0.0281 (113)</td>
<td>0.0511 (117)</td>
<td>0.0014 (8)</td>
<td>0.0014</td>
<td>92.05%</td>
</tr>
<tr>
<td>3rd</td>
<td>7</td>
<td>5</td>
<td>0.0167 (176)</td>
<td>0.0328 (95)</td>
<td>0.0195 (102)</td>
<td>0.0371 (21)</td>
<td>0.0443</td>
<td>92.50%</td>
</tr>
<tr>
<td>4th</td>
<td>6</td>
<td>4</td>
<td>0.0196 (164)</td>
<td>0.0179 (152)</td>
<td>0.0186 (67)</td>
<td>N/A</td>
<td>0.0018</td>
<td>93.50%</td>
</tr>
<tr>
<td>5th</td>
<td>7</td>
<td>5</td>
<td>0.0190 (183)</td>
<td>0.0170 (124)</td>
<td>0.0227 (99)</td>
<td>N/A</td>
<td>0.0164</td>
<td>94.25%</td>
</tr>
<tr>
<td>6th</td>
<td>7</td>
<td>5</td>
<td>0.0190 (173)</td>
<td>0.0299 (99)</td>
<td>0.0257 (104)</td>
<td>N/A</td>
<td>0.0264</td>
<td>91.25%</td>
</tr>
<tr>
<td>7th</td>
<td>7</td>
<td>5</td>
<td>0.0171 (159)</td>
<td>0.0270 (112)</td>
<td>0.0206 (121)</td>
<td>0.0308 (24)</td>
<td>0.0241</td>
<td>92.00%</td>
</tr>
<tr>
<td>8th</td>
<td>7</td>
<td>5</td>
<td>0.0184 (183)</td>
<td>0.0323 (132)</td>
<td>0.0359 (86)</td>
<td>0.0464 (14)</td>
<td>0.0344</td>
<td>93.50%</td>
</tr>
<tr>
<td>9th</td>
<td>5</td>
<td>5</td>
<td>0.0162 (138)</td>
<td>0.0203 (175)</td>
<td>0.0212 (78)</td>
<td>0.0324 (6)</td>
<td>0.0420</td>
<td>92.75%</td>
</tr>
<tr>
<td>10th</td>
<td>8</td>
<td>5</td>
<td>0.0194 (150)</td>
<td>0.0311 (125)</td>
<td>0.0408 (94)</td>
<td>N/A</td>
<td>0.0044</td>
<td>91.00%</td>
</tr>
</tbody>
</table>

\[ y(k) = \begin{cases} \sum_{j=1}^{40} \alpha_j H(\mathbf{a}_j, \mathbf{b}_j, \mathbf{u}(k)) + e(k) & \text{if } y(k-1) > 0.5 \\ \sum_{j=1}^{40} \beta_j H(\mathbf{a}_j, \mathbf{b}_j, \mathbf{u}(k)) + e(k) & \text{if } y(k-1) \leq 0.5 \end{cases} \]
As observed from Table VIII, we find that the generalization performance obtained by PLM is much better than that of ELMs. Furthermore, large than 0.25 RMSE implies that traditional single NNs cannot approximate these nonlinear hybrid system at all. These performance comparisons indicate that our proposed PLM can approximate nonlinear hybrid system with small RMSE. From the NNs learning method point of view, different from other traditional single NNs methods only have universal approximation capability for continuous system, this PLM can manages to extend NN to nonlinear hybrid system.

