A Semi-Supervised Learning Approach for Identification of Piecewise Affine Systems

Yingwei Du[1], Fangzhou Liu[1*], *Member, IEEE*, Jianbin Qiu[2], *Senior Member, IEEE*, and Martin Buss[1], *Fellow, IEEE*

Abstract

Piecewise affine (PWA) models are attractive frameworks that can represent various hybrid systems with local affine submodels and polyhedral regions due to their universal approximation properties. The PWA identification problem amounts to estimating both the submodel parameters and the polyhedral partitions from data. In this paper, we propose a novel approach to address the identification problem of PWA systems such that the number of submodels, parameters of submodels, and the polyhedral partitions are obtained. In particular, a cluster-based algorithm is designed to acquire the number of submodels, the initial labeled data set, and initial parameters corresponding to each submodel. Additionally, we develop a modified self-training support vector machine algorithm to simultaneously identify the hyperplanes and parameter of each submodel with the outputs of the cluster-based algorithm. The proposed algorithm is computationally efficient for region estimation and able to accomplish this task with only a small quantity of classified regression vectors. The effectiveness of the proposed identification approach is illustrated via simulation results.

Index Terms

PWA system; identification; cluster-based algorithm; modified self-training SVM.

I. Introduction

Piecewise affine (PWA) systems are switched affine systems with partitioned non-overlapping convex polyhedral input-state space partitions and state-dependent switching. Due to their simple structure and universal approximation properties, PWA systems have received drastic attention as attractive frameworks for approximating hybrid system [1]. The hybrid systems are ubiquitous in engineering and applied science, the control strategies of whom are widely researched [2]–[4]. Therefore, it is intuitive to build equivalent PWA systems to simplify the control of the hybrid systems. Moreover, prior to the analysis, verification, computation, and control of PWA systems, the identification is essential [5]–[7]. However, this problem is NP-hard in general [8] and suffers from the coupling between region estimation and submodel identification. Previous literature has documented various approaches to carry out the identification problem of PWA systems [9]–[14].

For the identification of PWA systems, piecewise affine autoregressive exogenous (PWARX) model, whose regression vector is defined as the collection of past input and output observations, plays a dominant role. The identification of PWARX model incorporates the estimation of both parameters of each submodel and polyhedral partitions. Thus the identification approaches developed in previous work generally consist of two stages, i.e., i) identifying the parameters of each affine submodel and ii) estimating polyhedral partitions of the input-state space [15]–[19]. For the first stage, various methods have been presented in previous works to identify the parameters of each affine submodel. In [15], a convex framework based on ℓ_1-regularization is proposed to accomplish the identification of submodel parameters. By introducing kernel function-based weighted least squares, a recursive identification algorithm is developed in [16] for parameter identification. The cluster-based algorithm proposed by [17] accomplishes the parameter estimation via employing a clustering method based on a Gaussian mixture model. In [18], the parameters of each submodel is computed through a clustering technique based on k-means. However, unlike the first stage, the SVM algorithm is dominantly employed to estimate the partitions of submodels in the papers mentioned above [15]–[19]. The reason is that the SVM algorithm is capable of estimating the hyperplanes precisely. However, effective and well-performed as SVM is regarding the identification tasks, it can be numerically inefficient. In SVM algorithm, all the data points are required to be labeled and classified before estimating the polyhedral partitions in these algorithms, which is generally expensive and time-consuming, especially for large data sets [20]. Besides, computing feasible partitions through the SVM algorithm may suffer high computational complexity [21] in this case. Taking into consideration these limitations of the SVM algorithm, recent literature propose various extensions [22]–[24], which, however, have not been applied in the PWA identification. Instead, the authors in [25] propose a linear multi-category discrimination method which is numerically efficient in the partitioning stage. This method, nonetheless, results in low accuracy while dealing with small training sets due to adopting the average stochastic

This work was supported in part by the Self-Planned Task of State Key Laboratory of Robotics and Systems of Harbin Institute of Technology (No. SKLRS201801A03).

[1]Y. Du, F. Liu and M. Buss are with the Chair of Automatic Control Engineering (LSR), Department of Electrical and Computer Engineering, Technical University of Munich, Theresienstr. 90, 80333, Munich, Germany; {yingwei.du, fangzhou.liu, mb}@tum.de

[2]J. Qiu is with the State Key Laboratory of Robotics and Systems, Harbin Institute of Technology, Harbin 150080, China, and also with the Research Institute of Intelligent Control and Systems, Harbin Institute of Technology, Harbin 150080, China; jbqiu@hit.edu.cn

gradient descent. Moreover, the large quantity of coefficients that required to be assigned and tuned in priority is also a drawback.

Another challenge in the PWARX identification is to specify a proper number of affine submodels to guarantee the overall accuracy. The trade-off between accuracy, generalization, and model complexity is necessary to be taken into account. Most of the previous works directly adopt the assumption that the number of affine submodels are given in advance [17], [18], [26]. Recently, various approaches are proposed to tackle this problem. A fuzzy clustering validation based algorithm was proposed in [27] to obtain the optimal submodel number. In [25], the optimal number is chosen by means of cross validation and the number of submodels corresponding to the largest best fit rate (BFR) is set to be the optimal. Specifically, the BFR is introduced as an index to assess model quality. In these methods, an algorithm to determine the optimal number of submodel needs to be specifically designed and implemented ahead of identification. As an alternative, [28] tackles this issue by observing the distribution of the data points and fixing the number manually, which is subjective and inapplicable for general identification problems.

Taking into account the limitations of the previous work, in this paper, we will propose a novel approach for the identification of PWA models. The identification approach consists of a cluster-based algorithm and a modified self-training SVM algorithm. The cluster-based algorithm is proposed to obtain the initial labeled data set, initial parameters of each submodel, and the number of submodels simultaneously. It provides a method for initialization and is generally applicable in other identification approaches for PWA systems which require initialization as well, e.g., Bayesian-based methods, clustering-based methods [9]. In the algorithm, we introduce two original concepts: *local cluster* and *sub local cluster* for splitting the data set into clusters. Based on these concepts, two cost functions are designed to accomplish the task. Particularly, one of them is utilized to estimate the number of submodel. In addition, the modified self-training SVM algorithm computes the parameters of submodels and estimates the polyhedral partitions with the outputs of cluster-based algorithm. By imposing a customized selection strategy, the modified self-training SVM algorithm is capable of estimating the polyhedral partitions precisely with partially-labeled data set. Moreover, the computational complexity of the modified self-training SVM algorithm is lower than the traditional SVM algorithm.

The remainder of the paper is organized as follows. In Section II and III, we introduce the necessary background knowledge and formulate the problem, respectively. Section IV describes the cluster-based algorithm and the modified self-training SVM algorithm for PWARX system identification. Numerical examples are reported in Section V to show the effectiveness of the proposed approach.

Notations: $\mathbb{R}$ and $\mathbb{N}_{>0}$ are the set of real numbers and the set of positive integers, respectively. The notation $\preceq$ stands for component-wise inequality. Given a vector $a \in \mathbb{R}^n$, $||a||$ and $||a||_\infty$ represent the Euclidean norm and the infinity norm of a, respectively. Var$(d_1, d_2, \dots d_n)$ denotes the variance of a sequence of scalars $d_1, d_2, \dots d_n$. For a set $\mathcal{S}$, $\#\mathcal{S}$ stands for its cardinality. For two vectors $x = [x_1, x_2, \dots, x_n]^\top$ and $y = [y_1, y_2, \dots, y_n]^\top$, $d(x, y)$ denotes the standardized Euclidean distance, i.e.,

$$d(x,y) = \sqrt{\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{s_i^2}}, \tag{1}$$

where s_i is the standard deviation of the x_i and y_i over the vectors.

II. Preliminaries

Before embarking on the identification problem, we briefly introduce background knowledge of SVM algorithm [29], which are fundamental for the novel modified self-training SVM algorithm proposed in this paper.

SVM is a supervised machine learning algorithm and widely used to solve classification problems. Consider a data set $\mathcal{D} = \{(\mathbf{x}_k, \ell_k)\}_{k=1}^N$ with cardinality $N \in \mathbb{N}_{>0}$, where each data point is a pair consisting of a vector $\mathbf{x}_k$ and a label ℓ_k. The fundamental task of SVM is to train a classifier (w, b) to separate the samples with labels, where $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$ are the weight and intercept of the classifier, respectively.

Equivalently, training an SVM classifier is to solve the following quadratic optimization problem.

$$\max_{0 \le \alpha_k \le C} \left(\min_{w,b} \ \frac{1}{2}||w||^2 + \sum_{k=1}^{N} \alpha_k (1 - \ell_k (w^\mathsf{T} \mathbf{x}_k + b)) \right), \tag{2}$$

where α_k is the Lagrange multiplier and C is the penalty parameter.

By Karush-Kuhn-Tucker (KKT) conditions [30], it requires that

$$\alpha_k [\ell_k (w^\mathsf{T} \mathbf{x}_k + b) - 1 + \xi_k] = 0 \text{ for all } k = 1, 2, \dots, N, \tag{3}$$

where $\xi_k \ge 0$ are the slack variables which measure the degree of misclassification. Since there holds $\ell_k (w^\mathsf{T} \mathbf{x}_k + b) \ge 1 - \xi_k$ for all $k = 1, 2, \dots, N$, it is straightforward that the data point with positive α_k implies $\ell_k (w^\mathsf{T} \mathbf{x}_k + b) \le 1$. In this regard, support vectors are defined as data points which correspond to non-zero α_k in [31]. Equivalently, we define the support vectors (SVs) as follows.

