Nonlinear Analysis: Hybrid Systems 36 (2020) 100882

Contents lists available at ScienceDirect

Nonlinear Analysis: Hybrid Systems

journal homepage: www.elsevier.com/locate/nahs

Data-driven switching modeling for MPC using Regression Trees and Random Forests☆

Francesco Smarra [a,*], Giovanni Domenico Di Girolamo [a], Vittorio De Iuliis [a], Achin Jain [b], Rahul Mangharam [b], Alessandro D'Innocenzo [a]

[a] Department of Information Engineering, Computer Science and Mathematics, Università degli Studi dell'Aquila, L'Aquila, Italy
[b] Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, USA

ARTICLE INFO

Article history:
Received 11 February 2019
Accepted 29 January 2020
Available online xxxx

Keywords:
Regression Trees
Random Forests
Model predictive control
Switching systems
Markov Jump Systems

ABSTRACT

Model Predictive Control is a well consolidated technique to design optimal control strategies, leveraging the capability of a mathematical model to predict a system's behavior over a time horizon. However, building physics-based models for complex large-scale systems can be cost and time prohibitive. To overcome this problem we propose a methodology to exploit machine learning techniques (i.e. Regression Trees and Random Forests) in order to build a Switching Affine dynamical model (deterministic and Markovian) of a large-scale system using historical data, and apply Model Predictive Control. A comparison with an optimal benchmark and related techniques is provided on an energy management system to validate the performance of the proposed methodology.

1. Introduction

Control of complex cyber–physical systems received an increasing attention in the last years [1]. Model Predictive Control (MPC) is a well known control strategy used to design optimal control actions to optimize a performance metric while guaranteeing a desired system behavior (i.e. reference tracking and constraints), and has been widely applied in past years to control a large variety of systems, as for example energy systems such as smart buildings, smart grids and power systems [2–7]. To provide such an optimal control strategy, MPC leverages a mathematical model to predict the system's behavior over a finite time horizon. However, creating a physics-based model for a large-scale system as the ones mentioned above is often cost and time prohibitive [8,9]. To overcome this issue a possibility is to use identification techniques to create models using historical data available from the system. Several works deal with this problem, and use both system identification from control theory and machine learning algorithms from computer science to construct models to be used for control applications. In what follows we first describe our previous work on such topic and the contributions of this paper. Then we will provide a survey of related literature on the field and illustrate the novelty of our work with respect to the state of the art.

Previous work and main contribution. To the best of the authors' knowledge, the use of regression trees with predictive control purposes has been addressed for the first time in [10], where data-driven models have been developed to enable one-step lookahead closed-loop control for the Demand-Response problem in buildings. This approach has been

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.nahs.2020.100882.

* Corresponding author.
E-mail addresses: francesco.smarra@univaq.it (F. Smarra), giovannidomenico.digirolamo@univaq.it (G.D. Di Girolamo), vittorio.deiuliis@univaq.it (V. De Iuliis), achinj@seas.upenn.edu (A. Jain), rahulm@seas.upenn.edu (R. Mangharam), alessandro.dinnocenzo@univaq.it (A. D'Innocenzo).

https://doi.org/10.1016/j.nahs.2020.100882
1751-570X/

extended in [11], where the authors proposed a regression trees and random forests-based strategy, that implements Model Predictive Control (MPC) over a horizon of arbitrary length. An application on a real structural monitoring case study has been presented in [12]. The basic idea was to bridge machine learning and control theory adapting the regression trees and random forests techniques to build control-oriented models that could be used to provide system guarantees. One of the main reasons for choosing regression trees resides in the fact that, other than providing a good model accuracy, they are by design highly interpretable, which is a fundamental desirable quality in any model. The aforementioned approaches make use of data-driven *static* models, where the input–output relation is represented by static affine functions instead of dynamical models: such modeling framework neglects the presence of the internal state evolution and loses the information of the past inputs applied to the system over the predictive horizon. We will discuss in the paper that this translates into a lack of internal consistency of the model, and in a loss of control performance.

This work extends the preliminary results in the conference paper [13]. Its contribution is to provide a novel methodology to build a Switching Affine dynamical model of a system using historical data by appropriately adapting the regression trees and random forests classical algorithms in order to implement MPC over a time horizon of arbitrary length. As we will illustrate in the paper, the knowledge of the forecast of future disturbances can greatly improve the MPC performance of our model, as shown in the building automation experimental setup we addressed in [14], where the disturbance consists of weather conditions. However, in many applications the disturbance forecast is not available. Thus, as a further contribution of this paper, we provide a methodology to build a Markov Switching Affine model that extracts the dynamics of the disturbance as a Markov Chain exploiting the historical data. The resulting model is a special case of Markov Jump Systems [15], and can be used to implement Stochastic MPC via standard algorithms [16,17]. As an added value of our methodology, while the model derived in [10,11] did not allow to formally define and characterize fundamental properties such as stability, stabilizability, controllability, etc., the models built in this paper do. In particular, there are many results in the literature that investigate such properties for Switching systems (see e.g. [18–24] and references therein), for Piecewise Affine systems (e.g. [25,26]), and for Markov Jump Systems (e.g. [15–17]). Also, the recent results in [27], which provide an algorithm to bound the Joint Spectral Radius (JSR) of an unknown switched linear system using historical data, can be used to formally characterize robustness and reliability in terms of stability of our approach.

We finally validate our approach on a benchmark consisting of a bilinear model of a building with 12 states, 4 inputs and 8 disturbances, whose parameters were identified using experiments on a building in Switzerland [28]. We will compare the performance of our techniques with a baseline method for switching ARX identification (k-LinReg) [29], with the static approaches in [11], and with an *oracle* consisting on the perfect knowledge of both the building bilinear model in [28] and the future disturbance variables (i.e. perfect weather forecast). Simulations show that our approach outperforms both (i) the k-LinReg methodology from [29] in terms of model accuracy, and (ii) the predictive control methodology from [11] in terms of model accuracy and control performance (providing results that are much closer to the *oracle*).

Paper organization. Section 2 offers a survey of classical and recent System Identification and Machine Learning related techniques, and illustrate the novelty of our work with respect to the state of the art. In Section 3 we define our problem formulation. In Section 4 we briefly recall the technique developed in [11]. In Sections 5 and 6, respectively, we present a novel methodology to derive, starting from a set of historical data, Switched Affine (SA) and Markov Switching Affine (MSA) dynamical models using both Regression Trees (RT) and Random Forests (RF), and setup a Model Predictive Control problem leveraging such modeling frameworks. In Section 7 we provide a comparison of our methodology with the techniques in [11] and [29], as well as with an MPC *oracle* on a building automation benchmark [28].

Notation. We denote by I and $\mathbf{0}$ respectively the identity matrix and a matrix with all entries equal to 0 of appropriate dimensions. Given a matrix $A = [a_{ij}]$, we denote by a_{ij} the matrix entry at row i, and column j and by $det(A)$ its determinant. Given a set $\mathcal{S}$ we denote by $|\mathcal{S}|$ its cardinality.

2. Related work in system identification and machine learning

In this Section, with no claim of being neither exhaustive nor complete, we aim to survey some of the most important results concerning the estimation of dynamical models with control purposes, starting with a system theoretical perspective, and then moving towards recent results on machine learning applications. Since the focus of our work is on deterministic and Markovian stochastic Switching Affine models, fitted in order to readily implement Model Predictive Control strategies, we will devote a special focus on results involving the estimation of these particular classes of dynamical models.