E. Sensitivity of the Parameters $L$ and $\varepsilon$ for PLM

1) Sensitivity of the Parameter $\varepsilon$: In this section, we discuss the sensitivity of the parameter $\varepsilon$ in PLM. The mean classification accuracy and mean generalization performance offered by the PLM with different parameter values $\varepsilon$ for hybrid system approximation are tabulated in Table IX. These test hybrid systems are tested by fixing $m = 10$, $s_2 = 10$, $s_3 = 10$. In these cases, the parameter $\varepsilon$ is not sensitive if $\varepsilon < 0.08$, such as PWA linear system (23), general PWA nonlinear system (29), and switched systems (27) and (26). The mean classification accuracy and mean RMSE with different values of $\varepsilon$ have no significant difference. However, very large subsystems are feasible for two large values of $\varepsilon$ and beyond a certain value the whole system becomes feasible, which corresponds to fitting a linear model to the data set. Hence, for large $\varepsilon$ the identified hybrid model is simple, because it contains very few submodels. However, the submodels do not fit the corresponding data points well, as large errors are tolerated. For example, considering (23) $(a = 0.2)$, if $\varepsilon > 0.12$, all data points will be classified into one subsystems. Thus, PLM can not learn this PWA system successfully. This is why we carry out all experiments using $\varepsilon = [0.01, 0.05]$. We also suggest fixing $\varepsilon$ at <0.05 in general.

2) Sensitivity of the Parameter $L$: We also consider the sensitivity of the parameter $L$ in PLM. The mean test accuracy by PLM with different parameter values $L$ for hybrid system approximation are shown in Table IX. These experimental results are obtained by two different situations: 1) the hidden-node numbers in each NN should be generated randomly and $L \in [5, 10]$ and 2) the hidden-node numbers in each NN equal 10. We find that in Table IX, the mean test accuracies for different $L$ are nearly the same. This indicates that parameter $L$ is not at all sensitive to the generalization performance, and the generalization performance of PLM does not change whenever $L$ changes ($L > 2$). Thus, in PLM, hidden-node numbers in each NN can be chosen randomly at the beginning by the user without affecting the generalization performance in the learning process.

F. Generalization Performance of PLM via Other Network Learning Methods

According to the above statement, our proposed PLM can be considered a multi-NN learning method via B-ELM (PLM-B-ELM). Inspired by the anonymous reviewers’ comments, in this section, we discuss the generalization performance of PLM via other learning methods, including PLM via ELM (PLM-ELM) and PLM via LSSVM (PLM-LSSVM).

For LSSVM, to achieve a good generalization performance, the cost parameter $\gamma$ of LSSVM need to be chosen appropriately. We have tried a wide range of $C$ and $\gamma$. For each data set, similar to [30], we have used 30 different values of $C$ and $\gamma$, resulting in a total of 900 pairs of $(C, \gamma)$. The 30 different values of

### Table VIII

<table>
<thead>
<tr>
<th>Methods</th>
<th>PLM $(s_2 = 10, s_3 = 10)$</th>
<th>I-ELM</th>
<th>B-ELM</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>accuracy</td>
<td>std RMSE</td>
<td>t-value</td>
</tr>
<tr>
<td>equation 29 (a=1)</td>
<td>92.45</td>
<td>0.0117</td>
<td>0.0617</td>
</tr>
<tr>
<td>equation 30</td>
<td>96.00</td>
<td>0.0009</td>
<td>0.0241</td>
</tr>
<tr>
<td>equation 32</td>
<td>96.45</td>
<td>0.0007</td>
<td>0.0158</td>
</tr>
<tr>
<td>equation 33</td>
<td>95.48</td>
<td>0.0015</td>
<td>0.0190</td>
</tr>
<tr>
<td>equation 31</td>
<td>92.48</td>
<td>0.0034</td>
<td>0.0240</td>
</tr>
</tbody>
</table>

