Hybrid System Identification through Optimization and Active Learning

Hadi Dayekh, Nicolas Basset, Thao Dang

Hybrid System Identification through Optimization and Active Learning

Hadi Dayekh * **Nicolas Basset** * **Thao Dang** *

* *Université Grenoble Alpes, CNRS, Grenoble INP, VERIMAG, Grenoble, France (e-mail: firstname.lastname@univ-grenoble-alpes.fr).*

Abstract: We present a method to identify state-dependent switched nonlinear dynamical systems with polynomial ODEs through optimization and active learning. Our approach extends and incorporates segmentation into a previous optimization-based approach for identifying SARX models. We use logistic regression to find the polynomial or linear mode boundaries of the system. Additionally, we provide a way to refine the result of the classifier through active learning and equivalence queries, assuming the correct identification of continuous dynamics. We provide results of our approach on multiple experiments, including a parametric experiment with increasing number of modes. We also compare our results with a different approach that deals with a similar class of problems and show that our method performs better.

Keywords: Hybrid Systems, Identification, Data-driven Modeling, Active Learning

1. INTRODUCTION

The problem of identifying hybrid systems attracted much attention as soon as hybrid system models became widely accepted in the 1990s by both computer science and control theory communities as a rigorous mathematical tool for describing and reasoning about processes involving both continuous and discrete dynamics. Due to the recent progress in machine learning, this identification problem, as part of data-driven model learning, is currently enjoying renewed interest. The classical identification problem can be seen as a passive model learning problem. In active model learning, we assume to be able to manipulate the system inputs and initial states to generate system trajectories, and then the question is how to generate new data to improve the model accuracy. Active learning has been intensively studied in the context of automata learning since Angluin [1987]. To our knowledge, active learning has not yet been much explored for hybrid systems.

In this paper, we propose an approach to identify switched nonlinear dynamical systems. Before explaining our contribution, we present a brief review of related work on (passive) hybrid system identification.

The first hybrid system identification work focused on affine hybrid systems and addressed Piecewise AutoRegressive eXogenous (PWARX) or Switched AutoRegressive eXogenous (SARX) models (see the tutorial of Paoletti et al. [2007]). One major approach relies on a solution where classifying data (that is, assigning a submodel to each data point) is coupled with estimating the dynamics of the submodels. This is done in Ferrari-Trecate et al. [2001] using clustering techniques, by exploiting the observation that data points close to each other are likely to belong to the same submodel. The bounded-error method proposed in Bemporad et al. [2005] performs the coupling in an iterative manner by first aiming at matching data points with a minimum number of submodels up to a given fitting error bound, and then refining to resolve matching inconsistency. Another major approach Vidal et al. [2003] is based on an algebraic formulation that describes the match of a data point to one submodel by a homogeneous polynomial equation. An extension of this algebraic approach to handle data noise was proposed in Ozay et al. [2009]. In Juloski et al. [2005], a Bayesian approach is used to identify PWARX models. On the other hand, to address the complexity of the optimisation problems associated with the identification procedures, Ozay [2016] solves the problem of segmentation of an ARX trajectory with a minimum number of mode changes using dynamic programming. Non-smooth programming is exploited in Berger et al. [2022] for identifying switched linear systems, to achieve an algorithm linear in the size of the data, instead of exponential as in a direct mixed-integer linear programming approach Bemporad et al. [2001].

For nonlinear hybrid systems, Lauer and Bloch [2008] propose an extension of the bounded-error approach where the dynamics of submodels are expressed in kernel expansion form and the associated relaxed optimization problem is then solved using kernel support vector regression (SVR). Besides optimization-based solutions, machine learning techniques are used, such as in Ly and Lipson [2012] which employs symbolic regression to infer a symbolic discrete dynamical model with continuous mappings.

While the above-mentioned approaches concern mainly discrete-time systems, other approaches are emerging to identify hybrid automata with ODEs, such as for systems with affine ODEs Soto et al. [2021], Yang et al. [2022]. Additionally, mode classification is done using shape-based similarity measures of trajectories, such as Gurung et al. [2023], Saberi et al. [2022]. On the other hand, the approach proposed in Jin et al. [2021] can identify switched nonlinear dynamical systems, based on either a novel

algorithm that combines segmentation with a merging algorithm which merges segments together if they fit well in a regression model, or a modified application of Alur and Singhania [2014]. This approach is closely related to the one we propose. However, it requires a stronger constraint on the nature of the state space partition.