Hybrid and Switching System Identification. As Lennart Ljung points out in his insightful paper [30], "System Identification is the art and science of building mathematical models from observed input–output data". This definition well summarizes the essence of this discipline, i.e. its mathematical rigor and its need for an imaginative kind of handicraft in understanding which particular technique or model has to be chosen to solve a given problem. System identification is a longstanding branch of Systems and Control, dating back at least to 1956, when Zadeh first used the term "Identification" in its classic paper [31]. Nevertheless, many mathematical tools used in system identification date back to decades or even centuries before the actual development of systems and control theory (just think of maximum likelihood estimators and least squares methods). More than sixty years of research in the topic have provided powerful tools to estimate dynamical

models of growing complexity. There has been a point in the 1990s, in which one would have thought that – at least for linear systems – system identification was a totally established discipline (a technology, one could say), and really no big breakthroughs were expected. Then in few years two of them came into play: subspace methods and machine learning tools, both shedding new lights on classic problems (such as linear and nonlinear identification) and recently tackled ones (such as hybrid systems identification). Clearly, surveying the contributions of nearly three decades of research is out of the scope of this section, but the interested reader is referred to classic and recent monographs such as [32–35]. Indeed, for what concerns machine learning methods applied to system identification, much of the early work has dealt with neural networks approaches to nonlinear systems (see e.g.[36]); then, a number of works on support vector machines appeared focused on both linear and some classes of nonlinear systems (see e.g. [37,38]). These works suggested to revisit kernel methods for classic and new problems in identification, which eventually led to a general and rigorous study on the links between the fields of machine leaning, system identification and function estimation, as recollected by the seminal survey [39]. In recent years, also Reinforcement Learning (RL) techniques have been exploited in data-driven approaches to control. We will cover the research in this field in the next subsection.

For what concerns the framework of hybrid systems the identification community has mainly focused on switching systems [35], strongly related to the results of this paper. Among these, mainly Switching Affine and Piecewise Affine systems have received an enduring interest due to their approximation capabilities, both in input–output and state-space representations (see again [35], which offers a comprehensive overview). It is indeed the capability of arbitrarily approximating nonlinearities that justifies the adoption of switching systems as a modeling (and estimating) class for complex and large-scale systems. This generality comes at some expense: exactly estimating such a system requires both solving a classification problem to assign data points to regions (modes) and thus determine the switching rule, and a parameter estimation of the local submodels operating in each mode. This gives rise to a mixed-integer optimization problem, whose NP-hardness is a well-known bottleneck [40]. Approximate (but computable) solutions have been proposed by means of relaxing the cited mixed-integer optimization problem or reformulating it in a more tractable form, usually by fixing some of the parameters in play (i.e. the number of modes or the order of the submodels). Among the main contributions in this respect, one can refer to the exact algebraic method proposed for noiseless systems by Vidal in [41], the continuous optimization problem reformulated by [42], and the bounded-error approach of [43]. A survey of many results in the field has been given in [44]. For what concerns state-space switching identification, subspace methods have been exploited in [45–48]. As opposed to most of linear identification methods, which rely on parametric approaches, nonlinear switching identification is currently being tackled in a nonparametric fashion, and kernel-based methods have shown good results (see [49] and references therein). Less attention has been devoted to the special class of Markov Jump System, whose switching between the submodels is driven by a Markov chain. Apart from the truncated maximum-likelihood approach presented in [50], the expectation–maximization scheme has proved effective in attacking the problem [51]. A number of works have exploited the Kullback–Leibler information measure [52,53]. More recently, a promising novel approach to the estimation of a broader class of jump models has been proposed in [54], where two alternating loss function minimizations are carried out to fit model parameters and assign an operating mode to each data point.

In this work, pursuing a step forward in bridging machine learning methods with control theory, we show how two influential techniques of machine learning, i.e. regression trees and random forests, can suitably be adopted in system identification to estimate a switching affine system in order to readily implement Model Predictive Control over an arbitrary predictive time horizon. The method proposed in this work belongs to the class of bounded-error techniques, since the number of modes of the system is not fixed a-priori but is decided by the regression trees algorithm in order to minimize the estimation error during the learning procedure. Nevertheless, a tuning parameter is left to the user, in that the minimum cardinality of each leaf of the tree (i.e. the minimum number of samples assigned to each mode) is a design parameter. The proposed approach is parametric, as the local submodels in each leaf are parametrized by ARX models of fixed order. However, the nuanced boundary between parametric and nonparametric methods suggests to classify it as semi-parametric, due to the fact that the selection of the number of modes is left to the algorithm. We can summarize the innovation of our techniques with respect to the state of the art widely discussed above as follows:

1. to the best of our knowledge, it is the first technique that combines classical algorithms of regression trees and random forests with ARX identification to estimate a Switching Affine model: our algorithm inherits the robustness and reliability of such classical techniques and, as shown in Section 7, outperforms a baseline approach to switching regression (the k-LinReg algorithm of [29]) in a benchmark case study;
2. apart from the structure of the system and the number of parameters (i.e. the size of the regressors), it does not rely on stronger assumptions such as the knowledge of the number of submodels of the original system or the switching time-instants;
3. it allows an intuitive and straightforward extension to the less explored class (in terms of identification literature) of Markov Jump Systems, whose estimation can offer the advantages discussed in Section 6.

Reinforcement Learning. Reinforcement Learning (RL) is a framework for experience-based autonomous learning of control policies: it is an alternative approach to MPC and, more in general, to classical control theory, with pros and cons [55–58]. Most of RL techniques are called Model Free Reinforcement Learning (MFRL), where a controller agent interacts with a system with control actions and measurements: the controller exploits ad-hoc optimization algorithms to

determine the control policies (actions) on the basis of on-the-fly interactions with the environment in order to maximize a measure of a long-term reward, without any a-priori knowledge of model, environment and reward function [59]. Although there have been remarkable results of MFRL in the most disparate fields, the known main disadvantage is that a large number of interactions between agent and environment is needed before learning and reaching the goals [60]. As a consequence there has been a growing interest in exploiting a-priori models, as they allow good solutions to be learned with less data. This paradigm, called Model Based Reinforcement Learning (MBRL), leverages prior information on the model to estimate future rewards and simulate (offline) sequences of actions without the necessity to directly actuate them in the real environment (see e.g. [61,62]). To the best of our knowledge the prior model is assumed to be given, and it generally consists of nonlinear static models such as Neural Networks [63,64], Regression Trees [65,66], Gaussian Processes [67], and so on. In [68] an exhaustive summary and performance analysis of several MBRL algorithms applied to different benchmarks is provided. MBRL has also been applied in building automation systems and energy management environments: in [69], an approach called Model-Assisted Batch Reinforcement Learning has been considered to provide data-driven control for the demand response problem in HVAC systems; in [70] the authors also used a hybrid approach where the learning algorithm is based on Q-Learning while the state estimation is performed with an Artificial Neural Network in order to reduce the complexity of the problem. However, due to the lack of standardized implementations of such recently developed algorithms, a full comparative study of standard techniques is difficult to find in literature.