Definition 1. Consider the SVM classifier (w, b) trained with data set $\mathcal{D} = \{(\mathbf{x}_k, \ell_k)\}_{k=1}^N$, the support vectors are the data points satisfying

$$\ell_k(w^\mathsf{T}\mathbf{x}_k + b) \leq 1. \tag{4}$$

By definition 1, the SVs are the data points lie on or in the margins of the SVM classifier which are defined as $w^\mathsf{T}\mathbf{x}_k + b = \pm 1$. It follows that only a small number of the training data points end up as SVs and the majority lie outside of the classifier margins [24]. For a data set $\mathcal{D}$ for SVM training, we denote the set of support vectors as $\mathcal{D}^{SV}$. The data points in $\mathcal{D} \setminus \mathcal{D}^{SV}$ are defined as non-support vectors (non-SVs). In addition, the trained classifier is fully determined by SVs in SVM [32].

Another concept in SVM algorithm is the geometric margin of trained classifiers [33] which is defined as follows.

Definition 2. For the classifier (w, b) trained by data set $\mathcal{D} = \{(\mathbf{x}_k, \ell_k)\}_{k=1}^N$, the geometric margin between a certain data point $(\mathbf{x}_k, \ell_k)$ and the classifier is given by

$$\gamma(w, b, \mathbf{x}_k) = \frac{|w^\mathsf{T}\mathbf{x}_k + b|}{||w||}. \tag{5}$$

Although the classifier introduced above is designed for binary classification, the concepts are the same for the multi-class classification. The reason is that the multi-class classification problems are generally considered as a collection of several binary classification problems [34]. As an extension of SVM algorithm, the self-training SVM is proposed as a semi-supervised algorithm [35]. The algorithm can provide a satisfactory classifier with only a small amount of labeled data and a large amount of unlabeled data [36]. In self-training SVM processes, a classifier is firstly trained with a small amount of labeled data. This classifier is used to separate the unlabeled data set and assigns predicted labels on them. Then the unlabeled data points, which are selected with the highest confidence, are added incrementally into the labeled training set with their predicted labels. These procedures work recursively for a given number of times or until meeting some convergence criterion.

III. Problem Formulation

Consider a PWARX model defined as follows

$$y_k = \begin{cases} \theta_1^\mathsf{T}\begin{bmatrix} x_k \\ 1 \end{bmatrix} + \varepsilon_k & \text{if } x_k \in \mathcal{X}_1, \\ \quad\vdots & \quad\vdots \qquad k = 1, 2, \ldots, N \\ \theta_s^\mathsf{T}\begin{bmatrix} x_k \\ 1 \end{bmatrix} + \varepsilon_k & \text{if } x_k \in \mathcal{X}_s, \end{cases} \tag{6}$$

where $x_k \in \mathbb{R}^{n_x}$, $y_k \in \mathbb{R}$ and $\varepsilon_k \in \mathbb{R}$ are the regression vector, output, and noise at time k, respectively. $\{\theta_i\}_{i=1}^s$ is the parameter vectors that define the constituent submodels of model (6). $s \in \mathbb{N}_{>0}$ is the number of submodels. The regression vector is defined as:

$$x_k = [y_{k-1} \quad \ldots \quad y_{k-n_y} \quad u_{k-1}^\mathsf{T} \quad \ldots \quad u_{k-n_u}^\mathsf{T}]^\mathsf{T}, \tag{7}$$

where $u_k \in \mathbb{R}^m$ is the input vector at time k, n_y and n_u are the system orders. It follows that $n_x = n_y + mn_u$. For convenience, we denote $\varphi_k = [x_k^\mathsf{T} \quad 1]^\mathsf{T}$ as the extended regression vector at time k.

The bounded polyhedron $\mathcal{X}_i$ with $i = 1, \ldots, s$ is the partition of the ith submodel and is defined as

$$\mathcal{X}_i = \{x_k \in \mathbb{R}^{n_x} \mid \mathcal{H}_i \cdot \varphi_k \preceq 0\}, \tag{8}$$

where $\mathcal{H}_i \in \mathbb{R}^{\mu_i \times (n_x+1)}$ is the hyperplane matrix and $\mu_i \in \mathbb{N}_{>0}$ is the number of hyperplanes that bounds the corresponding partition. Evidently, there hold $\mathcal{X}_i \bigcap \mathcal{X}_j = \emptyset$, $\forall i \neq j$ and $\mathcal{X} = \bigcup_{i=1}^s \mathcal{X}_i$. Based on the aforementioned settings, we formulate the identification problem as follows.

Problem 1. Given the order of input n_u and the order of output n_y of a PWARX model and a set of data points $\mathcal{D} = \{(x_k, y_k)\}_{k=1}^N$ generated therefrom, estimate the number of submodels s, the parameter vectors $\{\theta_i\}_{i=1}^s$, and polyhedral partitions $\{\mathcal{X}_i\}_{i=1}^s$.

IV. The Identification Approach

In this section, we propose a novel identification approach, consisting of the cluster-based algorithm and the modified SVM algorithm to tackle Problem 1. The cluster-based algorithm is designed to acquire the number of submodels, the initial labeled data set, and initial parameters. Subsequently, the modified SVM algorithm estimates the hyperplanes and parameter of each submodel simultaneously with the outputs of the cluster-based algorithm.

A. Cluster-Based Algorithm

The cluster-based algorithm aims at obtaining initial labeled data set, initial parameters, and the number of submodels. In order to implement the cluster-based algorithm, it is necessary to introduce the concept of *local cluster*.

Definition 3 (Local Cluster)**.** A local cluster $\mathcal{L}$ of a center $(x, y) \in \mathcal{D}$ is built by collecting the data points (x_k, y_k) in the hypersphere of radius δ. i.e.,

$$\{(x_k, y_k) \mid d(x, x_k) < \delta, \forall (x_k, y_k) \in \mathcal{D}\} \tag{9}$$

Moreover, an LC is said to be pure, if it only possesses data points belonging to the same submodel. Otherwise, it is said to be a mixed LC.

Definition 3 is inspired by [18] where a similar concept, local data set, is introduced to select a fixed number of data points based on Euclidean norm. In this article, as an alternative, we adopt the standardized Euclidean distance in the definition of LC due to its advantages in obtaining pure LCs. This advantage is revealed through an illustrative example in Section V. Meanwhile, Definition 3 indicates that the LCs are composed of the neighboring points. This characteristic is also beneficial to obtain pure LCs according to the formulation of PWARX model.

Note that the LCs are extracted from the data set $\mathcal{D}$ iteratively and we denote the nth LCs as $\mathcal{L}_n$. Additionally, we denote each LC as $\mathcal{L}_n = \{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{c_n}$ where c_n is the number of data points in corresponding LC. At each iteration, an LC is extracted with respect to a randomly selected center and fixed radius δ. Hence, the data set $\mathcal{D}$ is updated as $\mathcal{D} \setminus \mathcal{L}_n$ from which we extract the next LC. This process terminates until $\mathcal{D} = \emptyset$. Clearly, the LCs satisfy $\bigcup \mathcal{L}_n = \mathcal{D}$ and $\mathcal{L}_n \bigcap \mathcal{L}_m = \emptyset, \forall n \neq m$. As the process is terminated, the data points from $\mathcal{D}$ are allocated into a group of LCs. To acquire the initial labeled data set from LCs, the concept of a set of sub local clusters (sub LCs) is introduced as follows.

Definition 4 (Sub LCs)**.** A set of sub LCs are s pure LCs with respect to each submodel in the PWARX model (6), i.e.,

$$\{\hat{\mathcal{L}}_i | \hat{\mathcal{L}}_i \subseteq \mathcal{X}_i, i = 1, 2, \ldots, s.\}. \tag{10}$$

The set of sub LCs plays a fundamental role in the overall identification approach. It is assigned to be the initial labeled data set based on which the modified self-training SVM algorithm can be implemented. Thus the value of δ is required to be properly selected such that a set of sub LCs does exist. Detailed discussion on the choice of δ is presented in Section IV-C. Note that there could exist several sets of sub LCs and the pure LCs in one set of sub LCs may not be unique. In the cluster-based algorithm, we do not aim at selecting all sets of sub LCs but focus on extracting any one of them. To this end, we design the following cost function to extract a set of sub LCs.

$$J(\mathcal{L}_n) = R(\mathcal{L}_n) + \alpha S(\mathcal{L}_n). \tag{11}$$

In cost function (11), $R(\mathcal{L}_n)$ is the residual component designed to distinguish the pure LCs. Meanwhile, $S(\mathcal{L}_n)$ is presented as the similarity component for discriminating pure LCs from different submodels to comprise a set of sub LCs. α is the regularization coefficient to adjust the weights of the two components.

By the cost function (11), we can discriminate a set of sub LCs from $\mathcal{L}_n$ by solving the optimization problem $\min J(\mathcal{L}_n)$ iteratively. At each iteration, every $J(\mathcal{L}_n)$ is calculated and the minimum value corresponds to one element of a set of sub LC. Note that the results hold true only when the iteration time is less or equal to s. Now, we introduce the components of $J(\mathcal{L}_n)$ in detail.