### Table IX

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$\varepsilon$</th>
<th>Train RMSE $L = 10$</th>
<th>accuracy $L = 10$</th>
<th>Train RMSE $L \in [5, 10]$</th>
<th>accuracy $L \in [5, 10]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>equation 23 (a = 0.2)</td>
<td>0.01</td>
<td>0.0044</td>
<td>0.0130</td>
<td>0.0050</td>
<td>0.0253</td>
</tr>
<tr>
<td></td>
<td>0.02</td>
<td>0.0063</td>
<td>0.0314</td>
<td>0.0069</td>
<td>0.0387</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>0.0092</td>
<td>0.0001</td>
<td>0.0072</td>
<td>0.0180</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>0.0156</td>
<td>0.0404</td>
<td>0.0176</td>
<td>0.0509</td>
</tr>
<tr>
<td></td>
<td>0.08</td>
<td>0.0237</td>
<td>0.0577</td>
<td>0.0298</td>
<td>0.0589</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.0296</td>
<td>0.0686</td>
<td>0.0310</td>
<td>0.0589</td>
</tr>
<tr>
<td>equation 23 (a = 0.5)</td>
<td>0.02</td>
<td>0.0107</td>
<td>0.0574</td>
<td>0.0170</td>
<td>0.0572</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>0.0137</td>
<td>0.0847</td>
<td>0.0176</td>
<td>0.0635</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>0.0177</td>
<td>0.0628</td>
<td>0.0186</td>
<td>0.0675</td>
</tr>
<tr>
<td></td>
<td>0.08</td>
<td>0.0298</td>
<td>0.0895</td>
<td>0.0295</td>
<td>0.0844</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.0315</td>
<td>0.0841</td>
<td>0.0398</td>
<td>0.0829</td>
</tr>
<tr>
<td>equation 29</td>
<td>0.04</td>
<td>0.0053</td>
<td>0.0318</td>
<td>0.0057</td>
<td>0.1047</td>
</tr>
<tr>
<td></td>
<td>0.05</td>
<td>0.0062</td>
<td>0.0617</td>
<td>0.0063</td>
<td>0.0616</td>
</tr>
<tr>
<td></td>
<td>0.07</td>
<td>0.0094</td>
<td>0.0686</td>
<td>0.0094</td>
<td>0.0700</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.0192</td>
<td>0.0823</td>
<td>0.0112</td>
<td>0.1010</td>
</tr>
<tr>
<td>equation 26 (a = 0.5)</td>
<td>0.03</td>
<td>0.0116</td>
<td>0.9776</td>
<td>0.0101</td>
<td>0.9797</td>
</tr>
<tr>
<td></td>
<td>0.04</td>
<td>0.0142</td>
<td>0.9797</td>
<td>0.0139</td>
<td>0.9512</td>
</tr>
<tr>
<td></td>
<td>0.06</td>
<td>0.0199</td>
<td>0.9776</td>
<td>0.0228</td>
<td>0.8956</td>
</tr>
<tr>
<td></td>
<td>0.08</td>
<td>0.0265</td>
<td>0.9504</td>
<td>0.0285</td>
<td>0.8732</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.0318</td>
<td>0.8140</td>
<td>0.0378</td>
<td>0.8595</td>
</tr>
<tr>
<td>equation 26 (a = 2)</td>
<td>0.03</td>
<td>0.0239</td>
<td>0.9037</td>
<td>0.0201</td>
<td>0.9082</td>
</tr>
<tr>
<td></td>
<td>0.08</td>
<td>0.0329</td>
<td>0.8621</td>
<td>0.0303</td>
<td>0.8698</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.0368</td>
<td>0.8248</td>
<td>0.0397</td>
<td>0.8837</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>0.0108</td>
<td>0.9556</td>
<td>0.0154</td>
<td>0.9276</td>
</tr>
<tr>
<td>equation 27 (a = 1)</td>
<td>0.05</td>
<td>0.0186</td>
<td>0.9111</td>
<td>0.0127</td>
<td>0.9073</td>
</tr>
<tr>
<td></td>
<td>0.07</td>
<td>0.0255</td>
<td>0.8260</td>
<td>0.0279</td>
<td>0.8479</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.0416</td>
<td>0.7347</td>
<td>0.0466</td>
<td>0.7685</td>
</tr>
</tbody>
</table>
TABLE X
AVERAGE GENERALIZATION PERFORMANCE OVER 20 TRAILS [ACCURACY-MEAN TRAIN CLASSIFICATION ACCURACY FOR (24), (26), AND (27), ACCURACY-MEAN TEST RMSE FOR (21) AND (23), TIME-TRAINING TIME, AND TRAIN RMSE-MEAN TRAIN RMSE]