When relating MBRL with the techniques of this paper, one could safely say that they are complementary: on one hand, our identification algorithm can be used as the prior model in MBRL; on the other hand, the control algorithms used in MBRL can represent an alternative to Model Predictive Control in our methodology.

Data-driven Model Predictive Control. Other approaches related to this work have been investigated in the literature addressing data-driven MPC. The authors in [71] considered a data-driven predictive control based on Neural Networks to guarantee energy saving and thermal comfort in public buildings. Neural Networks are used in the closed-loop control scheme to determine a thermal comfort index based on parameters that can be measured or estimated, but no Neural Networks-based system state dynamics are included into the optimization problem. [72] exploits a Neural Networks-based data-driven state model as a plant simulator in the MPC closed-loop optimization. However, since Neural Networks models are nonlinear, the associated MPC problem is also nonlinear. Hence, a global optimal solution cannot be guaranteed and solving of the optimization problem is computationally hard, or even prohibitive in the frequent cases when the complexity of the Neural Network is large.

3. Problem formulation

Let a dataset $\mathcal{D}$ be collected from a system, where $\mathcal{D} = \{(y(k), u(k), d(k))\}_{k=1}^{\ell}$ is a finite set of $|\mathcal{D}| = \ell$ samples from measurements of, respectively, output signals $y(k) \in \mathbb{R}^n$, control signals $u(k) \in \mathbb{R}^m$ and disturbance signals $d(k) \in \mathbb{R}^p$ of a physical system: we address in this paper the problem of identifying a black-box model with the aim of applying Model Predictive Control. As anticipated in Section 1, the idea is to derive such model by combining RT and RF theory with ARX system identification, and provide an algorithm to derive from $\mathcal{D}$ a switching affine model

$$x(k+1) = A_{\sigma(x(k),d(k))}x(k) + B_{\sigma(x(k),d(k))}u(k) + f_{\sigma(x(k),d(k))}, \tag{1}$$

where $x(k) \doteq [y^\top(k) \ \cdots \ y^\top(k-\delta_y) \ u^\top(k-1) \ \cdots \ u^\top(k-\delta_u)]^\top$ is an extended state to characterize a switching ARX model, $\sigma : \mathbb{R}^{n(\delta_y+1)+m\delta_u+p} \to \mathcal{M} \subset \mathbb{N}$ associates (via a rectangular partition, as will be illustrated in the following sections) to each pair $(x(k), d(k))$ an operating mode in a finite set $\mathcal{M}$, and δ_y, δ_u are nonnegative scalars denoting the number of autoregressive terms used for the corresponding variables. However, model (1) cannot be used directly to compute the solution of the standard formulation of N-steps linear MPC if $N > 1$: indeed, at each time k the measurements of $x(k+j)$ and $d(k+j)$ are not available for $j = 1, \dots, N-1$, thus the switching sequence $\sigma(x(k+j), d(k+j))$ is unknown for any $j = 1, \dots, N-1$. As a consequence, the MPC problem turns in this case into a Mixed Integer Linear Program (MILP), which is well known to be NP-hard. To address this problem we will derive in Section 5 a predictive model for $j = 0, \dots, N-1$ as follows:

$$x(k+j+1) = A_{\sigma_j(x(k),d(k))}x(k+j) + B_{\sigma_j(x(k),d(k))}u(k+j) + f_{\sigma_j(x(k),d(k))} \tag{2}$$

where $\sigma_j : \mathbb{R}^{n(\delta_y+1)+m\delta_u+p} \to \mathcal{M}_j \subset \mathbb{N}$. Eq. (2) can be now used in the standard formulation of N-steps linear MPC: since the switching sequence of $\sigma_j(x(k), d(k))$, being a function of $(x(k), d(k))$, is now available for any $j = 0, \dots, N-1$, the optimal solution at each step k can be computed via Quadratic Programming (QP).

In many practical cases, the knowledge/forecast at each time k of the future disturbance signal $(d(k+1), \dots, d(k+N-1))$ can greatly improve the MPC performance, as in the building automation setup we addressed in [14] where the disturbance consists of weather conditions. In Section 5 we will first show that it is straightforward to derive the predictive model (2) with $\sigma_j(x(k), d(k), \dots, d(k+j))$, $j = 0, \dots, N-1$: using the forecast $(d(k+1), \dots, d(k+N-1))$ it is possible to predict the future switching sequence and solve MPC via QP. However, if the forecast of the disturbance signal is not available, the sequence $\sigma_j(x(k), d(k), \dots, d(k+j))$, $j = 0, \dots, N-1$, can arbitrarily assume $\prod_{j=0}^{N-1} |\mathcal{M}_j|$ values, and the MPC problem turns (again) into a MILP. To address this problem, in Section 6 we will show how to derive transition probabilities to characterize the switching sequence as a Markov Chain: this makes the solution of the MPC problem computationally feasible leveraging the theory of Markov Jump Systems.

Remark 1. We remark that, apart from the predictive model structure (2), the methodology we propose in this work does not rely on stronger assumptions on the data or on the operating conditions of the unknown system, which are typically encountered in a number of related works [35]. Indeed, neither the number of modes nor the knowledge of the switching time-instants is assumed to be known.

4. Background on static modeling via regression trees and random forests

In this section we recall the concept of regression trees partitioning and the techniques described in [11], which provide a static predictive model over a finite time horizon for a given system. This will be useful to both better understand the approach we propose in this paper and compare it with the related methods discussed in the introduction. The main idea is to exploit historical data to create system models using machine learning techniques (in the specific case, regression trees and random forests), so that they can be used in an MPC problem formulation.

Let a dataset $\mathcal{D} = \{(y(k), u(k), d(k))\}_{k=1}^{\ell}$ be given as defined in Section 3. Let us assume, without loss of generality and for simplicity of presentation, that we wish to predict the value of the scalar variable $y_1(k+1)$ given measurements at time k of the vector $[y^\top(k)\ u^\top(k)\ d^\top(k)]^\top \in \mathbb{R}^{n+m+p}$. The RT algorithm creates, following specific optimization rules [73] (see the Appendix for some details), a tree structure $\mathcal{T}$ by partitioning the set $\mathcal{D}$ into subsets $\mathcal{D}_i$, $i = 1, \dots, q$ (the q *leaves* of the tree): each leaf i is associated to a hyperrectangle $R_i \subset \mathbb{R}^{n+m+p}$ as for example in Fig. 9 in Appendix. Let $|\mathcal{T}|$ denote the number of leaves obtained from the partitioning, then the sets R_i, $i = 1, \dots, |\mathcal{T}|$ are a partition of $\mathbb{R}^{n+m+p}$, and each leaf i contains a certain number of samples from $\mathcal{D}$ belonging to R_i, i.e. $\mathcal{D}_i = \{(y(k_1), u(k_1), d(k_1)), \dots, (y(k_\epsilon), u(k_\epsilon), d(k_\epsilon))\}$, at time instants $k_1, \dots, k_\epsilon$ that are not necessarily adjacent. The algorithm associates to each leaf $\mathcal{D}_i$ a prediction

$$\hat{y}_{1,i}(k+1) = \frac{\sum_{(y(k),u(k),d(k))\in\mathcal{D}_i} y_1(k+1)}{|\mathcal{D}_i|} \tag{3}$$