To situate our work in the current state of the art, the identification approach we propose in the paper deals with state-dependent switched nonlinear dynamical systems and is based on a modified version of the approach in Lauer and Bloch [2008], presented in section 2. The first major novelty in our approach is the integration of segmentation information in the optimization problem, and the direct training of system coefficients rather than relying on kernel methods (section 3). The second major novelty is the use of active learning to correctly find the state space partition of the modes, assuming the correct identification of continuous dynamics (section 4). We provide results on several experiments in section 5, and finally, we conclude and talk about future work in section 6.

2. BACKGROUND AND PROBLEM STATEMENT

We first recall the method of Lauer and Bloch [2008] for identifying discrete-time SARX models. Given pairs of an input $\boldsymbol{u}_t$ and an observable output y_t, as well as model orders n_a and n_c, a *regressor vector* is built as follows:

$$\boldsymbol{x}_t = \left(y_{t-1} \dots \ y_{t-n_a} \boldsymbol{u}_{t-1}^T \dots \boldsymbol{u}_{t-n_c}^T\right)$$

The output y_t depends on this regressor such that $y_t = \boldsymbol{g}_j(\boldsymbol{x}_t)$, for some $j \in \{1, \dots, n\}$, representing the mode of the system.

In the affine case, given the data points $\{\boldsymbol{x}_i, y_i\}_{i=1}^{i=N}$, where $y_i = g_j(\boldsymbol{x}_i)$ and $j \in \{1, \dots, n\}$, the target function for each mode j is of the form $\hat{g}_j(\boldsymbol{x}) = \boldsymbol{\omega}_j^T \boldsymbol{x} + \beta_j$. The goal is to identify these functions and the modes of each data point. The optimization problem for identifying affine [1] SARX systems developed by Lauer and Bloch [2008] is:

$$\min_{\boldsymbol{\xi} \geq 0, \boldsymbol{\omega}, \beta} \sum_{j=1}^{n} \boldsymbol{\omega}_j^T \boldsymbol{\omega}_j + C \sum_{i=1}^{N} \prod_{j=1}^{n} \xi_{ij}$$

$$-\xi_{ij} - \delta_j \leq y_i - \underbrace{\left(\boldsymbol{\omega}_j^T \boldsymbol{x} + \beta_j\right)}_{\hat{g}_j(\boldsymbol{x})} \leq \delta_j + \xi_{ij}, \ i=1,\dots,N, \ j=1,\dots,n$$

where δ_j is an insensitivity error for each mode j.

The global minimum is reached when the second term in the objective function is zero. In this global minimum, for each data point x_i, there exists a $j^* \in \{1, \dots, n\}$ such that $\xi_{ij^*} = 0$. Given this information, we know that the data point x_i is correctly described by the dynamics of $\hat{f}_{j^*}$, up to an insensitivity error δ_{j^*}, as the constraint now ensures that. Hence, $\boldsymbol{x}_i$ belongs to mode j^*. The objective function also includes the sum of norms of coefficients $\boldsymbol{\omega}_j$, a *regularization* step that aims at hindering overfitting. The mode of a data point x_i is deduced as $\arg\min_j \xi_{ij}$.

[1] Lauer and Bloch [2008] also provides a way to identify nonlinear SARX models through kernel SVR. In our learning algorithm, we treat the nonlinear case as affine after transforming the data into a polynomial feature space. We thus do not use kernels and hence will not detail their optimization problem for the nonlinear case

Problem Statement

We aim at identifying a continuous-time state-dependent Switched Nonlinear Dynamical System (SNDS), which is a type of hybrid systems whose continuous dynamics are governed by n ordinary differential equations, as follows:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}_j(\boldsymbol{x}) \ \text{ if } \boldsymbol{x} \in \mathcal{X}_j, \ j = 1, \dots, n$$

where $\boldsymbol{x} \in \mathbb{R}^D = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \dots \cup \mathcal{X}_n$ and $\mathcal{X}_i \cap \mathcal{X}_j = \varnothing, \ i \neq j$. Furthermore, we only consider $\boldsymbol{f}_j(\boldsymbol{x})$ that are polynomial in the components of $\boldsymbol{x}$, with a known maximum degree. We also assume each $\mathcal{X}_j$ is defined by a conjunction of polynomial inequalities.