1) Residual Component $R(\mathcal{L}_n)$: In cost function (11), the residual component $R(\mathcal{L}_n)$ for distinguishing pure LCs is given by

$$R(\mathcal{L}_n) = ||\mathcal{Y}_n - \Phi_n \tilde{\theta}_n||_\infty \tag{12}$$

where

$$\begin{aligned} &\tilde{\theta}_n = (\Phi_n^\mathsf{T} \Phi_n)^{-1} \Phi_n^\mathsf{T} \mathcal{Y}_n, \\ &\Phi_n = \begin{bmatrix} \tilde{x}_1 & \tilde{x}_2 & \ldots & \tilde{x}_{c_n} \\ 1 & 1 & \ldots & 1 \end{bmatrix}^\mathsf{T}, \quad \mathcal{Y}_n = [\tilde{y}_1 \quad \tilde{y}_2 \ \ldots \ \tilde{y}_{c_n}]^\mathsf{T}. \end{aligned} \tag{13}$$

Here, the cost function stands for the maximum absolute value of residuals for each data point in $\mathcal{L}_n$. We elaborate the meaning of $R(\mathcal{L}_n)$ from the perspective of *outliers*. An outlier in a set of data is defined to be an observation which appears to be inconsistent with the remainder of that set of data [37]. Clearly, the data points in pure LC is generated by a linear model without outliers. Conversely, the mixed LC is a data set with some outliers, which are the data points from other submodels. Therefore, the residual used to detect outliers [37] can be imposed to distinguish pure LCs.

If a linear regression model is built with a data set without outliers, the residuals are assumed to have a normal distribution with a mean 0 [38]. Apparently, an LC is more likely to be pure if every residual has an absolute value closer to 0. Accordingly, we build a linear regression model for every LC and compute the corresponding value of $R(\mathcal{L}_n)$. If there is no existence of outliers in an LC, the corresponding $R(\mathcal{L}_n)$ is close to 0. Any outlier in an LC causes a greater value of $R(\mathcal{L}_n)$. Therefore, the cost function $R(\mathcal{L}_n)$ can be utilized as a criterion to distinguish pure LCs.

2) Similarity Component $S(\mathcal{L}_n)$: The similarity component $S(\mathcal{L}_n)$ is designed as follows to discriminate pure LCs from different submodels.

$$S(\mathcal{L}_n) = \begin{cases} 0 & \text{if } \hat{c} = 0, \\ 1 - \tanh(\min\limits_{i=1,\dots,\hat{c}}(||\tilde{\theta}_n - \hat{\theta}_i||)) & \text{if } \hat{c} > 0, \end{cases} \tag{14}$$

where $\hat{c}$ and $\hat{\theta}_i$ are respectively the number and the parameter of sub LCs which have been discriminated. Clearly, $S(\mathcal{L}_n)$ is updated with respect to $\hat{c}$. While discriminating the first sub LC ($\hat{c} = 0$), $S(\mathcal{L}_n)$ is dropped from the cost function J due to the fact that the first extracted pure LC must be a sub LC. Throughout the later procedure, $S(\mathcal{L}_n)$ is activated to obtain the rest members of a set of sub LCs. At each iteration, if one $\mathcal{L}_n$ and either of sub LC from $\{\hat{\mathcal{L}}_i\}_{i=1}^{\hat{c}}$ are from identical submodel, the corresponding $S(\mathcal{L}_n)$ value is close to 1. On the contrary, one $\mathcal{L}_n$ will correspond to a small $S(\mathcal{L}_n)$ if this LC is from a distinct submodel with any sub LCs in $\{\hat{\mathcal{L}}_i\}_{i=1}^{\hat{c}}$. Consequently, we can employ $S(\mathcal{L}_n)$ as a criterion to discriminate LCs from different submodels. Note that the scale of the parameters can be disparate in different local clusters. Thus $\tanh$ is adopted to normalize and re-scale the discrepancies. Moreover, the adoption of $\tanh$ ensures the cost function to be non-negative.

Remark 1. During the implementation of the cluster-based algorithm, the residual component $R(\mathcal{L}_n)$ of each LC is invariant at each iteration. Therefore, we can compute $R(\mathcal{L}_n)$ of each LC in advance and only update $J(\mathcal{L}_n)$ with new $S(\mathcal{L}_n)$ at each iteration.

Remark 2. In the cluster-based algorithm, the center of each LC, i.e., (x_n, y_n), is selected randomly from the data set $\mathcal{D}$. Although the choice of (x_n, y_n) affects the number of mixed and pure local clusters, the approach will still be well-performed as long as there exists at least one set of sub LCs in all local clusters. Even if all LCs are mixed due to the unfortunate choices of (x_n, y_n), we can avoid this situation effortlessly by adjusting σ of LCs. Therefore, the random choices of (x_n, y_n) does not lead to poor performance of the proposed approach.

Apart from the sub LCs, the number of the submodels s can be obtained by recording the value of $\min J(\mathcal{L}_n)$ at each iteration. We denote the value of $\min J(\mathcal{L}_n)$ at the j-th iteration as J_i. Consider the case when a set of s sub LCs has been obtained after the s-th iteration. If we continue to solve the minimization problem, the value of J_{s+1} increases immediately. No matter the $(s+1)$th LC is either a mixed one or from a repetitive submodel, leading to an abrupt increment of $R(\mathcal{L}_n)$ or $S(\mathcal{L}_n)$. Then, the cost function value of later LC fluctuates in a relative high value. Thus, the submodel number can be estimated with the following cost function.

$$s = \arg \min_{i=2,\dots,m-1} (\text{Var}(J_1, \dots, J_{i-1}) + \text{Var}(J_{i+1}, \dots J_m)) \tag{15}$$

where m is the iteration times and assigned as the upper bound of the number of submodels. The upper bound is given in advance and limits the number of submodels for the identification. A higher number of submodels s results in a more accurate description of the PWA model. However, this may cause poor generalization to the data not used in the identification phase and increase the computational burden of the proposed algorithm. Therefore, we impose an upper bound m and estimate the optimal submodel number inside the boundary.

Following the estimation of s, a set of sub LCs $\hat{\mathcal{L}}_i$ and the corresponding parameter $\hat{\theta}_i$ are picked up. Later, we assign the label ℓ_i to every data points in $\hat{\mathcal{L}}_i$ for $i = 1 \dots s$. Consequently, the initial labeled data set is obtained by assembling the labelled sub LCs. It is denoted as $\mathcal{D}_l = \{(\mathbf{x}_k, \ell_k)\}_{k=1}^{\hat{n}}$ where $\mathbf{x}_k = \begin{bmatrix} x_k^\mathsf{T} & y_k \end{bmatrix}^\mathsf{T}$ and $\hat{n}$ is the number of labeled data points.

To conclude, the number of submodels s, the initial parameters $\{\hat{\theta}_i\}_{i=1}^{s}$, and the initial labeled data set $\mathcal{D}_l$ are procured through the cluster-based algorithm. The algorithm is summarized in Algorithm 1.

B. Modified Self-Training SVM Algorithm

Armed with initial conditions obtained by the cluster-based algorithm, in this section we introduce the modified self-training SVM algorithm for estimating the hyperplanes and parameter vector of each submodel.

The modified self-training SVM algorithm is described in Algorithm 2. The initial training set $\mathcal{D}_T$ is a collection of the center corresponding to every sub LC in $\mathcal{D}_l$. At each iteration, the classifiers are trained with $\mathcal{D}_T$. Note that the kernel of the SVM for training is linear due to the definition of polyhedral region. Then $\mathcal{D}_T$ and $\mathcal{D}_l$ are updated and $\{\hat{\theta}_i\}_{i=1}^{s}$ are re-calculated with the updated $\mathcal{D}_l$. With the iterative procedures, the trained classifiers gradually converge to the optimal hyperplanes. The proof of the convergence can be referred to in the appendix.

We then provide the criterion for terminating the iteration as follows. For a given tolerance σ, if there holds

$$\max_{i=1,2,\dots,q} \left(f(w_i^{(j)}, \xi_i^{(j)}) - f(w_i^{(j-1)}, \xi_i^{(j-1)}) \right) < \sigma, \tag{16}$$

where

$$f(w_i, \xi_i) = \frac{1}{2}||w_i||^2 + C \sum_{k=1}^{\#\mathcal{D}_T} \xi_{k,i}, \tag{17}$$

Algorithm 1 Cluster-based algorithm

Input:

Data set $\mathcal{D} = \{(x_k, y_k)\}_{k=1}^{N}$, coefficients δ, upper bound m

Output:

Number of submodels s, initial labeled data set $\mathcal{D}_l$ and initial parameters $\{\hat{\theta}_i\}_{i=1}^{s}$

1: Set $n = 1$, $\hat{c} = 0$
2: **repeat**
3: Select a data point randomly from $\mathcal{D}$ and denoted as (x_n, y_n)
4: Build $\mathcal{L}_n$ of (x_n, y_n) by Definition 3.
5: $\mathcal{D} \leftarrow \mathcal{D} \setminus \mathcal{L}_n$
6: $n \leftarrow n + 1$
7: **until** $\mathcal{D} = \emptyset$
8: Calculate the parameter $\tilde{\theta}_n$ of each $\mathcal{L}_n$ with the equation (13)
9: Calculate $R(\mathcal{L}_n)$ of each $\mathcal{L}_n$ with the equation (12)
10: **for** $i = 1$ to m **do**
11: Calculate $S(\mathcal{L}_n)$ of each $\mathcal{L}_n$ with the equation (14)
12: Calculate $J(\mathcal{L}_n)$ of each $\mathcal{L}_n$ with the equation (11)
13: $p \leftarrow \arg\min_{n} J(\mathcal{L}_n)$
14: $J_i \leftarrow J(\mathcal{L}_p)$
15: $\hat{\theta}_i \leftarrow \tilde{\theta}_p$, $\hat{\mathcal{L}}_i \leftarrow \mathcal{L}_p$
16: $\hat{c} \leftarrow i$
17: Update $S(\mathcal{L}_n)$ with $\hat{\theta}_i$ and $\hat{c}$
18: **end for**
19: Calculate the number of submodels s with the equation (15)
20: Assign labels $\ell_1 \dots \ell_s$ to every data points in $\hat{\mathcal{L}}_1 \dots \hat{\mathcal{L}}_s$ and compose the labeled data set $\mathcal{D}_l$
21: **return** s, $\mathcal{D}_l$, $\{\hat{\theta}_i\}_{i=1}^{s}$

the iteration is terminated. In the equation (16), j indicates the index of iteration, $\xi_{k,i}$ indicates ξ of every sample in $\mathcal{D}_T$ w.r.t the classifier (w_i, b_i). Afterwards, the classifiers are trained for n_{re} times without discarding process (Step 19), where n_{re} is assigned to a small scalar by users.