as the average of the response values associated to each sample in $\mathcal{D}_i$. This algorithm is known as CART (see [73] for more details). In run-time, once the tree is created, and given a real-time measurement $(y(k), u(k), d(k)) \in R_i$ belonging to leaf i, the CART algorithm provides as the prediction of $y_{1,i}(k+1)$ the value $\hat{y}_{1,i}(k+1)$. It is well known that the price to pay for its simplicity is related to the variance and overfitting issues the regression trees method suffers of. To address this problem *ensemble methods*, such as Random Forests (RF) [74], have been introduced in the past years. The basic idea of the random forests algorithm is to grow multiple trees, indeed a forest, considering different random subsets of the dataset to train each tree. The prediction is given by averaging the response of all the trees in the forest [74]: this (generally) reduces the overall variance in the prediction and mitigates the effect of the overfitting. The price to pay for such robustness is that RF suffer a tremendous increase in the computational complexity, depending on the number of trees of the forest. Moreover, the random choice of subsets in the trees generation does not deterministically provide the optimal predictions, as instead done by the CART algorithm, and in fact, in some rare cases, RF can even provide worse prediction accuracy than RT.

Since the prediction provided by RT and RF is an averaged value obtained through (3) it cannot be used to setup an MPC problem formulation, thus the learning procedure needs to be modified. We recall in the next subsections the approaches taken in [11,14] to address this issue respectively for RT and RF. In order to predict any variable of the vector $y = [y_1\ \cdots\ y_n]^\top$ the same arguments can be easily generalized considering multiple trees for multiple outputs (as shown in [11,14]) and/or multi-output trees (as shown in [75]).

4.1. Static modeling with Regression Trees (Static-RT, [11])

Let us consider a dataset $\mathcal{D} = \{(y(k), u(k), d(k))\}_{k=1}^{\ell}$ and let us again assume that we wish to predict the value of the scalar variable $y_1(k+1)$ given $[y^\top(k)\ u^\top(k)\ d^\top(k)]^\top \in \mathbb{R}^{n+m+p}$. For reasons related to the MPC computational complexity that will be clear at the end of this section, and are hence detailed in Remark 2, we need to partition the dataset $\mathcal{D}$ in the two disjoint sets $\mathcal{D}_c = \{u(k)\}_{k=1}^{\ell}$ of data associated to the control variables, and $\mathcal{D}_{nc} = \{(y(k), d(k))\}_{k=1}^{\ell}$ of data associated to non-control variables, and then apply the CART algorithm only on $\mathcal{D}_{nc}$. Moreover, since we wish to use our model to apply MPC with horizon N, we need a model that provides the prediction of the state over the whole horizon. To this aim, we first apply the CART algorithm to create N trees $\{\mathcal{T}_j\}_{j=0}^{N-1}$, each constructed to predict the variable $y_1(k+j+1)$, and then associate to each $y_1(k+j+1)$ a model to be used in the MPC formulation. More precisely, the training process to derive each $\mathcal{T}_j$ is divided in 2 steps, as illustrated in the left side of Fig. 1:

1. each tree $\mathcal{T}_j$ is trained applying the CART algorithm, related to the prediction of $y_1(k+j+1)$, only partitioning the set $\mathcal{D}_{nc}$ instead of $\mathcal{D}$;
2. in each leaf $\mathcal{D}_{nc,i_j}$ of each tree $\mathcal{T}_j$ we fit the following affine static model

$$\hat{y}_1(k+j+1) = \beta_{1,i_j}[1\ u^\top(k) \cdots u^\top(k+j)]^\top,\ \forall i_j,\ \forall j, \tag{4}$$

where $\beta_{1,i_j} \in \mathbb{R}^{1\times(j+1)m+1}$ is obtained by fitting the set of samples

$$\{(y(k+j+1), u(k), \dots, u(k+j)) : (y(k), d(k)) \in \mathcal{D}_{nc,i_j}\}$$

via the Least Squares method as illustrated in [11], and similarly to Problem 2 illustrated later on.

As discussed above, and as shown in [11], in order to predict any variable of the vector $y = [y_1 \ \cdots \ y_n]^\top$ the same arguments can be easily generalized using multiple trees, one for each component of y, so deriving

$$\hat{y}(k+j+1) = \beta_{i_j}[1 \ u^\top(k) \cdots u^\top(k+j)]^\top, \ \forall i_j, \ \forall j, \tag{5}$$

by $\beta_{i_j} = [\beta_{1,i_j}^\top \cdots \beta_{n,i_j}^\top]^\top \in \mathbb{R}^{n\times(j+1)m+1}$. From now on, for ease of reading, we remove the "hat" from the estimated model variables such as $\hat{y}$ in (4) and (5). The difference with the measured variables y in the dataset will be clear from the context. Consider now the following MPC problem:

Problem 1.

$$\begin{aligned}
\underset{u}{\text{minimize}} \quad & y_{k+N}^\top Q_N y_{k+N} + \sum_{j=0}^{N-1} \left(y_{k+j}^\top Q y_{k+j} + u_{k+j}^\top R u_{k+j} \right), \\
\text{subject to} \quad & y_{k+j+1} = \beta_{i_j}[1 \ u_k^\top \cdots u_{k+j}^\top]^\top \\
& y_{k+j} \in \mathcal{O} \\
& u_{k+j} \in \mathcal{U} \\
& y_{k+N} \in \mathcal{O}_N \\
& y_k = y(k), \ j = 0, \ldots, N-1,
\end{aligned}$$

where $\mathcal{O}, \mathcal{U}, \mathcal{O}_N$ are polyhedra that specify the variables constraints. At each step k of the run-time solution of Problem 1, we can use the current output and disturbance measurements $(y(k), d(k))$ to narrow down exactly one leaf for each tree $\mathcal{T}_j$ (namely the leaf such that $(y(k), d(k)) \in R_{i_j}$), thus obtaining the coefficients in β_{i_j}, for $j = 0, \ldots, N-1$, derived in (5) (see Algorithm 1 in [11] for details). Now the problem can be solved as in classical MPC: at each time step the optimal inputs $u_k^*, \ldots, u_{k+N-1}^*$ are computed, and only the first one is applied to the system, i.e. $u(k) = u_k^*$.

Remark 2. If we had also used in the CART algorithm the whole dataset $\mathcal{D}$ (i.e. also the input variables) to construct the tree, as illustrated in the right side of Fig. 1, the resulting model would have been computationally inconvenient for solving Problem 1. Indeed, in that case the optimization variables $u_k, \ldots, u_{k+N-1}$ must be used, together with state and disturbance measurements, to narrow down one leaf for each tree $\mathcal{T}_j$ in order to obtain the matrices $\beta_{i_j}, j = 0, \ldots, N-1$. This makes Problem 1 a Mixed-Integer Linear Program (MILP), which is much more complex to solve from the computational point of view. Based on a tradeoff between accuracy and computational complexity, our choice to only consider outputs and disturbances to construct the tree makes MPC easily solvable using Quadratic Programming, and provides excellent prediction accuracy both on simulative benchmarks [11,75] and experimental setups [14]. Nevertheless, our technique would also work, with trivial changes, considering control inputs: in that case, MPC could be solved using one of the approximate sub-optimal methods present in the literature to solve MILP.