Given data points $\{t_k, \boldsymbol{x}(t_k)\}$ collected from a set of trajectories over time at discrete time steps, we present a way to identify the dynamical equations $\boldsymbol{f}_1, \boldsymbol{f}_2, \dots, \boldsymbol{f}_n$ as well as the mode each data point belongs to, assuming we know the number of modes n. At a later step, knowing the modes of each of the training data points, we apply a classical machine learning classification problem to deduce the state space partition $\{\mathcal{X}_j\}_{j=1}^{j=n}$.

3. PASSIVE LEARNING OF DYNAMICS

3.1 Identifying Dynamics and Data Modes

In this section, we modify the optimization-driven method of Lauer and Bloch [2008] for SARX systems to our continuous-time problem, through three major changes: (1) changing the nature of "input" and "output" in SARX systems as to respectively refer to the continuous state $\boldsymbol{x}$ and its estimated derivative $\dot{\boldsymbol{x}}$, (2) using polynomial features to identify polynomial functions, and (3) incorporating in the optimization problem (see section 2) the segmentation information by merging all ξ variables that represent points belonging to the same segment. The last modification greatly reduces the complexity of the optimization problem as we go from $n \cdot N$ variables ξ to $n \cdot m$, with m being the number of segments.

Segmentation The first step in the identification process is segmentation, which is dividing each trajectory into segments such that each segment follows a certain dynamic and thus belongs to one mode. We integrate segmentation in the identification process, as this greatly reduces the complexity of the optimization problem.

In order to segment each trajectory, we follow a similar method to Jin et al. [2021], where we compare forward and backward derivatives at each point of the trajectory, concluding that there is a change in modes when the difference is greater than a certain threshold.

After the segmentation process is done, we get a set of non-intersecting segments $S = \{S_1, S_2, \dots, S_m\}$, whose union represents our training data. We denote by $\sigma(i)$ the index of the segment in which the point $\boldsymbol{x}_i$ belongs.

Estimating Derivatives Given data points $\{t_k, \boldsymbol{x}(t_k)\}$ collected from trajectories over time at discrete time steps, we use the five-point-stencil method Sauer [2012], which uses forward and backward data along the trajectory, to estimate the derivatives of points in each segment.

Our training data set is then $\boldsymbol{X} = (\boldsymbol{x}_1 \boldsymbol{x}_2 \cdots \boldsymbol{x}_N)^T$, $\dot{\boldsymbol{X}} = (\dot{\boldsymbol{x}}_1 \dot{\boldsymbol{x}}_2 \cdots \dot{\boldsymbol{x}}_N)^T$ where $x_i^{(d)}$ and $\dot{x}_i^{(d)}$ respectively represent the dth element of $\boldsymbol{x}_i$ and $\dot{\boldsymbol{x}}_i$, each of dimension D.

Extracting Polynomial Features We now extract polynomial features by transforming each point in the dataset to include all monomials of that point's vector components up to a certain degree. This allows us to then treat a nonlinear problem as affine on the transformed dataset instead of using kernels as in Lauer and Bloch [2008].

Given a vector $\boldsymbol{x}$ of dimension D and a polynomial degree γ, denote by $\boldsymbol{\Psi}_\gamma(\boldsymbol{x})$ the vector of polynomial features extracted from $\boldsymbol{x}$, excluding 1. The dimension of this vector is $Q = \binom{D+\gamma}{\gamma} - 1$. We also define $\boldsymbol{\Psi}_\gamma(\boldsymbol{X})$ as the matrix of extracted features of all row vectors $\boldsymbol{x}$ in $\boldsymbol{X}$. For affine systems, $\gamma = 1$, $\boldsymbol{\Psi}_1(\boldsymbol{X}) = \boldsymbol{X}$ and $Q = D$.

Optimization Problem The goal now is to find each of the vector functions $\boldsymbol{f}_j(\boldsymbol{x})$ as well as the modes of each $\boldsymbol{x}_i$ in the training data. In other words, for each mode j and vector component d, we have to find a scalar function of the form $\hat{f}_j^{(d)}(\boldsymbol{x}) = \boldsymbol{\Psi}_\gamma(x)^T \cdot \boldsymbol{\omega}_j^{(d)} + \beta_j^{(d)}$, where $\boldsymbol{\omega}_j^{(d)}$ is a vector of Q coefficients and $\beta_j^{(d)}$ is a bias term. The optimization problem in the multidimensional case is:

$$\min_{\boldsymbol{\xi} \geq 0, \boldsymbol{\omega}, \boldsymbol{\beta}} \sum_{r=1}^{m} \prod_{j=1}^{n} \xi_{rj} + C \sum_{d=1}^{D} \sum_{j=1}^{n} {\boldsymbol{\omega}_j^{(d)}}^T \boldsymbol{\omega}_j^{(d)}$$

$$-\xi_{\sigma(i)j} - \delta_j \leq \dot{x}_i^{(d)} - \left(\boldsymbol{\Psi}_\gamma(\boldsymbol{x}_i)^T \cdot \boldsymbol{\omega}_j^{(d)} + \beta_j^{(d)} \right) \leq \delta_j + \xi_{\sigma(i)j}$$

$$i=1,\ldots,N,\ j=1,\ldots,n,\ d=1,\ldots,D$$

The mode of a segment S_r is deduced to be $\arg\min_j \xi_{rj}$.

We place C in front of the norm of coefficients, giving it a more familiar meaning as a regularization hyperparemeter.

Note that unlike Lauer and Bloch [2008], we find the continuous dynamics coefficients directly by extracting polynomial features, instead of using kernel methods. For small polynomial degrees γ and dimensions D, our approach is more efficient as it reduces the number of decision variables in the optimization problem. It also overcomes the need to compute kernel matrices, each of which having a space complexity of N^2. Using polynomial features allows us to identify many nonlinear examples that the kernel-based optimization problem fails to identify.

Hyperparameter Tuning In the aforementioned optimization problem, we encounter four main hyperparameters: the number of modes n, the maximum polynomial degree γ, the insensitivity error δ, and the regularization hyperparameter C. As for the first three, here we assume that they are user defined. What remains is the hyperparameter C, which expresses how much regularization we want to incorporate into the minimization problem.

In classical machine learning, a very common technique for hyperparameter tuning is to split the data into a training set and a validation set (Hastie et al. [2009]). For different values of the hyperparameter, a model is trained using the training set, then it is tested on the validation set, allowing to calculate a validation error. The hyperparameter value that minimizes this validation error is selected.

We split each segment separately to construct our training and validation datasets, ensuring that each segment gets its share of training and validation points. This way, we are sure that the segment in which each validation point belongs to contains training data. Hence, the optimizer already found which mode the entire segment belongs to. Using this information, we can deduce the mode of these validation points and know which function to use to find the predicted derivatives and compute the validation error. Moreover, the validation error would be calculated using data from all segments and would thus be more representative of the state space and the trajectories.

3.2 Finding the State Space Partition

To wrap up a complete model, we need to find the state space partition of the modes. Given the learned modes of the data points, we set up a multi-class classification problem using logistic regression. For polynomial boundaries, we extract polynomial features similarly to what we discussed in subsection 3.1, where $\boldsymbol{\Psi}_\kappa(\boldsymbol{x})$ is the vector of polynomial features of $\boldsymbol{x}$ up to a certain degree κ.

Binary case ($n = 2$). The model provides coefficients of the boundary that separates the two classes/modes: $\mathcal{B}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\Psi}_\kappa(\boldsymbol{x}) + b = 0$, consequently defining the two regions: $\mathcal{X}_1 = \{\boldsymbol{x} \ : \ \mathcal{B}(\boldsymbol{x}) \leq 0\}$, $\mathcal{X}_2 = \{\boldsymbol{x} \ : \ \mathcal{B}(\boldsymbol{x}) \geq 0\}$.

Multi-class case ($n > 2$). As for the multi-class case, we opt for multinomial logistic regression. The model returns a set of coefficients $(\boldsymbol{w}_j, b_j)$ per mode j. The regions that construct the state space partition for the modes are:

$$\mathcal{X}_j = \left\{ \boldsymbol{x} \ : \ \bigwedge_{k \neq j} (\boldsymbol{w}_j - \boldsymbol{w}_k)^T \boldsymbol{\Psi}_\kappa(\boldsymbol{x}) + (b_j - b_k) \geq 0 \right\}$$

Furthermore, the decision boundary between modes j and k is $(\boldsymbol{w}_j - \boldsymbol{w}_k)^T \boldsymbol{\Psi}_\kappa(\boldsymbol{x}) + (b_j - b_k) = 0$. One could use linear programming techniques to get rid of redundant or useless hyperplanes that may occur (Bemporad et al. [2005]).