In Algorithm 2, we design the strategy to update the training set based on the fact that only the SVs contribute to the optimal hyperplane. Consequently, data points which are not SVs could be eliminated without affecting the trained hyperplanes. To sum up, for a data set $\mathcal{D}$, the same optimal hyperplanes can be obtained with a small training set $\mathcal{D}_T$ as long as the SVs of $\mathcal{D}$ are involved in $\mathcal{D}_T$. In other words, Algorithm 2 converges to optimal hyperplanes in this circumstance. Therefore, our update strategy is designed to collect the samples which are more likely to be SVs into $\mathcal{D}_T$. The update strategy is composed of the union process and discard process.

As pointed out in Section II, the geometric margins between SVs and the classifiers of $\mathcal{D}$ are smaller than any other samples in $\mathcal{D}$. Thus, the data points whose geometric margin corresponding to either classifier is the minimum are collected as $\mathcal{M}$ in Algorithm 2. Note that these data points are selected from $\mathcal{D} \setminus \mathcal{D}_T$. Clearly, the data points collected based on the equation (20) is more likely to be the SVs of $\mathcal{D}$.

The design of the update strategy also improves the convergence speed of Algorithm 2. According to the proof in Appendix, the data points from $\mathcal{M}$ formulate the new constraints in the optimization problem of SVM algorithm. Note that the trained classifiers remain constant if the new constraints have been fulfilled. Clearly, this situation happens when all data points in $\mathcal{M}$ are correctly classified and their geometric margins w.r.t the current classifiers are greater than 1. Therefore, the update strategy based on minimum geometric margins ensures that the classifiers do not remain constant at each iteration in early stage. In summary, by avoiding the classifiers from being constant, the update strategy improves the convergence rate of the proposed algorithm.

The update strategy of $\mathcal{D}_T$ also includes the procedure of discarding samples from D_T. The non-SVs of D_T are discarded after training the classifiers at each iteration. This procedure is designed to increase the computational efficiency of the algorithm. Moreover, the discarding procedure will not affect the convergence of the modified self-training algorithm, which is described in the appendix.

In the modified self-training SVM algorithm, the parameters of submodels are updated with the labeled data set $\mathcal{D}_l$ instead of using the training set $\mathcal{D}_T$ at each iteration. The reason is the parameters calculated with more samples are more plausible in the regression analysis. Compare to the training set, the labeled data set is updated without the procedure of discarding and contains more samples for computing the parameters.

For the termination criterion (16), the choice of σ influences the efficiency and performance of our algorithm, which is

presented in section IV-C. In the appendix, we analyze the convergence of modified self-training SVM algorithm and have the following theorem which is inspired by [35].

Theorem 1. For $f(w_i, \xi_i)$ defined in (17), we have

$$f(w_i^{(j)}, \xi_i^{(j)}) \geq f(w_i^{(j-1)}, \xi_i^{(j-1)}) \tag{18}$$

Hence, $f(w_i, \xi_i)$ can be employed as the criterion to stop the iteration.

Remark 3. Comparing with the self-training SVM, the proposed modified self-training SVM designs a novel selection strategy for collecting the unlabeled data points. In our algorithm, the unlabeled data which are collected to augment the training set are corresponding to the minimum geometric margin instead of the highest confidence. Moreover, the predicted labels are determined by the estimated parameter of submodels in our algorithm rather than the classifiers trained at last iteration in the original self-training. Clearly, the predicted labels in our algorithm is more plausible in the PWA system identification. Thus, the proposed modified self-training SVM is specifically designed for the PWA identification and adopts the features of PWA systems sufficiently.

Remark 4. By Theorem 1, the termination criterion $f(w_i, \xi_i)$ may converge to a local minimum. The update strategy only guarantees that the criterion will not converge to a local minimum in the early stage. Consequently, we design the retraining step (step 19) in the algorithm to avoid local minimum.

The estimated hyperplanes $\{(w_i^{(j)}, b_i^{(j)})\}_{i=1}^q$ and the estimated submodel parameters $\{\theta_i\}_{i=1}^s$ can be computed with the Algorithm 2 . Thus the identification of PWARX system described in Problem 1 is achieved by implementing the proposed cluster-based algorithm and modified self-training SVM algorithm.

Remark 5. The modified self-training SVM algorithm is designed for linear separable and inseparable case by adjusting the penalty parameter C. The submodel of PWA model is generally linear separable as stated in the identification of PWA model. Therefore, we generally choose a relative large values of C in the algorithm corresponding to assigning a higher penalty to errors [31].

Remark 6. The modified self-training SVM algorithm includes calculating the geometric margins, computing submodel parameters with $\mathcal{D}_l$, and training the classifiers with $\mathcal{D}_T$ at each iteration. The computational complexity of the algorithm can be calculated as follows.

The complexity of calculating the geometric margins and submodel parameter is $O(r \cdot N) + O(r \cdot N_{D_l})$, where r is the number of iteration, N is the number of samples in $\mathcal{D}$, $N_{\mathcal{D}_l}$ is the number of samples in $\mathcal{D}_l$. The complexity of the SVM training is $O\left([r(n_{sv} + s)]^3\right)$, where n_{sv} is the number of support vectors at each iteration, s is the number of submodels. Therefore, the total computational complexity of modified self-training SVM is

$$O(r \cdot N) + O(r \cdot N_{D_l}) + O\left([r(n_{sv} + s)]^3\right) \tag{19}$$

Above complexity grows linearly with respect to the number of iterations r. The number of iteration $r \ll N$ in the algorithm. The scale of r is presented in the section V. Moreover, the other coefficients in the (19) also fulfill $n_{sv}, s, N_{D_l} \ll N$ as well. Therefore, it is obvious that (19) is much smaller than the computational complexity of a normal SVM $O(N^3)$.

C. Tuning the parameters δ, σ, and α

In Definition 3, δ is given in advance to define LC. Consider the LC extraction procedure in cluster-based algorithm, a lower δ implies more LCs and further, more pure LCs. However, when the noise level is not negligible, a lower δ implies fewer data points in the regression leading to poor initial parameter estimation. This causes inaccuracy in the identification procedure. On the contrary, a higher δ provides more data for regression in every LC and improves the accuracy of parameters estimations. Whereas, a higher δ denotes fewer pure LCs and fewer number of iterations for extracting sub LCs. Therefore, the choice of the parameter δ is a trade-off between the number of pure LCs and the accuracy of regression. Before extracting LCs, the data points are normalized to adjust them to a notionally common scale. This strategy ensures the selection of δ is not determined by the PWA models. As a result, it is reasonable to choose a relatively large δ since it increases the credibility of estimated parameters and reduces the number of iterations as well. The reduction of pure LCs caused by the choice will not influence the algorithm performance as long as at least one set of sub LCs still exist. Nevertheless, too large δ that vanishes all the pure LCs is apparently unacceptable.

The parameter σ is the terminal condition of our algorithm. Obviously, a smaller σ leads to more iterations before converge and brings a higher BFR. However, the rise of $f(w, \xi)$ is decided by the newly joined samples and not continuous. This characteristics illustrates there is a lower bound for σ, The accuracy of the estimation will not getting better after σ exceeds the boundary.

Consider the parameter α, which is a regularization coefficient to adjust the weights of the residual component and the similarity component in the cost function (11). Too small α leads to the situation that the LCs from the same submodels are

Algorithm 2 Modified self-training SVM algorithm

Input: Initial labeled data set $\mathcal{D}_l$, initial parameters $\{\hat{\theta}_i\}_{i=1}^s$, tolerance σ, Penalty coefficient C.
Output: Hyperplanes group $\mathcal{G}$ and estimated submodel parameters $\{\theta_i\}_{i=1}^s$

1: Let $j = 0$
2: Extract initial training set $\mathcal{D}_T$ from $\mathcal{D}_l$
3: **repeat**
4: Train a classifier group $\mathcal{G}^{(j)} = \{(w_i^{(j)}, b_i^{(j)})\}_{i=1}^q$ with training data set $\mathcal{D}_T$
5: Let $\mathcal{D}_T \leftarrow \mathcal{D}_T^{SV}$
6: Let

$$\mathcal{M} = \bigcup_{i=1}^{q} \arg \min_{\mathbf{x} \in \mathcal{D} \setminus \mathcal{D}_T} \gamma(w_i^{(j)}, b_i^{(j)}, \mathbf{x}) \tag{20}$$