4.2. Static modeling with Random Forests (Static-RF, [14])

The goal in Static-RF is to construct a forest $\mathcal{F}_j$ (instead of a tree $\mathcal{T}_j$) for any prediction step $j = 0, \ldots, N-1$, and to replace in Problem 1 the matrices $\beta_{i_j}, j = 0, \ldots, N-1$, derived as in (4) with the following matrices Θ_{i_j} derived using the Random Forests:

$$y(k+j+1) = \Theta_{i_j}[1 \ u^\top(k) \cdots u^\top(k+j)]^\top, \ \forall i_j, \ \forall j, \tag{6}$$

where $\Theta_{i_j} \in \mathbb{R}^{n\times(j+1)m+1}$ is obtained by averaging over all the matrices β_{i_j} of all the trees of the forest $\mathcal{F}_j$.

5. Switching Affine Modeling

The static modeling framework illustrated above, although simple and easy to implement, is characterized by a main drawback: the models in the leaves are affine functions of the inputs that just provide input–output *static* relations as described in (5), thus an internal state evolution of the system, which is a fundamental characteristic in control systems theory, is not considered. The main contribution of this paper is to address this issue providing a dynamical modeling framework leveraging RT and RF. To this aim we start from the techniques introduced in Section 4, i.e. the training set splitting, and replace the affine static models (5) with the following predictive model $\forall j = 0, \ldots, N-1$

$$x(k+j+1) = A_{\sigma_j(x(k),d(k))}x(k+j) + B_{\sigma_j(x(k),d(k))}u(k+j) + f_{\sigma_j(x(k),d(k))}, \tag{7}$$

where $x(k) \doteq [y^\top(k) \ \cdots \ y^\top(k-\delta_y) \ u^\top(k-1) \ \cdots \ u^\top(k-\delta_u)]^\top$ is an extended state to characterize a switching ARX model, and $\sigma_j : \mathbb{R}^{n(\delta_y+1)+m\delta_u+p} \to \{1, \ldots, |\mathcal{T}_j|\}$ associates to each pair $(x(k), d(k))$ one leaf of $\mathcal{T}_j$. To this aim we derive, in Sections 5.1 and 5.2, an algorithm to construct a model as in (7) using respectively RT and RF, and finally setup the MPC problem.

5.1. Switching Affine modeling with Regression Trees (SA-RT)

As a first step to create a switching ARX model as discussed above we need to construct an extended dataset starting from $\mathcal{D} = \{(y(k), u(k), d(k))\}_{k=1}^{\ell}$: we define $\mathcal{X} = \{(x(k), u(k), d(k))\}_{k=1}^{\ell}$, where $x(k) \doteq [y^\top(k) \ \cdots \ y^\top(k-\delta_y) \ u^\top(k-1) \ \cdots \ u^\top(k-\delta_u)]^\top$. Following the same idea of Section 4.2, and given a finite arbitrary predictive horizon N, we create nN trees $\mathcal{T}_{\iota,j}$, $\iota = 1, \ldots, n, \ j = 0, \ldots, N-1$ as illustrated in Section 4.2, each to predict $x_\iota(k+j+1)$. For each leaf $i_{\iota,j}$ of tree $\mathcal{T}_{\iota,j}$ we consider the samples associated to the time instants $k_1, \ldots, k_\epsilon$ (that are not necessarily adjacent), such that $\{(x(k_1), d(k_1)), \ldots, (x(k_\epsilon), d(k_\epsilon))\}$ are all contained in the partition $\mathcal{X}_{i_{\iota,j}}$, and solve the following least squares problem:

Problem 2.

$$\begin{aligned}
&\underset{\xi_{i_{\iota,j}}}{\text{minimize}} \quad \| \ \Lambda_{i_{\iota,j}} \xi_{i_{\iota,j}} - \lambda_{i_{\iota,j}} \ \|_2^2 \\
&\text{subject to } \Gamma_{eq} \, \xi_{i_{\iota,j}} = \gamma_{eq} && (8)\\
&\qquad\quad \Gamma_{ineq} \, \xi_{i_{\iota,j}} \leq \gamma_{ineq}, && (9)
\end{aligned}$$

where

$$\Lambda_{i_{\iota,j}} = \begin{bmatrix}
1 & \cdots & 1 \\
y_1(k_1) & \cdots & y_1(k_\epsilon) \\
\vdots & & \vdots \\
y_n(k_1) & \cdots & y_n(k_\epsilon) \\
\vdots & & \vdots \\
y_1(k_1-\delta_x) & \cdots & y_1(k_\epsilon-\delta_x) \\
\vdots & & \vdots \\
y_n(k_1-\delta_x) & \cdots & y_n(k_\epsilon-\delta_x) \\
u_1(k_1-1) & \cdots & u_1(k_\epsilon-1) \\
\vdots & & \vdots \\
u_m(k_1-1) & \cdots & u_m(k_\epsilon-1) \\
\vdots & & \vdots \\
u_1(k_1-\delta_u) & \cdots & u_1(k_\epsilon-\delta_u) \\
\vdots & & \vdots \\
u_m(k_1-\delta_u) & \cdots & u_m(k_\epsilon-\delta_u) \\
u_1(k_1+j) & \cdots & u_1(k_\epsilon+j) \\
\vdots & & \vdots \\
u_m(k_1+j) & \cdots & u_m(k_\epsilon+j) \\
\vdots & & \vdots \\
u_1(k_1) & \cdots & u_1(k_\epsilon) \\
\vdots & & \vdots \\
u_m(k_1) & \cdots & u_m(k_\epsilon)
\end{bmatrix}^\top, \quad \xi_{i_{\iota,j}} = \begin{bmatrix}
f \\
a_{1,1} \\
\vdots \\
a_{n,1} \\
\vdots \\
a_{1,\delta_x+1} \\
\vdots \\
a_{n,\delta_x+1} \\
a_{1,\delta_x+2} \\
\vdots \\
a_{m,\delta_x+2} \\
\vdots \\
a_{1,\delta_x+\delta_u+1} \\
\vdots \\
a_{m,\delta_x+\delta_u+1} \\
b_{1,j+1} \\
\vdots \\
b_{m,j+1} \\
\vdots \\
b_{1,1} \\
\vdots \\
b_{m,1}
\end{bmatrix}, \quad \lambda_{i_{\iota,j}} = \begin{bmatrix}
y_\iota(k_1+j+1) \\
y_\iota(k_2+j+1) \\
\vdots \\
y_\iota(k_\epsilon+j+1)
\end{bmatrix}. \qquad (10)$$

Once $\xi_{i_{\iota,j}}$ is computed, we can associate to each leaf the prediction

$$y_\iota(k+j+1) = [x^\top(k) \ u^\top(k) \ \cdots \ u^\top(k+j)] \ \xi_{i_{\iota,j}}. \qquad (11)$$

Remark 3. The linear equality (8) and inequality (9) can be used to constrain elements in $\xi_{i_{\iota,j}}$ due to additional information on the plant's dynamics induced by e.g. physical modeling and/or constraints, as we will illustrate in Section 7.