<table>
<thead>
<tr>
<th>Datasets</th>
<th>PLM-B-ELM</th>
<th>PLM-ELM</th>
<th>PLM-LSSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>accuracy</td>
<td>std dev</td>
<td>time(s)</td>
</tr>
<tr>
<td>equation 21</td>
<td>0.0078</td>
<td>0.0090</td>
<td>0.0037</td>
</tr>
<tr>
<td>equation 23(a = 0.2)</td>
<td>0.0130</td>
<td>0.0041</td>
<td>0.0009</td>
</tr>
<tr>
<td>equation 23(a = 0.5)</td>
<td>0.0647</td>
<td>0.0137</td>
<td>0.0105</td>
</tr>
<tr>
<td>equation 24(a = 0.2)</td>
<td>98.92%</td>
<td>0.0165</td>
<td>0.0107</td>
</tr>
<tr>
<td>equation 24(a = 1)</td>
<td>72.42%</td>
<td>0.0052</td>
<td>0.0182</td>
</tr>
<tr>
<td>equation 26 (a = 0.5)</td>
<td>96.12%</td>
<td>0.0033</td>
<td>0.0013</td>
</tr>
<tr>
<td>equation 26(a = 2)</td>
<td>95.16%</td>
<td>0.0161</td>
<td>0.0024</td>
</tr>
<tr>
<td>equation 27(a = 0.2)</td>
<td>98.58%</td>
<td>0.0031</td>
<td>0.0002</td>
</tr>
<tr>
<td>equation 27(a = 1)</td>
<td>95.38%</td>
<td>0.0108</td>
<td>0.0025</td>
</tr>
</tbody>
</table>

TABLE XI
AVERAGE GENERALIZATION PERFORMANCE OVER 20 TRAILS (RANK: 1-BEST AND 3-WORST)