4. ACTIVE LEARNING OF BOUNDARIES

In many of the examples tried, including affine systems, the original trajectories and those simulated from the learned system differ by shape not due to a difference in the dynamical equations between the original and learned systems, but rather due to wrong state space partition. In this section, we explain how we refine the result of the classifier by using active learning. We assume we have access to the system under learning (SUL) as a black box that we can execute to generate trajectories given an initial condition $\boldsymbol{x}_0$, a time step t_s, and a time range T. Furthermore, we also assume that we correctly identify the continuous dynamics of the system, and thus we only need to improve the classification boundaries.

In principle, the idea is to execute the SUL to generate new data points then estimate their derivatives through numerical differentiation methods. Having these points and their derivatives, we can now guess their modes

by comparing the estimated derivatives with the learned dynamics of each mode and choosing the mode that minimizes the difference. Finally, we add this new data to the training data of the classifier and retrain it.

The main question here is: *how to choose the initial conditions of these new trajectories?*

4.1 By Generating Points Between Segments

We argue that points near the classifier boundaries play an important role in the classifier result, as these crucially separate one mode from another. That being said, a first step is to execute the SUL to generate finely spaced trajectories between extremities of consecutive segments. This ensures that the new trajectories cross mode boundaries, thus helping refine the classifier results. We re-perform segmentation on this newly generated set of data points, as to avoid erroneously estimating the derivatives by using points from different modes.

4.2 Through Equivalence Query Checking

In active learning terminology, given a hypothesis H of the learned system, an equivalence query is a check made to determine whether H is equivalent to the SUL. In case is not, the equivalence query provides a counterexample of an execution that differs between H and the SUL. This counterexample is used to improve the hypothesis H before posing a new equivalence query. Such improvements are done until the equivalence query returns true, in which case the learned system and the SUL are equivalent.

We apply a similar technique to improve the state space partition, still assuming that the dynamics are correctly identified in the passive learning step. As there is no formal check to compare the SUL and a hypothesis H, we approximate this check by comparing a large number of executed trajectories between the two systems.

To approximately check for an equivalence query, we randomly choose a certain number $ntrajs$ of initial conditions within predefined bounds of the state space, following some coverage criteria (Dang and Nahhal [2009]). The SUL and the learned hypothesis are then executed with the same time step and time range. We then proceed to compare each trajectory of the SUL with its corresponding one from the learned system, by computing an average of the relative difference (RD), a normalized difference introduced in Jin et al. [2021], between each pair of corresponding points in these trajectories.

As long as the average relative differences of some pairs of trajectories are greater than a certain tolerance, we consider these trajectories to be the counterexamples. We then apply segmentation to these counterexample trajectories to generate finely spaced points between extremities of segments (if any), as in subsection 4.1. All this new data is used to refine the classifier result.

5. EXPERIMENTS

We demonstrate our approach through a parametric example with different numbers of modes, as well as eight other examples, five of which are taken from Jin et al. [2021]. We run the optimization problem using inter-point optimization with , made accessible in Python with CasADi (Andersson et al. [2019]). Additionally, we use the scikit-learn library for extracting polynomial features, hyperparameter tuning, and logistic regression classification. As for the simulation of trajectories, we use the `solve_ivp` function of SciPy. The experiments are run on a MacBook Pro, with the M1 Pro chip and 32 GB of memory. For each experiment, we perform a hyperparameter search of C among the values: 0, 0.001, 0.01, 0.1, and 1, but we report results on the hyperparameter that minimizes the validation error. Furthermore, in all the experiments, we train with a single insensitivity error $\delta = 0.001$. Moreover, in the active learning part, we consider 50 test trajectories per equivalence query, with RD tolerance of 0.1.

In order to test our learning algorithm, we consider a fixed set of test trajectories generated by the SUL and chosen with random initial conditions. At every step (passive learning, active learning through generating points between segments of training data, and active learning through equivalence queries), we provide the average RD between these SUL test trajectories and those generated from the hypothesis with the same initial conditions, as well as the time of computation and number of equivalence queries. Furthermore, we provide the average RD of the predicted and true derivatives of each point of the test trajectories. This is to compare with the two methods implemented by Jin et al. [2021] who use this measure.

5.1 Parametric Example

We provide a parametric example with linear dynamics and a nonlinear state space partition in 2D. The example is parametric in the number of modes n, designed to try and push the limits of our learning model by choosing increasing number of modes.

Given a two-dimensional square box centered around $O = (0, 0)$ with side length equal to 2, we construct $n-1$ equally spaced circles centered around O inside this box, such that the radius of the smallest circle is $1/n$ and the difference between the radii of two consecutive circles is also $1/n$. These circles act as the mode barriers in the state space.