7: Denote $\mathcal{M} = \{(x'_k, y'_k)\}_{k=1}^q$
8: **for** $k = 1$ to q **do**
9: Let

$$\rho = \arg \min_{i=1,\ldots,s} ||y'_k - \hat{\theta}_i x'_k|| \tag{21}$$

10: $\ell'_k \leftarrow \ell_\rho$
11: **end for**
12: Denote $\mathcal{M}_\ell = \{(\mathbf{x}'_k, \ell'_k)\}_{k=1}^q$ where $\mathbf{x}'_k = [x'^{\mathsf{T}}_k \quad y'_k]^{\mathsf{T}}$
13: $\mathcal{D}_T \leftarrow \mathcal{D}_T^{SV} \cup \mathcal{M}_\ell$
14: $\mathcal{D}_l \leftarrow \mathcal{D}_l \cup \mathcal{M}_\ell$
15: Recomputed $\{\hat{\theta}_i\}_{i=1}^s$ with $\mathcal{D}_l$ and the equation (13)
16: $j \leftarrow j + 1$
17: **until** $\max_{i=1,2,\ldots,q} \left(f(w_i^{(j)}, \xi_i^{(j)}) - f(w_i^{(j-1)}, \xi_i^{(j-1)}) \right) < \sigma$
18: **for** j to $j + n_{re}$ **do**
19: Step 4 and Step 6-18
20: **end for**
21: $\mathcal{G} = \{(w_i^{(j)}, b_i^{(j)})\}_{i=1}^q$, $\{\theta_i\}_{i=1}^s = \{\hat{\theta}_i\}_{i=1}^s$
22: **return** Hyperplane group $\mathcal{G}$ and estimated submodel parameters $\{\theta_i\}_{i=1}^s$

involved in the set of sub LCs. Conversely, mixed LCs can be collected into the set of sub LCs while choosing a too large α. Therefore, When a validation data set is available, a sensible idea would be to tune α with cross-validation techniques.

V. Simulation Results

A. Comparison of LC definitions

The main challenge of the cluster-based algorithm is to extract sub LCs from the data set. In the definition, we adopt the standardized Euclidean distance to define LCs instead of the Euclidean distance mentioned in [18]. Both of the algorithms aim at obtaining LCs but our algorithm concentrates more on the pure LCs. Utilizing the Euclidean distance may cause no existence of pure LCs in some circumstances. To illustrate this drawback, a PWARX system is given as an instance.

$$y_k = \begin{cases} [50 \;\; 0.2 \;\; 0.1]\varphi_k \;\text{ if } x_k \in \mathcal{X}_1, \\ [60 \;\; 0.1 \;\; 0.3]\varphi_k \;\text{ if } x_k \in \mathcal{X}_2, \end{cases} \tag{22}$$

where $\varphi_k = [x_k^{\mathsf{T}} \; 1]^{\mathsf{T}}$, $x_k = [y_{k-1} \; u_{k-1}]^{\mathsf{T}}$ and

$$\mathcal{X}_1 = \{x \in \mathbb{R}^2 : [0 \quad 1 \quad 0]\varphi_k < 0\},$$
$$\mathcal{X}_2 = \{x \in \mathbb{R}^2 : [0 \; -1 \; 0]\varphi_k \leq 0\}.$$

In this simulation, the input u_k is distributed uniformly in the interval $[-0.5, 0.5]$ and the initial regression vector is $x_0 = [y_0, u_0]^{\mathsf{T}} = [0, 0]^{\mathsf{T}}$. The regression vectors generated by the system (22) is shown in Fig 1 where the data points with the same color belong to an LC. As is presented in Fig 1(a), all LCs are mixed LCs if the LC is defined with Euclidean distance which is sensitive to the scaling of vectors. Obviously, this circumstance makes the cluster-based algorithm infeasible. In contrast, using the standardized Euclidean distance avoids this drawback by introducing normalization. The LCs defined with standard Euclidean distance are illustrated in Fig. 1(b) and six of them are pure LCs. Clearly, our definition of LC is more suitable for our algorithm.

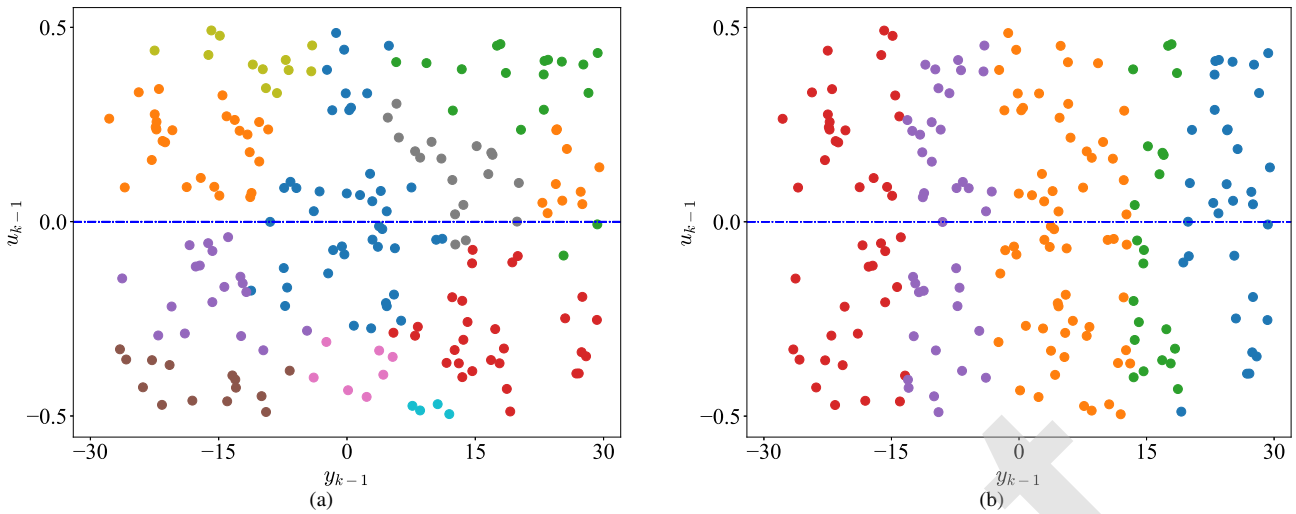


Fig. 1. The generated regression vectors (cross points) and the hyperplane (gray line) of the system (22). (a) Extracted LCs (data points with same color) by adopting standard Euclidean distance in the definition. (b) Extracted LCs (data points with same color) by adopting Euclidean distance in the definition.

B. Identification of PWA model

In this section, a series of simulations are conducted to illustrate the effectiveness of the proposed identification approach. We consider the following single-input-single-output (SISO) PWARX model composed of three submodels, i.e, $s = 3$, with the orders $n_a = n_b = 1$.

$$y_k = \begin{cases} [-0.4 \quad 1 \quad 1.5]\,\varphi_k + \varepsilon_k & \text{if } x_k \in \mathcal{X}_1, \\ [0.5 \quad -1 \quad -0.5]\,\varphi_k + \varepsilon_k & \text{if } x_k \in \mathcal{X}_2, \\ [-0.3 \quad 0.5 \quad -1.7]\,\varphi_k + \varepsilon_k & \text{if } x_k \in \mathcal{X}_3 \end{cases} \tag{23}$$

where $\varphi_k = [x_k^{\mathsf{T}} \ 1]^{\mathsf{T}}$, $x_k = [y_{k-1} \ u_{k-1}]^{\mathsf{T}}$ and

$$\mathcal{X}_1 = \{x \in \mathbb{R}^2 : \begin{bmatrix} 4 & -1 & 10 \end{bmatrix} \varphi_k < 0\},$$
$$\mathcal{X}_2 = \{x \in \mathbb{R}^2 : \begin{bmatrix} -4 & 1 & -10 \\ 5 & 1 & -6 \end{bmatrix} \varphi_k \leq 0\},$$
$$\mathcal{X}_3 = \{x \in \mathbb{R}^2 : \begin{bmatrix} -5 & -1 & 6 \end{bmatrix} \varphi_k < 0\}.$$

The PWARX model is typical and widely utilized for simulation in the existing papers [1], [17], [27], [39]. The input u_k is distributed randomly on the interval $[-2, 2]$ with $N = 1000$. The output y_k of the system is corrupted by an additive zero-mean Gaussian noise ε_k with the variance 0.12. To qualify the effect of noise, we impose the Signal-to-Noise Ratio (SNR) as follows

$$\text{SNR} = 10 \log \frac{\sum_{k=1}^{N} (y_k - \varepsilon_k)^2}{\sum_{k=1}^{N} \varepsilon_k^2}. \tag{24}$$

By implementing (24), the SNR of the white noise ε_k is 25dB in the simulation. Under aforementioned configurations, 1000 data points $\{(x_k, y_k)\}_{k=1}^{1000}$ conforming to the model (23) are generated and shown in Fig. 2 in the regression vector space. Then the proposed identification approach is implemented with the coefficients $\delta = 0.1$, $C = 1000$, $\alpha = 2$, $n_{re} = 6$, and $\sigma = 10^{-3}$. By running the cluster-based algorithm, the number of submodels is estimated as $s = 3$. In addition, a set of sub LCs is extracted, which is shown as stars in Fig. 2. The set D_T for classifier training is also shown as pentagons in the same figure. Apparently, all the sub LCs in this set satisfy Definition 4. Subsequently, the modified self-training SVM converges after 24 iterations and only 24.2% data points are classified and labeled in total. Compared with previous works, our algorithm is capable of estimating the hyperplanes precisely with a small fraction of labeled data points.