<table>
<thead>
<tr>
<th>Datasets</th>
<th>PLM-B-ELM</th>
<th>PLM-ELM</th>
<th>PLM-LSSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a = 0.2)</td>
<td>mean</td>
<td>0.0040</td>
<td>0.0076</td>
</tr>
<tr>
<td>Std Dev</td>
<td>0.0037</td>
<td>0.0043</td>
<td>0.0029</td>
</tr>
<tr>
<td>T-value</td>
<td>N/A</td>
<td>4.4874</td>
<td>4.0612</td>
</tr>
<tr>
<td>Rank</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>(a = 0.5)</td>
<td>mean</td>
<td>0.0044</td>
<td>0.0063</td>
</tr>
<tr>
<td>Std Dev</td>
<td>0.0009</td>
<td>0.0004</td>
<td>0.0008</td>
</tr>
<tr>
<td>T-value</td>
<td>N/A</td>
<td>13.6412</td>
<td>21.7271</td>
</tr>
<tr>
<td>Rank</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>(a = 1)</td>
<td>mean</td>
<td>0.0017</td>
<td>0.0021</td>
</tr>
<tr>
<td>Std Dev</td>
<td>0.0010</td>
<td>0.0019</td>
<td>0.0009</td>
</tr>
<tr>
<td>T-value</td>
<td>N/A</td>
<td>-0.3858</td>
<td>0.8985</td>
</tr>
<tr>
<td>Rank</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>(a = 0.2)</td>
<td>mean</td>
<td>0.0055</td>
<td>0.0077</td>
</tr>
<tr>
<td>Std Dev</td>
<td>0.0017</td>
<td>0.0021</td>
<td>0.0012</td>
</tr>
<tr>
<td>T-value</td>
<td>N/A</td>
<td>0.5253</td>
<td>-0.0848</td>
</tr>
<tr>
<td>Rank</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>(a = 0.5)</td>
<td>mean</td>
<td>0.0053</td>
<td>0.0016</td>
</tr>
<tr>
<td>Std Dev</td>
<td>0.0013</td>
<td>0.0021</td>
<td>0.0031</td>
</tr>
<tr>
<td>T-value</td>
<td>N/A</td>
<td>3.7219</td>
<td>-7.3650</td>
</tr>
<tr>
<td>Rank</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>(a = 0.2)</td>
<td>mean</td>
<td>0.0011</td>
<td>0.0038</td>
</tr>
<tr>
<td>Std Dev</td>
<td>0.0024</td>
<td>0.0058</td>
<td>0.0037</td>
</tr>
<tr>
<td>T-value</td>
<td>N/A</td>
<td>2.7036</td>
<td>-0.6413</td>
</tr>
<tr>
<td>Rank</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>(a = 0.5)</td>
<td>mean</td>
<td>0.0031</td>
<td>0.0053</td>
</tr>
<tr>
<td>Std Dev</td>
<td>0.0002</td>
<td>0.0009</td>
<td>0.0006</td>
</tr>
<tr>
<td>T-value</td>
<td>N/A</td>
<td>18.8712</td>
<td>-6.6937</td>
</tr>
<tr>
<td>Rank</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>(a = 1)</td>
<td>mean</td>
<td>0.0010</td>
<td>0.0010</td>
</tr>
<tr>
<td>Std Dev</td>
<td>0.0025</td>
<td>0.0017</td>
<td>0.0029</td>
</tr>
<tr>
<td>T-value</td>
<td>N/A</td>
<td>0.2335</td>
<td>-2.2162</td>
</tr>
<tr>
<td>Rank</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Overall Ranking (Average ranking number)</td>
<td>1(1.78)</td>
<td>3(2.44)</td>
<td>1(1.78)</td>
</tr>
</tbody>
</table>

Algorithm 2 Held-Out Testing Steps

for count1 = 1 to 20 do
    Generating input u(k) randomly and obtain 1000 data points \( x \) such that \( k = 1 \) to 1000.
    Set training set \( \{(y_k, x_k)\}_{k=1}^{600} \), validation set \( \{(y_k, x_k)\}_{k=601}^{800} \), and testing set \( \{(y_k, x_k)\}_{k=801}^{1000} \).
end for

for count2 = 1 to 10 do
    Step 1: p = 0.01 \times count2 or p = 1.
    Step 2: Training and obtain the estimated model over training set \( \{(y_k, x_k)\}_{k=1}^{600} \).
    Step 3: Classifying data points from validation set or testing set into corresponding sub-models by using desired classification label.
    Step 4: Calculate validation accuracy \( VR_{count2} \) over validation set \( \{(y_k, x_k)\}_{k=601}^{800} \).
end for

Obtain the average testing accuracy based on \( TR_{count1} \) over the testing set by using the optimal estimated model.

G. Generalization Performance Based on Held-Out Set Test

In this section, we carry out some compared experiment based on held-out set test. Different from above test set, in performances are shown in Table X. Table X shows that PLM via LSSVM generally provides similar or better generalization performance than that of PLM via B-ELM or PLM via ELM. However, because the learning speed of ELM methods can be several hundreds of times faster than LSSVM, we note that the learning of the PLM-B-ELM can be hundreds of times faster than PLM-LSSVM. Thus, if the user wants to achieve better generalization performance without concerning computational time, we suggest using PLM-LSSVM. However, PLM-B-ELM can provide very good compromise among the training time and testing accuracy.