For the sake of simplicity we impose $\delta_u = 0$, however the result can be easily extended to the case of $\delta_u > 0$. This choice also comes from experimental findings where we noticed that the contribution in terms of model accuracy introduced by the regressive terms of the inputs is negligible, since they are already considered in the tree structure. Fixed a prediction horizon j, in order to derive a model for the whole vector $y(k+j+1)$ we can easily construct the following model by

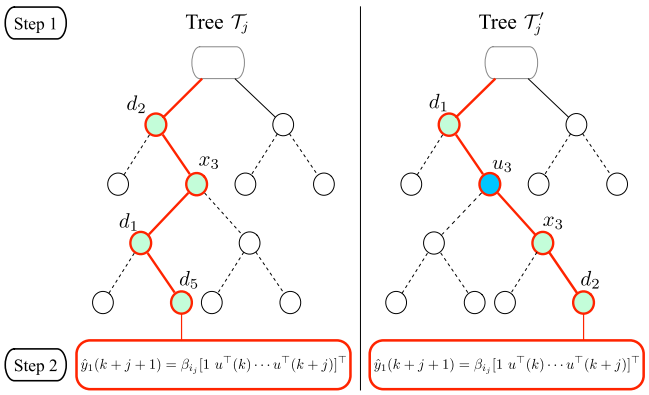


Fig. 1. *Step 1:* Tree $\mathcal{T}_j$ is trained splitting only on the set $\mathcal{D}_{nc}$. Tree $\mathcal{T}'_j$ uses both the set $\mathcal{D}_{nc}$ and the set $\mathcal{D}_c$ for splitting, and is thus computationally unsuitable for control, as we will illustrate in Remark 2. *Step 2:* In each leaf $\mathcal{D}_{nc,i_j}$ of tree $\mathcal{T}_j$, a linear regression model parametrized by β_{i_j} is computed.

combining the values $\xi_{i_{\iota,j}}$ for each component $\iota = 1, \dots, n$:

$$x(k+j+1) = A'_{i_j} x(k) + \sum_{\alpha=0}^{j} B'_{i_j,\alpha} u(k+\alpha) + f'_{i_j}, \tag{12}$$

where matrices A'_{i_j}, $B'_{i_j,\alpha}$ and f'_{i_j} have the following block form

$$A'_{i_j} = \begin{bmatrix} A'_1 & A'_2 & \cdots & A'_{\delta_x} & A'_{\delta_x+1} \\ I & 0 & \cdots & 0 & 0 \\ 0 & I & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I & 0 \end{bmatrix}, \quad B'_{i_j,\alpha} = \begin{bmatrix} b'_{1,\alpha} & \cdots & b'_{m,\alpha} \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}, \quad f'_{i_j} = \begin{bmatrix} f' \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{13}$$

where $A'_1, \dots, A'_{\delta_x+1}$, $b'_{1,\alpha}, \dots, b'_{m,\alpha}$ and f' respectively consist of the coefficients $a_{\cdot,\cdot}$, $b_{\cdot,\cdot}$ and f of $\xi_{i_{\iota,j}}$, $\iota = 1, \dots, n$. The whole procedure is summarized in Algorithm 1.

We will show in the following how, given a model as in (12) obtained from Algorithm 1, it is possible to construct a model as in (7) that is equivalent to it for any initial condition, any switching sequence and any control sequence.

Let us define an extended state $x_e(k+j) = \left[\bar{x}^\top(k+j)\ u^\top(k+j-N+1)\ \cdots\ u^\top(k+j-1)\right]^\top$.

Proposition 1. *Let A'_{i_j}, $B'_{i_j,\alpha}$ and f'_{i_j}, $\alpha = 0, \dots, j$, $j = 0, \dots, N-1$, be given as output of Algorithm 1. If A'_{i_j} is invertible for $j = 0, \dots, N-1$, then there exists a model in the form*

$$x_e(k+j+1) = A_{i_{j-1},i_j} x_e(k+j) + B_{i_{j-1},i_j} u(k+j) + f_{i_{j-1},i_j}$$

such that for any initial condition $\bar{x}(k) = x(k) = x_k$, any switching sequence $i_0, \dots, i_{N-1}$ and any control sequence $u(k), \dots, u(k+N-1)$, then $\bar{x}(k+j+1) = x(k+j+1)$, $\forall j = 0, \dots, N-1$.

Proof. For $j = 0$ we can set

$$x(k+1) = A'_{i_0} x(k) + B'_{i_0,0} u(k) + f'_{i_0} \doteq \bar{A}_{i_{-1},i_0} x(k) + \bar{B}_{i_{-1},i_0} u(k) + \bar{f}_{i_{-1},i_0}.$$

If A'_{i_0} is invertible,

$$x(k) = (A'_{i_0})^{-1}(x(k+1) - B'_{i_0,0} u(k) - f'_{i_0}).$$

For $j = 1$,

$$\begin{aligned} x(k+2) =& A'_{i_1} x(k) + B'_{i_1,0} u(k) + B'_{i_1,1} u(k+1) + f'_{i_1} \\ =& A'_{i_1}(A'_{i_0})^{-1}(x(k+1) - B'_{i_0,0} u(k) - f'_{i_0}) + B'_{i_1,0} u(k) + B'_{i_1,1} u(k+1) + f'_{i_1} \\ \doteq& \bar{A}_{i_0,i_1} x(k+1) + \bar{B}_{-1,1} u(k) + \bar{B}_{i_0,i_1} u(k+1) + \bar{f}_{i_0,i_1}. \end{aligned}$$

Note that $x_e(k+2) = [y^\top(k+2)\ \cdots\ y^\top(k+2-\delta_y)\ u^\top(k+2-N+1)\ \cdots\ u^\top(k+1)]^\top$. Thus, the component $u(k)$ belongs to $x_e(k+2)$: therefore, the term $\bar{A}_{i_0,i_1}$ and $\bar{B}_{-1,1} = (B'_{i_1,0} - A'_{i_1}(A'_{i_0})^{-1} B'_{i_0,0})$ can be embedded in A_{i_0,i_1}.

By iteration, at any step $j = 0, \dots, N-1$

$$\begin{aligned} x(k+j+1) =& A'_{i_j} x(k) + \sum_{\alpha=0}^{j} B'_{i_j,\alpha} u(k+\alpha) + f'_{i_j} \\ =& A'_{i_j} (A'_{i_{j-1}})^{-1} \left(x(k+j) - \sum_{\alpha=0}^{j-1} B'_{i_{j-1},\alpha} u(k+\alpha) - f'_{i_{j-1}} \right) + \sum_{\alpha=0}^{j} B'_{i_j,\alpha} u(k+\alpha) + f'_{i_j} \\ \doteq& \bar{A}_{i_{j-1},i_j} x(k+j) + \sum_{\nu=1}^{j-1} \bar{B}_{-\nu,j-1} u(k+\nu-1) + \bar{B}_{i_{j-1},i_j} u(k+j) + \bar{f}_{i_{j-1},i_j}. \end{aligned}$$

Since the components $u(k+j-N+1), \dots, u(k+j-1)$ belong to $x_e(k+j)$, the terms $\bar{A}_{i_{j-1},i_j}$ and $\bar{B}_{-\nu,j-1}$ can all be embedded in $A_{i_j,i_{j-1}}$. This concludes the proof.