Based on the results obtained by the cluster-based algorithm, we then show the identification performance of the modified self-training SVM algorithm. As is presented in Fig. 3, the blue and red dashed lines are the estimated and the true hyperplanes, respectively. In detail, we plot the estimated hyperplanes with respect to the iteration times in Fig. 4. Note that w_i of the hyperplane is a vector which is donated by $w_i = \begin{bmatrix} w_i^1 & w_i^2 \end{bmatrix}^{\mathsf{T}}$ in the simulation. In Fig. 4, the entries of each hyperplane are plotted with different colors and styles. As is illustrated in Fig. 4, the estimated hyperplanes converge to the true hyperplanes $\begin{bmatrix} -4 & 1 & -10 \end{bmatrix}^{\mathsf{T}}$ and $\begin{bmatrix} 5 & 1 & -6 \end{bmatrix}^{\mathsf{T}}$ after 24 iterations. Apart from the hyperplanes, as is presented in Table I, the estimated parameters of each submodel are close to the true ones. Thus the simulation results indicates the submodel parameters and

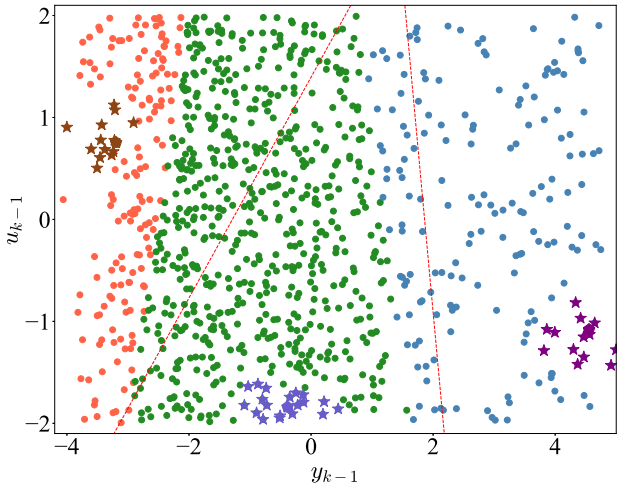


Fig. 2. Regression vectors (red, blue, green) generated by system (23), whose color represents the corresponding submodel. The sub LCs $\hat{\mathcal{L}}_i$ (pentagon points with different colors) extracted with cluster-based algorithm and hyperplanes (dashed red line) trained with initial data set $\mathcal{D}_T$.

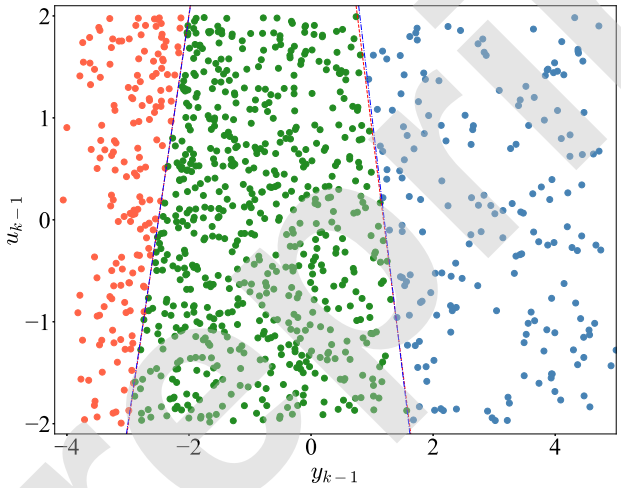


Fig. 3. Regression vectors (red, blue, green) generated by system (23), whose color represents the corresponding submodel. The true hyperplanes (blue dashed line) and the estimated hyperplanes (red dashed line) of system (23). By implementing Algorithm 2, the hyperplanes update at each iteration and eventually converge to the estimated hyperplanes.

TABLE I
TRUE (θ_i) AND ESTIMATED ($\hat{\theta}_i$) PARAMETER VECTORS IN THE SIMULATION

θ_1	$\hat{\theta}_1$	θ_2	$\hat{\theta}_2$	θ_3	$\hat{\theta}_3$
-0.4	-0.3890	0.5	0.5036	-0.3	-0.2816
1	0.9711	-1	-1.0102	0.5	0.4945
1.5	1.5801	-0.5	-0.4766	-1.7	-1.7084

hyperplanes are well-estimated. In addition, the proposed algorithm is more efficient than traditional SVM since less data points are employed.

We evaluate the performance of the proposed approach by the best fit rate (BFR) [40] which is defined as follows

$$\text{BFR} = \max\left\{1 - \frac{||y - \hat{y}||}{||y - \bar{y}\mathbf{1})||}, 0\right\} \times 100\%, \tag{25}$$

where $y \in \mathbb{R}^N$ is the true output sequence, $\hat{y} \in \mathbb{R}^N$ is the estimated output sequence. Additionally, $\bar{y} \in \mathbb{R}$ is the mean of the true output sequence and $\mathbf{1}$ is an all-ones vector with suitable dimension.

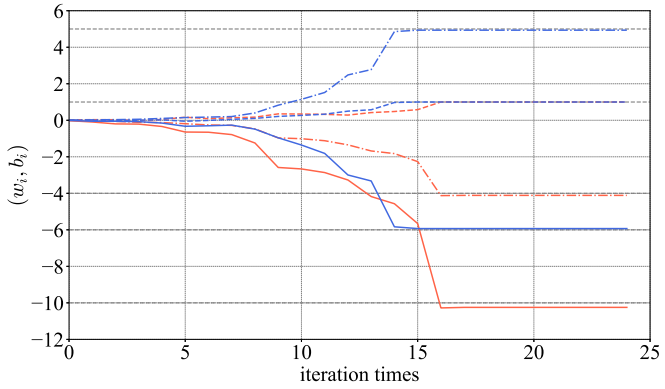


Fig. 4. The hyperplanes (w_1, b_1) and (w_2, b_2) with respect to the iteration times while implementing Algorithm 2. The red lines are the values of w_1^1, w_1^2, b_1 (dash-dotted line, dashed line, and solid line), respectively. The blue lines are the value of w_2^1, w_2^2, and b_2 (dash-dotted line, dashed line, and solid line), respectively.

A validation data set containing 200 data points is generated in the same conditions as (23). Fig. 5 depicts the true output y and the estimated output $\hat{y}$, as well as their error $y - \hat{y}$. For the sake of vision, 100 data points are plotted. The BFR value of the estimated PWARX system is 92.289%. The value implies the accuracy of the proposed approach is quite satisfying.

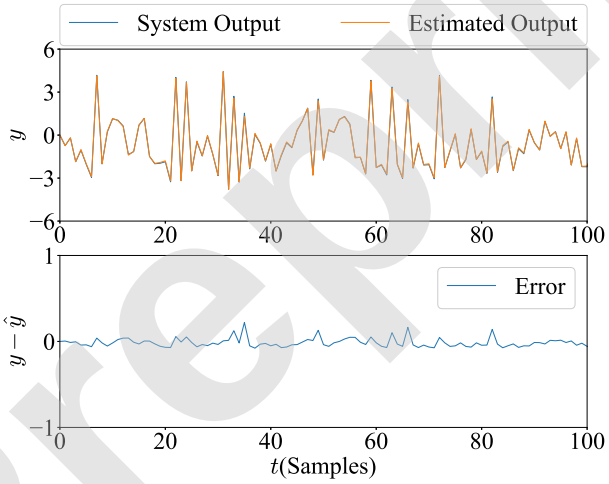


Fig. 5. System output y (blue line) and estimated output $\hat{y}$ (red line) of the validation data set, as well as their error $y - \hat{y}$.

The total amount of training data N_T and the number of iterations r are crucial factors to evaluate the computational complexity of our algorithm.The total amount of training data is calculated by summing the cardinality of $\mathcal{D}_T$ at each iteration. Therefore, we record them with different σ, N and PWA model (from [1] and [18]) in Table II. The percentage of N_T/N is around 20% while $N = 1000$. The percentage is getting much smaller with respect to the increment of N. While $N = 50000$, the percentage is around 4%. The result shows that fewer points are used in the training process compared with traditional SVM. Therefore, the modified self-training SVM is more efficient than traditional SVM, especially for large data set. The scale of iteration times of two different PWA models are similar to each other, showing the efficiency of modified self-training is not highly related to the PWA model type. According to our simulations, the convergent speed is related to the choice of σ. Generally, the algorithm convergence very fast (in 40 iterations). In addition, a smaller tolerance σ requires more iterations to converge. The values of BFR are around 90% in various conditions, showing a satisfying performance of our algorithm even for a small training set.

The coefficients δ, σ and α influence the performance of our algorithm greatly. Therefore, we analysis the sensitivity of δ, σ and α with respect to BFR for the PWA system (23) through the same validation set used in Fig. 5. Figs. 6, 7, and 8 illustrate the values of BFR with respect to different values of δ, σ, and α, respectively. Note that, the BFR rates are relatively high with the coefficient $0.10 \leq \delta \leq 0.15$, indicating a satisfying performance of our algorithm. For $\sigma \leq 10^{-3}$, the final estimate is fairly insensitive to the tolerance. Moreover, a wild range of $1.0 \leq \alpha \leq 10.0$ is allowed under the condition that the performance of our algorithm is satisfying.