C and y are \(2^{-15}, 2^{-14}, \ldots, 2^{14}, 2^{15}\). For ELM, all the 50 hidden-node parameters are randomly generated based on a uniform distribution. The user-specified parameters are C, where C is chosen from the range \(2^{-15}, 2^{-14}, \ldots, 2^{15}\). The 50 hidden nodes are used in ELM, while for B-ELM, 10 hidden nodes are used. The average results of 50 trials of simulations with each parameter are obtained, and the best
TABLE XII
GENERALIZATION PERFORMANCE COMPARED BASED ON HELD-OUT TESTING. ALL RESULTS ARE AVERAGE ONCE OVER 20 RUNS (RANK: 1-BEST AND 5-WORST)

<table>
<thead>
<tr>
<th>Equation</th>
<th>PLM</th>
<th>K-linreg</th>
<th>PE-MCS</th>
<th>ME-MCS</th>
<th>Algebraic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>0.0054</td>
<td>0.0068</td>
<td>0.0071</td>
<td>0.0068</td>
</tr>
<tr>
<td></td>
<td>Std Dev</td>
<td>0.0013</td>
<td>0.0005</td>
<td>0.0003</td>
<td>0.0007</td>
</tr>
<tr>
<td></td>
<td>T-value</td>
<td>N/A</td>
<td>7.1074</td>
<td>9.0100</td>
<td>6.7048</td>
</tr>
<tr>
<td></td>
<td>Rank</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td>0.0125</td>
<td>0.0418</td>
<td>0.0502</td>
<td>0.0410</td>
</tr>
<tr>
<td>(a = 0.2)</td>
<td>Std Dev</td>
<td>0.0020</td>
<td>0.0018</td>
<td>0.0012</td>
<td>0.0017</td>
</tr>
<tr>
<td></td>
<td>T-value</td>
<td>N/A</td>
<td>76.9967</td>
<td>114.2649</td>
<td>86.4033</td>
</tr>
<tr>
<td></td>
<td>Rank</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td>0.0251</td>
<td>0.0470</td>
<td>0.0714</td>
<td>0.0447</td>
</tr>
<tr>
<td>(a = 0.5)</td>
<td>Std Dev</td>
<td>0.0019</td>
<td>0.0032</td>
<td>0.0028</td>
<td>0.0058</td>
</tr>
<tr>
<td></td>
<td>T-value</td>
<td>N/A</td>
<td>41.6106</td>
<td>96.7527</td>
<td>22.7080</td>
</tr>
<tr>
<td></td>
<td>Rank</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td>0.0110</td>
<td>0.0099</td>
<td>0.0075</td>
<td>0.0079</td>
</tr>
<tr>
<td>(a = 0.2)</td>
<td>Std Dev</td>
<td>0.0013</td>
<td>0.0003</td>
<td>0.0004</td>
<td>0.0008</td>
</tr>
<tr>
<td></td>
<td>T-value</td>
<td>N/A</td>
<td>-5.8300</td>
<td>-18.1956</td>
<td>-14.3605</td>
</tr>
<tr>
<td></td>
<td>Rank</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td>0.0314</td>
<td>0.0402</td>
<td>0.0384</td>
<td>0.0414</td>
</tr>
<tr>
<td>(a = 1)</td>
<td>Std Dev</td>
<td>0.0015</td>
<td>0.0025</td>
<td>0.0010</td>
<td>0.0027</td>
</tr>
<tr>
<td></td>
<td>T-value</td>
<td>N/A</td>
<td>21.3431</td>