Remark 4. The invertibility of the matrices A'_{i_j} constructed via Algorithm 1 is violated only if the determinant of matrix A'_{δ_x+1} in Eq. (13) is equal to zero. This can happen when the coefficients modeling the dependence of one component of $x(k)$ are all equal to zero. Since numerical approximations of estimated coefficients are involved, it never happens in practice that A'_{i_j} is not invertible. Indeed, applying our methodology on different use cases in this and previous work, it never happened that A'_{i_j} was not invertible.

The obtained model can be now used to formalize the following MPC problem.

Problem 3.

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & x_{k+N}^\top Q_N x_{k+N} + \sum_{j=0}^{N-1} \left(x_{k+j}^\top Q x_{k+j} + u_{k+j}^\top R u_{k+j} \right) \\ \text{subject to} \quad & x_{k+j+1} = A_{i_{j-1},i_j} x_{k+j} + B_{i_{j-1},i_j} u_{k+j} + f_{i_{j-1},i_j} \\ & x_{k+j} \in \mathcal{O} \\ & u_{k+j} \in \mathcal{U} \\ & x_{k+N} \in \mathcal{O}_N \\ & x_k = x_e(k), \ j = 0, \dots, N-1, \end{aligned}$$

where $\mathcal{O}, \mathcal{U}, \mathcal{O}_N$ are polyhedra that specify the variables constraints. As illustrated in Algorithm 1, at any time k we can use the measurements $(x(k), d(k))$ to determine $i_1, \dots, i_N$, hence characterizing the values $A_{i_{j-1},i_j}, B_{i_{j-1},i_j}, f_{i_{j-1},i_j}$ in Problem 3.

Remark 5. Note that, while using static models (i.e. where an internal state is not present) as in [11] it is not even possible to formally characterize important properties of the model. On the contrary such properties can be characterized and verified for model (7) using several results available in the literature, e.g. for stability [21,22] and controllability [23,24], as well as for stability and recursive feasibility related to Problem 3 [18–20]. As a consequence, a further contribution of this paper is to enable to use standard techniques for dynamical models to perform verification of important system properties.

5.2. Switching Affine modeling with Random Forests (SA-RF)

As for Section 4.2 the idea is to estimate the matrices in model (7) using Random Forests instead of Regression Trees. To this aim, let us consider nN Random Forests $\mathcal{F}_{\iota,j}$, $\iota = 1, \dots, n$, $j = 1, \dots, N$, with $|\mathcal{F}_{\iota,j}|$ the number of trees in the forest $\mathcal{F}_{\iota,j}$. We can solve Problem 2 for each leaf $\mathcal{X}_{i_\iota,j,\tau}$, $\iota = 1, \dots, n$ of each tree $\mathcal{T}_\tau$, $\tau = 1, \dots, |\mathcal{F}_{\iota,j}|$ of each forest $\mathcal{F}_{\iota,j}$, thus obtaining a vector of parameters $\tilde{\xi}_{i_\iota,j,\tau}$. The parameters to build matrices in (12) can be obtained by averaging, for each $\iota = 1, \dots, n$, $j = 0, \dots, N-1$, the parameters

$$\tilde{\xi}_{i_\iota,j} = \frac{\sum_{\tau=1}^{|\mathcal{F}_j|} \tilde{\xi}_{i_\iota,j,\tau}}{|\mathcal{F}_j|} \tag{14}$$

over all the trees of forest $\mathcal{F}_{\iota,j}$. Model (7) can be easily obtained by using $\tilde{\xi}_{i_\iota,j}$ in (14) instead of $\xi_{i_\iota,j}$ to derive (12).

Algorithm 1 Data-driven MPC with Regression Trees

DESIGN TIME: OFFLINE
INPUT: DATASET $\mathcal{X} = \{(x(k), u(k), d(k))\}_{k=1}^{\ell}$
procedure TRAINING LTI MODELS IN LEAVES
  Partition $\mathcal{X}$ in sets $\mathcal{X}_{nc}$ and $\mathcal{X}_c$;
  Build nN trees $\mathcal{T}_{i,j}$ using $\mathcal{X}_{nc}$, each to predict $y_i(k+j+1)$;
  for all $j = 0, \dots, N-1$ **do**
    for all n-uple $(i_{1,j} \in \mathcal{T}_{1,j}, \dots, i_{n,j} \in \mathcal{T}_{n,j})$ **do**
      Compute $(\xi_{i_{1,j}}, \dots, \xi_{i_{n,j}})$ by solving Problem 2 for each leaf;
    end for
    Build matrices $A'_{i_j}, B'_{i_j}, f'_{i_j}$ in (13) using $\xi_{i_{i,j}}$, $\forall\, i_j$;
  end for
  Compute matrices $A_{i_{j-1},i_j}, B_{i_{j-1},i_j}, f_{i_{j-1},i_j}$ using Proposition 1, $\forall\,(i_{j-1}, i_j)$;
end procedure

RUN TIME: ONLINE
INPUT: MATRICES $A_{i_{j-1},i_j}, B_{i_{j-1},i_j}, f_{i_{j-1},i_j}$, $\forall\,(i_{j-1}, i_j)$, CONSTRAINT SETS $\mathcal{O}$, $\mathcal{U}$, $\mathcal{O}_N$, WEIGHT MATRICES Q_N,Q,R
procedure MODEL PREDICTIVE CONTROL VIA SA-RT
  while $k \geq 0$ **do**
    for all $j = 0, \dots, N-1$ **do**
      Using $(x(k), d(k))$ narrow down each tree and determine i_j;
      Determine $A_{i_{j-1},i_j}, B_{i_{j-1},i_j}, f_{i_{j-1},i_j}$;
    end for
    Solve Problem 3 using QP to determine optimal inputs $u_k^*, \dots, u_{k+j}^*$;
    Apply the first input $u(k) = u_k^*$;
  end while
end procedure

Note: as illustrated in the proof of Proposition 1, we remark that $A_{i_{-1},i_0}, B_{i_{-1},i_0}, f_{i_{-1},i_0}$ only depend on i_0.

6. Markov Switching Affine Modeling

In many practical cases, the knowledge at each time k of the future disturbance signal $(d(k+1), \dots, d(k+N-1))$ can greatly improve the MPC performance, as in the building automation setup we addressed in [14] where the disturbance consists of weather conditions. In that case, we assumed to have knowledge of weather forecast, and used it to derive a dynamical model where the switching signal also depends on the future disturbance signal, i.e. $\sigma_j(x(k), d(k), \dots, d(k+j))$, $\forall j = 0, \dots, N-1$. In the technique described in Section 5, this can be easily done be appropriately redefining the dataset as $\mathcal{X} = \{(x(k), u(k), d(k), \dots, d(k+N-1))\}_{k=1}^{\ell}$.

However, in many applications the disturbance prediction is not available. In this case one can extract a model of the dynamics of the disturbance as a Markov Chain exploiting the historical data. In the following sections we illustrate how to modify the algorithm proposed in Section 5 to derive a Markov Switching Affine model that takes into account the probabilistic jumps between leaves due to the disturbance dynamics. The resulting model is a special case of Markov Jump Systems (since it is just defined over a finite time horizon) [15], and can be used to implement Stochastic MPC via standard algorithms [16,17].