TABLE II
NUMBER OF ITERATIONS, TOTAL AMOUNT OF TRAINING SAMPLES AND BFR OF MODIFIED SELF-TRAINING SVM IN VARIOUS CONDITIONS

N	PWA model I [1]		PWA model II [18]	
	$\sigma = 10^{-2}$	$\sigma = 10^{-3}$	$\sigma = 10^{-2}$	$\sigma = 10^{-3}$
1000	$r = 16$ $N_T = 193$	$r = 26$ $N_T = 242$	$r = 18$ $N_T = 215$	$r = 27$ $N_T = 287$
10000	$r = 41$ $N_T = 607$	$r = 45$ $N_T = 679$	$r = 38$ $N_T = 581$	$r = 47$ $N_T = 712$
50000	$r = 58$ $N_T = 1372$	$r = 64$ $N_T = 1907$	$r = 52$ $N_T = 1057$	$r = 61$ $N_T = 1812$
1000	BFR = 88.29	BFR = 92.29	BFR = 89.82	BFR = 89.59
10000	BFR = 89.04	BFR = 93.86	BFR = 92.02	BFR = 91.33
50000	BFR = 90.95	BFR = 93.41	BFR = 93.34	BFR = 91.30

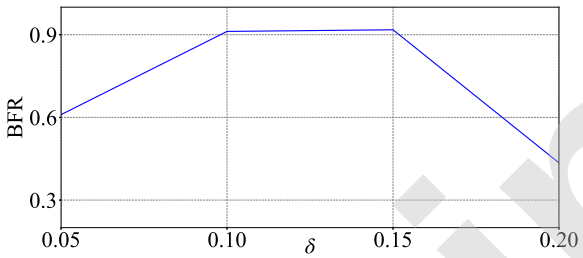


Fig. 6. BFR with respect to the tuning parameter δ in the PWA system (23)

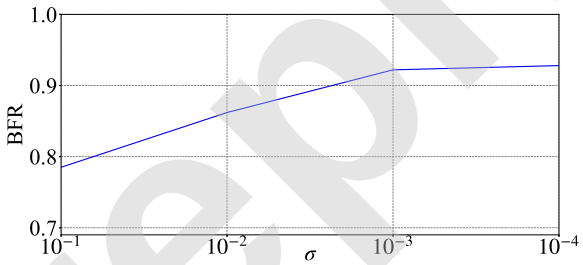


Fig. 7. BFR with respect to tuning parameter σ in the PWA system (23)

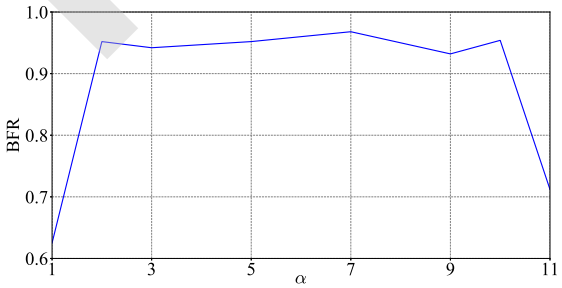


Fig. 8. BFR with respect to tuning parameter α in the PWA system (23)

C. Comparison with previous work

For comparison, the models simulated in the [25], [27] are also identified with our approach by using the data points generated in the same conditions. In [27], a 3-submodel PWA system is identified with its algorithm and BFR value is 74.43%. The same regression problem is solved by our identification algorithm and the BFR equals 90.04%. Moreover, to evaluate the ability of the proposed approach, we approximate the hybrid dynamic model mentioned in [25] with the data points generated in same conditions. Likewise, the BFR value computed with our algorithm is 88.19% whereas 85.19% in [25] and 53.40% in [18] as is listed in Table III. In [25], the hybrid dynamic model is approximated with a PWA system owing 12 local

TABLE III
COMPARISON OF BFR VALUES BETWEEN THE PROPOSED APPROACH AND OTHER RECENT WORKS

Approaches	hybrid dynamic model [25]
Proposed approach	88.37%
Breschi [25]	85.19%
Ferrari-Trecate [18]	53.40%

submodels to obtain the best BFR (85.19%). By using our approach, a 5-submodels PWA system is established with 88.19% BFR. Therefore, the proposed approach provides a better quality PWA model with fewer submodels in the approximation of hybrid dynamic model.

The simulation results indicate that the proposed approach provides a good performance for the PWA system identification, even with a quite low signal/noise ratio. Moreover, fewer data points are required to be labeled and used for training by using the modified self-training algorithm. The total numbers of training samples in various conditions are listed in the simulation. This advantage increases the efficiency of modified self-training algorithm compared to the traditional SVM algorithm. Moreover, compared with the very recent works [25], the simulation results imply that the proposed approach achieves satisfying performance for the approximation of hybrid dynamic systems. As a matter of fact, our algorithm approximates the model even preciser with fewer number of submodels.

VI. CONCLUSION

In this paper, a novel identification approach designed for PWA systems is presented. The approach consists of the cluster-based algorithm and modified self-training SVM algorithm. The submodel number, parameter of each submodel, and hyperplanes of PWA systems are estimated via the approach. By implementing the cluster-based algorithm, we obtain the initial data set and initial parameters, as well as the number of submodels. Furthermore, the modified self-training SVM algorithm is proposed to identify the polyhedral regions. This algorithm is computationally efficient and capable of accomplishing the estimation of the polyhedral region with partially-labeled data set. Naturally, the proposed approach also suffers some minor drawbacks: a) Several parameters, e.g., δ, σ, and α, are required to be tuned. The choice of these parameters may affect the performance of the algorithms and b) The proposed cluster-based algorithm can be time-consuming for very large-scale data set.

Future research will be devoted to searching a more efficient cluster-based algorithm for initialization. In addition, proposing a novel method for the online identification of PWA systems is also considered as future work.

REFERENCES

[1] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino, "A bounded-error approach to piecewise affine system identification," *IEEE Transactions on Automatic Control*, vol. 50, no. 10, pp. 1567–1580, 2005.

[2] Z. Feng and P. Shi, "Sliding mode control of singular stochastic markov jump systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 4266–4273, 2017.

[3] W. Zou, P. Shi, Z. Xiang, and Y. Shi, "Consensus tracking control of switched stochastic nonlinear multiagent systems via event-triggered strategy," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[4] Z. Feng and P. Shi, "Admissibilization of singular interval-valued fuzzy systems," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 6, pp. 1765–1776, 2016.

[5] Y. Wei, J. H. Park, J. Qiu, and H. Jung, "Reliable output feedback control for piecewise affine systems with markov-type sensor failure," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 7, pp. 913–917, 2017.

[6] J. Qiu, Y. Wei, and L. Wu, "A novel approach to reliable control of piecewise affine systems with actuator faults," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 8, pp. 957–961, 2016.

[7] R. Datta, R. Dey, B. Bhattacharya, R. Saravanakumar, and C. K. Ahn, "New double integral inequality with application to stability analysis for linear retarded systems," *IET Control Theory & Applications*, vol. 13, no. 10, pp. 1514–1524, 2019.

[8] F. Lauer, "On the complexity of piecewise affine system identification," *Automatica*, vol. 62, pp. 148–153, 2015.

[9] A. Garulli, S. Paoletti, and A. Vicino, "A survey on switched and piecewise affine system identification," *IFAC Proceedings Volumes*, vol. 45, no. 16, pp. 344–355, 2012.

[10] J. de Jesús Rubio, "Stable kalman filter and neural network for the chaotic systems identification," *Journal of the Franklin Institute*, vol. 354, no. 16, pp. 7444–7462, 2017.

[11] S. Paoletti, A. L. Juloski, G. Ferrari-Trecate, and R. Vidal, "Identification of hybrid systems a tutorial," *European Journal of Control*, vol. 13, no. 2-3, pp. 242–260, 2007.

[12] J. de Jesús Rubio, "Sofmls: online self-organizing fuzzy modified least-squares network," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 6, pp. 1296–1309, 2009.

[13] J. A. Meda-Campaña, "On the estimation and control of nonlinear systems with parametric uncertainties and noisy outputs," *IEEE Access*, vol. 6, pp. 31 968–31 973, 2018.

[14] J. de Jesús Rubio, "Usnfis: uniform stable neuro fuzzy inference system," *Neurocomputing*, vol. 262, pp. 57–66, 2017.

[15] H. Ohlsson and L. Ljung, "Identification of switched linear regression models using sum-of-norms regularization," *Automatica*, vol. 49, no. 4, pp. 1045–1050, 2013.

[16] W.-X. Zhao and T. Zhou, "Weighted least squares based recursive parametric identification for the submodels of a pwarx system," *Automatica*, vol. 48, no. 6, pp. 1190–1196, 2012.

[17] H. Nakada, K. Takaba, and T. Katayama, "Identification of piecewise affine systems based on statistical clustering technique," *Automatica*, vol. 41, no. 5, pp. 905–913, 2005.