<td>27.4563</td>
<td>22.8934</td>
</tr>
<tr>
<td></td>
<td>Rank</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td>0.0088</td>
<td>0.0031</td>
<td>0.0035</td>
<td>0.0043</td>
</tr>
<tr>
<td>(a = 0.5)</td>
<td>Std Dev</td>
<td>0.0008</td>
<td>0.0002</td>
<td>0.0004</td>
<td>0.0004</td>
</tr>
<tr>
<td></td>
<td>T-value</td>
<td>N/A</td>
<td>-48.8771</td>
<td>-41.2829</td>
<td>-35.5756</td>
</tr>
<tr>
<td></td>
<td>Rank</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td>0.0188</td>
<td>0.0131</td>
<td>0.0157</td>
<td>0.0153</td>
</tr>
<tr>
<td>(a = 2)</td>
<td>Std Dev</td>
<td>0.0031</td>
<td>0.0009</td>
<td>0.0014</td>
<td>0.0026</td>
</tr>
<tr>
<td></td>
<td>T-value</td>
<td>N/A</td>
<td>-8.1050</td>
<td>-2.2867</td>
<td>-2.6215</td>
</tr>
<tr>
<td></td>
<td>Rank</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td>0.0209</td>
<td>0.0315</td>
<td>0.0356</td>
<td>0.0386</td>
</tr>
<tr>
<td>(a = 0.5)</td>
<td>Std Dev</td>
<td>0.0029</td>
<td>0.0018</td>
<td>0.0038</td>
<td>0.0064</td>
</tr>
<tr>
<td></td>
<td>T-value</td>
<td>N/A</td>
<td>21.9598</td>
<td>21.7450</td>
<td>20.4192</td>
</tr>
<tr>
<td></td>
<td>Rank</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td>0.0799</td>
<td>0.1290</td>
<td>0.1311</td>
<td>0.1379</td>
</tr>
<tr>
<td>(a = 2)</td>
<td>Std Dev</td>
<td>0.0182</td>
<td>0.0114</td>
<td>0.0157</td>
<td>0.0148</td>
</tr>
<tr>
<td></td>
<td>T-value</td>
<td>N/A</td>
<td>16.1667</td>
<td>15.0623</td>
<td>17.4832</td>
</tr>
<tr>
<td></td>
<td>Rank</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td>0.0132</td>
<td>0.1155</td>
<td>0.4825</td>
<td>0.4331</td>
</tr>
<tr>
<td>(a = 0.2)</td>
<td>Std Dev</td>
<td>0.0010</td>
<td>0.0028</td>
<td>0.0120</td>
<td>0.0093</td>
</tr>
<tr>
<td></td>
<td>T-value</td>
<td>N/A</td>
<td>243.2957</td>
<td>275.5824</td>
<td>317.4327</td>
</tr>
<tr>
<td></td>
<td>Rank</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td>0.0438</td>
<td>0.1358</td>
<td>0.4769</td>
<td>0.4326</td>
</tr>
<tr>
<td>(a = 1)</td>
<td>Std Dev</td>
<td>0.0027</td>
<td>0.0044</td>
<td>0.0071</td>
<td>0.0063</td>
</tr>
<tr>
<td></td>
<td>T-value</td>
<td>N/A</td>
<td>126.0156</td>
<td>403.1673</td>
<td>401.1019</td>
</tr>
<tr>
<td></td>
<td>Rank</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Overall Ranking</td>
<td>1(1.91)</td>
<td>2(2.73)</td>
<td>3(3.91)</td>
<td>4(3.36)</td>
<td>3(3.00)</td>
</tr>
</tbody>
</table>