More concretely, given a point $\boldsymbol{x}$, define $d^2(\boldsymbol{x})$ to be the squared distance from the origin O. The state space partition is defined as follows:

$$\mathcal{X}_1 = \left\{ \boldsymbol{x} : d^2(\boldsymbol{x}) \geq \left(\frac{n-1}{n}\right)^2 \right\}$$

$$\mathcal{X}_j = \left\{ \boldsymbol{x} : \left(\frac{n-j}{n}\right)^2 \leq d^2(\boldsymbol{x}) < \left(\frac{n-j+1}{n}\right)^2 \right\}, \; {}_{1<j\leq n}$$

Furthermore, the continuous dynamics in a mode j are $\dot{x}^{(1)} = -2.5x^{(1)} + \alpha_j x^{(2)}$ and $\dot{x}^{(2)} = -\alpha_j x^{(1)} - 2.5x^{(2)}$, where $\alpha_j = 2j$ if j is odd, otherwise $\alpha_j = -2j$, leading to alternating spiraling towards the center between clockwise and anti-clockwise. Fig. 1 illustrates a set of trajectories executed with the parametric example for 3 and 5 modes.

We run our algorithm on this example with different numbers of modes. For each run, we choose the same initial conditions $(1, -1), (-1, 1), (-1, -1)$ for the training data, with time step $t_s = 0.02$ and time range $T = 1$.

Table 1. Experimental results on the parametric example with different number of modes n.

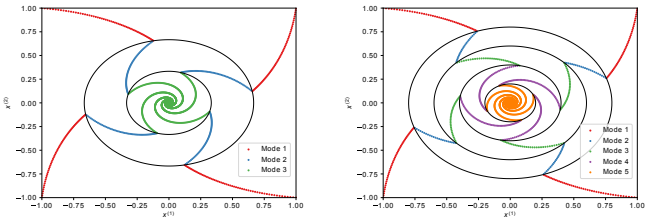
n	d_1	d_2	d_3	t_1	t_2	t_3	N_{EQ}	d_{der}	d_{merge}	d_{pwa}	t_{merge}	t_{pwa}
2	0.15205	0.14529	0.00182	0.4	0.2	20.8	2	1.5e-03	4.5e-02	4.5e-02	0.1	0.1
3	0.21835	0.14256	0.00661	0.7	0.4	22.6	2	3.4e-06	2.1e-01	1.3e-01	0.1	0.2
4	0.27007	0.18318	0.00543	1.2	0.7	54.0	2	5.2e-03	–	–	–	–
5	0.37468	0.23661	0.01390	5.6	0.9	138.5	2	3.8e-03	–	–	–	–

d_1, d_2, d_3: The average relative difference of test trajectories after passive learning, after active learning through generating points between training data segment extremities, and after active learning through equivalence queries respectively.
t_1, t_2, t_3: The non-cumulative running time (in seconds) of the passive learning with the best hyperparameter C, active learning through generating points between training data segment extremities, and active learning through equivalence queries respectively.
N_{EQ}: The number of equivalence queries posed.
d_{der}: The average relative difference of the derivatives of test data points with the final learned model.
d_{merge}, d_{pwa}: The average relative difference of the derivatives of test data points using the methods of Jin et al. [2021].
t_{merge}, t_{pwa}: The non-cumulative running time (in seconds) of the methods of Jin et al. [2021].

Table 2. Information about other experiments: the number of modes n, the state vector dimension D, and the polynomial degrees γ and κ for the dynamics and classifier respectively.

Experiment Name	n	D	γ	κ
Isolette†	2	2	1	1
Three Mode Affine	3	2	1	1
Switched Lorentz Attractor†	2	3	2	1
Three Mode Poly-2†	3	2	2	1
Two Mode Poly-3†	2	2	3	2
Four Dimensional†	2	4	1	1
Biological System	2	2	3	1
Hamiltonian	2	2	3	2

† This experiment is taken from Jin et al. [2021].



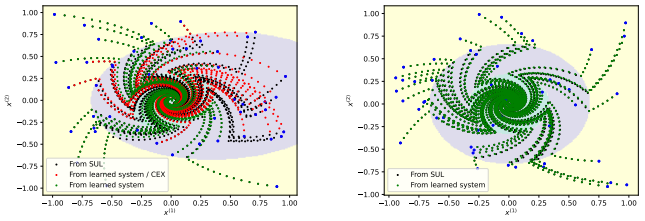
(a) $n = 3$ (b) $n = 5$

Fig. 1. A set of trajectories given with the parametric example for $n = 3$ and $n = 5$.