[18] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari, "A clustering technique for the identification of piecewise affine systems," *Automatica*, vol. 39, no. 2, pp. 205–217, 2003.
[19] E. Khanmirza, M. Nazarahari, and A. Mousavi, "Identification of piecewise affine systems based on fuzzy pca-guided robust clustering technique," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, p. 131, 2016.
[20] X. Zhu, "Semi-supervised learning literature survey," Computer Sciences, University of Wisconsin-Madison, Tech. Rep. 1530, 2005.
[21] L. Bottou and C.-J. Lin, "Support vector machine solvers," *Large-scale Kernel Machines*, vol. 3, no. 1, pp. 301–320, 2007.
[22] R. Saravanakumar, H. S. Kang, C. K. Ahn, X. Su, and H. R. Karimi, "Robust stabilization of delayed neural networks: Dissipativity-learning approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 3, pp. 913–922, 2018.
[23] U. Ekong, H.-K. Lam, B. Xiao, G. Ouyang, H. Liu, K. Y. Chan, and S. H. Ling, "Classification of epilepsy seizure phase using interval type-2 fuzzy support vector machines," *Neurocomputing*, vol. 199, pp. 66–76, 2016.
[24] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Advances in Neural Information Processing Systems*, 2001, pp. 409–415.
[25] V. Breschi, D. Piga, and A. Bemporad, "Piecewise affine regression via recursive multiple least squares and multicategory discrimination," *Automatica*, vol. 73, pp. 155–162, 2016.
[26] A. L. Juloski, S. Weiland, and W. Heemels, "A bayesian approach to identification of hybrid systems," *IEEE Transactions on Automatic Control*, vol. 50, no. 10, pp. 1520–1533, 2005.
[27] A. K. Shah and D. M. Adhyaru, "Parameter identification of pwarx models using fuzzy distance weighted least squares method," *Applied Soft Computing*, vol. 25, pp. 174–183, 2014.
[28] M. Vaezi and A. Izadian, "Piecewise affine system identification of a hydraulic wind power transfer system," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 6, pp. 2077–2086, 2015.
[29] N. Cristianini, J. Shawe-Taylor *et al.*, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge university press, 2000.
[30] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge university press, 2004.
[31] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.
[32] I. El-Naqa, Y. Yang, M. N. Wernick, N. P. Galatsanos, and R. M. Nishikawa, "A support vector machine approach for detection of microcalcifications," *IEEE Transactions on Medical Imaging*, vol. 21, no. 12, pp. 1552–1563, 2002.
[33] D. M. Christopher, R. Prabhakar, and S. Hinrich, "Introduction to information retrieval," *An Introduction To Information Retrieval*, vol. 151, no. 177, p. 5, 2008.
[34] J. Weston and C. Watkins, "Multi-class support vector machines," Computer Science, Royal Holloway, University of London, Tech. Rep. CSD-TR-98-04, 1998.
[35] Y. Li, C. Guan, H. Li, and Z. Chin, "A self-training semi-supervised svm algorithm and its application in an eeg-based brain computer interface speller system," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1285–1294, 2008.
[36] K. Nigam and R. Ghani, "Analyzing the effectiveness and applicability of co-training," in *Proceedings of the Ninth International Conference on Information and Knowledge Management*, 2000, pp. 86–93.
[37] V. Barnett and T. Lewis, *Outliers in statistical data*. Wiley, 1974.
[38] S. Chatterjee and A. S. Hadi, *Sensitivity Analysis in Linear Regression*. John Wiley & Sons, 2009, vol. 327.
[39] M. Tabatabaei-Pour, K. Salahshoor, and B. Moshiri, "A modified k-plane clustering algorithm for identification of hybrid systems," in *2006 6th World Congress on Intelligent Control and Automation*, vol. 1. IEEE, 2006, pp. 1333–1337.
[40] R. Tóth, *Modeling and Identification of Linear Parameter-varying Systems*. Springer-Germany, 2010, vol. 403.

APPENDIX

Proof. According to the modified self-training SVM Algorithm, we assume $(w^{(1)}, b^{(1)})$ is the classifier at the first iteration, which is the solution of the optimization problem with initial training data set $\mathcal{D}_T$.

$$\begin{aligned} &\min\ \frac{1}{2}||w^{(1)}||^2 + C\sum_{k=1}^{N_1}\xi_k \\ &s.t.\ \ell_k((w^{(1)})^{\mathsf{T}}\mathbf{x}_k + b^{(1)}) \geq 1-\xi_k, k = 1, \cdots, N_1 \end{aligned} \tag{26}$$

where N_1 is the number of data points in $\mathcal{D}_T$, $\mathbf{x}_k$ is the data point, ℓ_k is the corresponding label. The label ℓ_k of each data point remains constant over the iterations.

Clearly, the solution of the optimization problem (26) is identical to the solution of the following problem based on the property of support vectors.

$$\begin{aligned} &\min\ \frac{1}{2}||w^{(1)}||^2 + C\sum_{k=1}^{N_{sv}}\xi_k \\ &s.t.\ \ell_k^{sv}((w^{(1)})^{\mathsf{T}}\mathbf{x}_k^{sv} + b^{(1)}) \geq 1-\xi_k, k = 1, \cdots, N_{sv} \end{aligned} \tag{27}$$

where N_{sv} is the number of support vectors in $\mathcal{D}_T$, $\mathbf{x}_k^{sv}$ is one support vector of $\mathcal{D}_T$, ℓ_k^{sv} is the corresponding label.

For the second iteration of Algorithm 1, the updated training data set D_T is composed of support vectors $\mathbf{x}_k^{sv}$ and the data points from $\mathcal{M}$. We can find that $(w^{(2)}, b^{(2)})$ is the solution of the following optimization problem.

$$\begin{aligned} &\min\ \frac{1}{2}||w^{(2)}||^2 + C\sum_{k=1}^{N_{sv}}\xi_k + C\sum_{i=1}^{q}\xi_i \\ &\text{s.t. } \ell_k^{sv}((w^{(2)})^{\mathsf{T}}\mathbf{x}_k^{sv} + b^{(2)}) \geq 1-\xi_k, k = 1, \cdots, N_{sv} \\ &\quad \ell_i'((w^{(2)})^{\mathsf{T}}\mathbf{x}_i' + b^{(2)}) \geq 1-\xi_i, i = 1, \cdots, q \end{aligned} \tag{28}$$

where (x'_k, ℓ'_k) is the sample of $\mathcal{M}$. The classifier is trained with the updated D_T. From the equations (27) and (28), $(w^{(2)}, \xi^{(2)}, b^{(2)})$ is a feasible solution of the equation (26). Since $(w^{(1)}, \xi^{(1)}, b^{(1)})$ is an optimal solution of the equation (26), we have

$$f(w^{(1)}, \xi^{(1)}) \leq f(w^{(2)}, \xi^{(2)}) \tag{29}$$

where

$$f(w, \xi) = \frac{1}{2}||w||^2 + C\sum_{k=1}^{N} \xi_k. \tag{30}$$

It yields that

$$f(w^{(j-1)}, \xi^{(j-1)}) \leq f(w^{(j)}, \xi^{(j)}). \tag{31}$$

For a given labeled data set $\mathcal{D}$, the value of $f(w, \xi)$ trained with the modified self-training SVM is bounded. The maximum value of f is obtained while (w^*, ξ^*, b^*) is the solution of

$$\begin{aligned} &\min \ \frac{1}{2}||w^*||^2 \\ &\text{s.t. } \ell_k({w^*}^{\mathsf{T}}\mathbf{x}_k + b^*) \geq 1 - \xi_k^*, k = 1, \cdots, N \end{aligned} \tag{32}$$

where $(\mathbf{x}_k, \ell_k)$ are the data points of $\mathcal{D}$. Thus $\frac{1}{2}||w^{(j)}||^2$ of the modified self-training SVM is convergent and monotonically non-decreasing over the iterations.

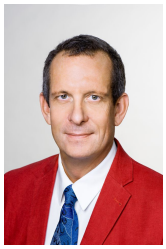
Yingwei Du received the B.Sc. degree in electrical engineering from Harbin Institute of Technology, Harbin, China, in 2014 and the M.Sc. degree in electrical engineering from Harbin Institute of Technology, Harbin, China, in 2016. He is currently pursuing the Ph.D. degree in electrical engineering in the Chair of Automatic Control Engineering, Technical University of Munich, Germany. His current research interests include hybrid discrete-continuous systems, adaptive control, and system identification.

Fangzhou Liu (S'14-M'19) received the M.Sc. degree in control theory and engineering from Harbin Institute of Technology, Harbin, China, in 2014 and the Doktor-Ingenieur degree in electrical engineering from Technical University of Munich, Germany, in 2019. He is now a Lecturer and Research Fellow with the Chair of Automatic Control Engineering, Technical University of Munich, Germany. His current research interests include networked control systems; modeling, analysis, and control on social networks; and their applications.

Jianbin Qiu (M'10-SM'15) received the B.Eng. and Ph.D. degrees in Mechanical and Electrical Engineering from the University of Science and Technology of China, Hefei, China, in 2004 and 2009, respectively. He also received the Ph.D. degree in Mechatronics Engineering from the City University of Hong Kong, Kowloon, Hong Kong, in 2009.

He is currently a Full Professor at the School of Astronautics, Harbin Institute of Technology, Harbin, China. He was an Alexander von Humboldt Research Fellow at the Institute for Automatic Control and Complex Systems, University of Duisburg-Essen, Duisburg, Germany. His current research interests include intelligent and hybrid control systems, signal processing, and robotics.

Prof. Qiu is a Senior Member of IEEE and serves as the chairman of the IEEE Industrial Electronics Society Harbin Chapter, China. He is an Associate Editor of *IEEE Transactions on Cybernetics*.



Martin Buss (F'14)received the Diploma Engineering degree in electrical engineering from the Technische Universität Darmstadt, Darmstadt, Germany, in 1990 and the Doctor of Engineering degree from the University of Tokyo, Tokyo, Japan, in 1994, both in electrical engineering.

In 1988, he was a Research Student for one year with Science University of Tokyo. From 1994 to 1995, he was a Postdoctoral Researcher in the Department of Systems Engineering, Australian National University, Canberra, ACT, Australia. From 1995 to 2000, he was a Senior Research Assistant and Lecturer in the Chair of Automatic Control Engineering, Department of Electrical Engineering and Information Technology, Technical University of Munich,Munich, Germany. From 2000 to 2003, he was a Full Professor, the Head of the Control Systems Group, and the Deputy Director of the Institute of Energy and Automation Technology, Faculty IV, Electrical Engineering and Computer Science, Technical University Berlin, Berlin, Germany. Since 2003, he has been a Full Professor (Chair) in the Chair of Automatic Control Engineering, Faculty of Electrical Engineering and Information Technology, Technical University of Munich, where he has been in the Medical Faculty since 2008. He has been awarded the ERC Advanced Grant SHRINE. His research interests include automatic control, mechatronics, multimodal human system interfaces, optimization, nonlinear, and hybrid discrete-continuous systems.