this section, the 3/5 of the data set are used to create the training set, then other 1/5 is used for the validation set to select parameter, and remaining 1/5 is used for the test set. To make all the tests are run under the same condition, in this section, the experiment steps (see Algorithm 2) are shown as follows:

1) For the PLM, there are two parameters (ε, L), but based on our analysis above, the generalization performance of PLM does not change whenever L changes (L > 2). Thus, in this test, we only consider the parameter ε and set p = ε (see Step 1).

2) Different from the PLM, the true submodel number in other compared methods must be known. In other compared methods, we set the submodel number parameter n equal to the true submodel number. Therefore, for the Algebraic and the K-linreg, there is only one parameter δ and we set p = δ. For the PE-MCS and the ME-MCS, we set p = 1 because there is no parameter here (see Step 1).

3) To test SARX systems, we should know the submodel index for each testing data. However, based on the definition of SARX, it is impossible to estimate the submodel
index for the reason that the model index in SARX is switching freely. Furthermore, all the methods in this section except the PLM have no predictive capability for PWA system, i.e., these methods cannot give the estimated submodel index for testing data. Therefore, in this test, all the testing data points from validation sets or testing sets are classified using true submodel index (see Step 3).

Table XII shows the performance comparison of K-linreg, ME-MCS, PE-MCS, Algebraic method, and PLM. When the $t$-value is $>1.645$ ($\nu = 40$), there is a significant difference between the two algorithms with a 95% confidence level. The mean cost values, standard deviation values, and $t$-value between the PLM and other compared methods are shown in Table XII. We see that most $t$-values in this table are $>1.645$ especially for high-noise data set. Therefore, the performance of the PLM is significantly better than that of other methods with a 95% confidence level.

VI. CONCLUSION

This paper proposes an efficient learning algorithm—the PLM-based on multi-SLFNs, which has also been rigorously proven in this paper. The proposed PLM has several interesting and significant features different from traditional methods.

1) Unlike other identification algorithms that only work for one type hybrid system, the proposed PLM method can learn and approximate any hybrid system, such as general PWA state space systems, general piecewise nonlinear systems, and switched linear/nonlinear systems. In particular, unlike other hybrid system identification algorithms only operate in linear subsystem hybrid systems, the proposed PLM method can approximate nonlinear submodel hybrid systems.

2) Unlike other hybrid system learning algorithms that require some a priori information, such as submodel order, submodel number, and so on, this proposed method can provide similar or better generalization performance than other methods without needing to know any a priori information.

3) Unlike other NN-based learning algorithms that only have universal approximation capability for continuous systems, the proposed multinetwork-based method has universal approximation capability for general nonlinear/linear hybrid systems. Thus, this paper extends the universal approximation capability of NN-based methods from continuous systems to hybrid systems.

This method can be used efficiently in many applications, such as video-shot segmentation, multi-instances learning, and image dimension reduction, which are currently under our investigation. In addition, compared with some identification methods, the PLM have higher computation complexity. How to significantly reduce the computation complexity of the PLM is worth investing.

ACKNOWLEDGMENT

The authors would like to thank the editor and anonymous reviewers for their invaluable suggestions, which have been incorporated to improve the quality of this paper dramatically. They would also like to thank Dr. S. Paoletti, Department of Information Engineering, University of Siena, Siena, Italy, for a demo and source code of MILP and BEA, Dr. F. Lauer, Department of Computer Science, University of Lorraine, Nancy, France, for source code of K-linreg, ME-MCS, and PE-MCS, Dr. N. Ozay, Department of Control and Dynamical Systems, California Institute of Technology, Pasadena, CA, USA, for source code of method [15], and Dr. G.-B. Huang, Nanyang Technological University, Singapore, for constructive suggestions.

REFERENCES


Yimin Yang (S’10–M’13) received the M.Sc. and Ph.D. degrees in electrical engineering from the College of Electrical and Information Engineering, Hunan University, Changsha, China, in 2009 and 2013, respectively. He is currently an Assistant Professor with the College of Electrical Engineering, Hunan University. His current research interests include neural networks, approximate dynamic programming, and industrial process control.

Yaonan Wang received the Ph.D. degree in electrical engineering from Hunan University, Changsha, China, in 1994. He was a Post-Doctoral Research Fellow with the Normal University of Defence Technology, Changsha, from 1994 to 1995. From 1998 to 2000, he was a Senior Humboldt Fellow in Germany, and, from 2001 to 2004, he was a Visiting Professor with the University of Bremen, Bremen, Germany. Since 1995, he has been a Professor with the College of Electrical and Information Engineering, Hunan University. His current research interests include intelligent control, robotics, and image processing.