Table 1 shows our results for the parametric example, with n ranging from 2 to 5. Our optimization algorithm was not able to recover the dynamics for number of modes greater than 5, due to the complexity of the optimization problem.

In this experiment, we choose initial conditions that are asymmetrical in the state space of the system to highlight the importance of active learning. Indeed, we notice a great improvement across all numbers of modes after refining the classifier through equivalence queries. Furthermore, given this refinement, our algorithm performs better than that of Jin et al. [2021] when $n = 2$ and $n = 3$. Their provided code does not support a higher number of modes.

By looking at the results, we also note that the active learning part, mainly posing equivalence queries, takes the most amount of time. This is due to the cost of simulation of the learned model across different trajectories, as well as retraining the classifier with a large amount of data.



(a) First equivalence query. (b) Second equivalence query.

Fig. 2. Result of each equivalence query posed in the three mode parametric example, showing conforming (green) and counterexample (red) trajectories.

As for the latter part, one could be more selective when it comes to the data used for retraining by discarding new points that are far from the classifier decision boundaries, which we do not do at this point yet.

Fig. 2 shows the result of the equivalence queries for the parametric example with $n = 3$. As seen in Fig. 2a, the first equivalence query returns many counterexamples, which are used to refine the boundaries, resulting in an equivalence query with no counterexamples (Fig. 2b).

5.2 Other Experiments

In addition to the parametric example, we run our learning algorithm on eight other experiments, with different polynomial degrees of dynamics and of the mode boundaries. Table 2 provides information about each experiment.

Similarly to the parametric example, Table 3 provides our experimental results for these examples, with the same measures. We notice that in all cases, the active learning of the boundaries improves the average RD of test trajectories. Moreover, our final model performs better than those of Jin et al. [2021] in all cases.

6. CONCLUSION AND FUTURE WORK

We present an optimization based approach for the identification of state-dependent switched nonlinear dynamical systems, augmented with active learning of the discrete dynamics. Our approach in learning the continuous dynamics and modes of training data is based on a previously proposed optimization for identifying SARX systems by Lauer and Bloch [2008]. In addition to integrating segmentation

Table 3. Experimental results on examples whose characteristics are described in Table 2.

Experiment Name	d_1*	d_2*	d_3*	t_1*	t_2*	t_3*	N_{EQ}*	d_{der}*	d_{merge}*	d_{pwa}*	t_{merge}*	t_{pwa}*
Isolette	0.00029	0.00004	0.00004†	0.5	0.2	14.0	1	2.9e-11	1.7e-03	1.7e-03	0.1	0.2
Three Mode Affine	0.21167	0.11653	0.01669	4.3	0.3	31.5	3	3.2e-03	6.3e-03	6.3e-03	0.1	0.2
Switched Lorentz Attractor	0.01853	0.00076	0.00076†	4.7	1.3	70.1	1	1.0e-05	2.0e-02	2.0e-02	0.2	0.3
Three Mode Poly-2	0.15703	0.00974	0.01038	1.5	0.6	72.7	2	2.4e-03	5.8e-03	6.9e-03	0.3	0.4
Two Mode Poly-3	0.02233	0.00308	0.00308†	3.1	0.4	81.7	1	6.5e-04	1.5e-02	2.3e-01	0.8	0.3
Four Dimensional	0.00531	0.00056	0.00056†	5.3	0.5	72.2	1	5.7e-04	4.7e-03	1.1e-02	1.9	0.4
Biological System	0.00357	0.00057	0.00057†	7.5	0.3	88.5	1	7.8e-05	6.6e-04	6.7e-03	0.8	1.6
Hamiltonian	0.17670	0.17670	0.00251	4.5	0.1	60.1	3	5.6e-03	2.1e-02	2.1e-02	7.6	0.4

* For the meaning of the different measures, refer to Table 1.

† Equivalent to d_2 as the first equivalence query returns no counterexamples.

and identifying coefficients directly, we propose a way to tune hyperparameters of the optimization, as well as a generative approach to improving classification of modes through active learning. Our approach achieves very good experimental results on various affine and polynomial examples with varying number of modes and dimensions. Compared to Jin et al. [2021], our algorithm achieves better results across all of our experiments.

A limitation of our current approach is that we assume knowing the number of modes and the maximum degree of the polynomial. Our future work includes doing experiments on over- and underestimating these quantities during the identification process.

We intend to incorporate active learning for the continuous dynamics and to extend the approach to hybrid automata with resets in the mode switches. We also plan to provide an analysis of the completeness and convergence conditions of our method in the future, in particular the probably-approximate correctness (PAC) of the algorithms.

REFERENCES

Alur, R. and Singhania, N. (2014). Precise piecewise affine models from input-output data. In *Proceedings of the 14th Int. Conf. on Embedded Software*, EMSOFT '14, 1–10. Association for Computing Machinery.

Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., and Diehl, M. (2019). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1), 1–36.

Angluin, D. (1987). Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2), 87–106.

Bemporad, A., Garulli, A., Paoletti, S., and Vicino, A. (2005). A bounded-error approach to piecewise affine system identification. *IEEE Transactions on Automatic Control*, 50(10), 1567–1580.

Bemporad, A., Roll, J., and Ljung, L. (2001). Identification of hybrid systems via mixed-integer programming. In *Proceedings of the 40th IEEE Conf. on Decision and Control (Cat. No.01CH37228)*, volume 1, 786–792 vol.1.

Berger, G., Narasimhamurthy, M., Watanabe, K., Lahijanian, M., and Sankaranarayanan, S. (2022). An algorithm for learning switched linear dynamics from data. In *Advances in Neural Information Processing Systems*, volume 35, 30419–30431. Curran Associates, Inc.

Dang, T. and Nahhal, T. (2009). Coverage-guided test generation for continuous and hybrid systems. *Formal Methods in System Design*, 34(2), 183–213.

Ferrari-Trecate, G., Muselli, M., Liberati, D., and Morari, M. (2001). A clustering technique for the identification of piecewise affine systems. In *Hybrid Systems: Computation and Control*, 218–231. Springer Berlin Heidelberg.

Gurung, A., Waga, M., and Suenaga, K. (2023). *Learning Nonlinear Hybrid Automata from Input–Output Time-Series Data*, volume 14215 of *Lecture Notes in Computer Science*, 33–52. Springer Nature Switzerland.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics.

Jin, X., An, J., Zhan, B., Zhan, N., and Zhang, M. (2021). Inferring Switched Nonlinear Dynamical Systems. *Formal Aspects of Computing*, 33(3), 385–406.

Juloski, A., Weiland, S., and Heemels, W. (2005). A bayesian approach to identification of hybrid systems. *IEEE Trans. on Automatic Control*, 50(10), 1520–1533.

Lauer, F. and Bloch, G. (2008). Switched and PieceWise Nonlinear Hybrid System Identification. In *HSCC*, volume 4981, 330–343. Springer Berlin Heidelberg.

Ly, D.L. and Lipson, H. (2012). Learning symbolic representations of hybrid dynamical systems. *Journal of Machine Learning Research*, 13(115), 3585–3618.

Ozay, N. (2016). An exact and efficient algorithm for segmentation of arx models. *ACC 2016*, 38–41.

Ozay, N., Lagoa, C., and Sznaier, M. (2009). Robust identification of switched affine systems via moments-based convex optimization. In *Proc. of the 48h IEEE Conf. on Decision and Control (CDC)*, 4686–4691.

Paoletti, S., Juloski, A.L., Ferrari-Trecate, G., and Vidal, R. (2007). Identification of Hybrid Systems A Tutorial. *European Journal of Control*, 13(2-3), 242–260.

Saberi, I., Faghih, F., and Bavil, F.S. (2022). A passive online technique for learning hybrid automata from input/output traces. *ACM Transactions on Embedded Computing Systems*, 22(1), 9:1–9:24.

Sauer, T. (2012). *Numerical Analysis.* Pearson Education.

Soto, M.G., Henzinger, T.A., and Schilling, C. (2021). Synthesis of hybrid automata with affine dynamics from time-series data. In *Proceedings of the 24th Int. Conf. on Hybrid Systems: Computation and Control*, 1–11.

Vidal, R., Soatto, S., Yi Ma, and Sastry, S. (2003). An algebraic geometric approach to the identification of a class of linear hybrid systems. In *Conf. on Decision and Control*, volume 1, 167–172. IEEE.

Yang, X., Beg, O.A., Kenigsberg, M., and Johnson, T.T. (2022). A framework for identification and validation of affine hybrid automata from input-output traces. *ACM Transactions on Cyber-Physical Systems*, 6(2), 1–24.