6.1. Markov Switching Affine modeling with Regression Trees (MSA-RT)

Let us consider a predictive horizon equal to N and a dataset $\mathcal{X} = \{(x(k), u(k), d(k), \dots, d(k+N-1))\}_{k=1}^{\ell}$. Using the technique illustrated in the previous section, we create a model as in (7) using, for each predictive horizon j, the dataset $\mathcal{X}_{nc,j} = \{(x(k), d(k), \dots, d(k+j))\}_{k=1}^{\ell}$. As a consequence, the switching signal at time $k+j$ depends on the disturbance up to $k+j$, i.e. $\sigma_j(x(k), d(k), \dots, d(k+j))$. Clearly, when using such model to solve the MPC problem at time k, the switching sequence depends on future unknown disturbances. To address this problem we can derive the transition probabilities that characterize the switching signal as a Markov Chain and define the following Markov Switching Affine model:

$$x_e(k+1) = A_{\theta(k)} x_e(k) + B_{\theta(k)} u(k) + f_{\theta(k)}, \tag{15}$$

where $\theta(k)$ is a Markov Chain that drives the switching rule among the leaves. In order to define the probability distribution of $\theta(k)$ we need to compute, for any pair of trees $\mathcal{T}_j$ and $\mathcal{T}_{j+1}$, with $j = 1, \dots, N-1$, the transition probability $p_{i_j,i_{j+1}}$ from each leaf i_j of $\mathcal{T}_j$ to each leaf i_{j+1} of $\mathcal{T}_{j+1}$. Let $|\mathcal{X}_{i_j}|$ be the number of samples in $\mathcal{X}_{i_j}$, and $n(i_j, i_{j+1})$ be the number of samples $(x(k), d(k), \dots, d(k+j)) \in \mathcal{X}_{i_j}$ such that $(x(k), d(k), \dots, d(k+j+1)) \in \mathcal{X}_{i_{j+1}}$, then we define

$$p_{i_j,i_{j+1}} \doteq \frac{n(i_j, i_{j+1})}{|\mathcal{X}_{i_j}|}. \tag{16}$$

As a consequence, one can construct a transition probability matrix P as in (17)

$$P = \begin{bmatrix} 0 & P_{1,2} & 0 & \cdots & 0 \\ 0 & 0 & P_{2,3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & P_{N-1,N} \\ 0 & 0 & 0 & \cdots & I \end{bmatrix}, \tag{17}$$

where $P_{j,j+1} = [p_{i_j,i_{j+1}}]$, that characterizes the Markov Chain $\theta(k)$ with state space cardinality given by the set of all leaves of all trees, i.e. $\sum_{j=0}^{N-1} |\mathcal{T}_j|$. To fully characterize our Markov Chain we still need to define the initial state θ_0: this can only be done in run-time at any time step k using the measurement of $(x(k), d(k))$, which deterministically determines the initial leaf i_0 such that $(x(k), d(k)) \in R_{i_0}$, thus $\theta_0 \doteq i_0$.

As a consequence, the dynamical system defined in (15) is a Markov Jump System. Note that such model only provides the system's dynamics for a future horizon of N time steps, which is enough to implement an MPC with predictive horizon of N steps. For this reason the transition probabilities starting from the leaves of $\mathcal{T}_N$ are irrelevant for the solution of the MPC optimization problem, and can thus be chosen arbitrarily, as for example the identity matrix in the last row of (17). The Markov Switching Affine dynamical model (15) can be thus used to formalize the following MPC problem.

Problem 4.

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \mathbb{E}\left[x_{k+N}^\top Q_N x_{k+N} + \sum_{j=0}^{N-1} \left(x_{k+j}^\top Q x_{k+j} + u_{k+j}^\top R u_{k+j} \right) \right] \\ \text{subject to} \quad & x_{k+j+1} = A_{\theta(k+j)} x_{k+j} + B_{\theta(k+j)} u_{k+j} + f_{\theta(k+j)} \\ & x_{k+j} \in \mathcal{O} \\ & u_{k+j} \in \mathcal{U} \\ & x_{k+N} \in \mathcal{O}_N \\ & x_k = x_e(k), \ j = 0, \ldots, N-1, \end{aligned}$$

where $\mathcal{O}, \mathcal{U}, \mathcal{O}_N$ are polyhedra that specify the variables constraints. Note that, differently from the previous sections, in this case we do not need to determine the future switching sequence at step k, but only the initial state θ_0 of $\theta(k)$ and its transition probability matrix P. The solution of Problem 4 can be computed via standard algorithms [16,17].

Remark 6. As discussed in the previous section, a further contribution of this paper relies in the fact that important properties of the dynamical systems (15) can be characterized and verified using several classical techniques available in the literature, e.g. for stability, stabilizability, controllability and observability [15], as well as for stability and recursive feasibility related to Problem 4 [16,17].

Algorithm 2 Data-driven Stochastic MPC with Regression Trees

1: Design time: Offline
2: Input: dataset $\mathcal{X} = \{(x(k), u(k), d(k)), \ldots, d(k+j)\}_{k=1}^{\ell}$
3: **procedure** Training LTI models in leaves
4: Compute matrices A_{i_{j-1},i_j}, B_{i_{j-1},i_j}, f_{i_{j-1},i_j}, $\forall (i_{j-1}, i_j)$ following lines 3-13 of Algorithm 1;
5: **for all** $j = 1, \ldots, N-1$ **do**
6: Compute $p_{i_j,i_{j+1}}$ using (16) and the nN trees $\mathcal{T}_{i,j}$ built in step 4;
7: **end for**
8: Compute P using (17);
9: **end procedure**
10:
11: Run time: Online
12: Input: matrices $A_{i_{j-1},i_j}, B_{i_{j-1},i_j}, f_{i_{j-1},i_j}, \forall (i_{j-1}, i_j)$, matrix P, constraint sets $\mathcal{O}, \mathcal{U}, \mathcal{O}_N$, weight matrices Q_N, Q, R
13: **procedure** Stochastic Model Predictive Control via SA-RT
14: **while** $k \geq 0$ **do**
15: Using $(x(k), d(k))$ determine the initial state $\theta_0 = i_0$ of $\theta(k)$;
16: Using $A_{i_{j-1},i_j}, B_{i_{j-1},i_j} f_{i_{j-1},i_j}$, P, i_0 solve Problem 4 as in [16,17] to determine optimal inputs $u_k^*, \ldots, u_{k+j}^*$;
17: Apply the first input $u(k) = u_k^*$;
18: **end while**
19: **end procedure**

Note: as illustrated in the proof of Proposition 1, we remark that $A_{i_{-1},i_0}, B_{i_{-1},i_0}, f_{i_{-1},i_0}$ only depend on i_0.

6.2. Markov Switching Affine modeling with Random Forests (MSA-RF)

As for Section 5.2 the idea is to estimate the matrices and the transition probabilities in model (15) using Random Forests instead of Regression Trees, thus obtaining

$$x_e(k+1) = \tilde{A}_{\theta(k)} x_e(k) + \tilde{B}_{\theta(k)} u(k) + \tilde{f}_{\theta(k)}. \tag{18}$$

Matrices in (18) can be computed as illustrated in Section 5.2. Transition probabilities can be computed similarly to (16). More precisely, let us consider N Random Forests $\mathcal{F}_j$, $j = 1, \dots, N$. We can compute, for each pair of leaves $\mathcal{X}_{i_j,\tau}$, $\mathcal{X}_{i_{j+1},\tau'}$ of each pair of trees $\mathcal{T}_\tau$, $\tau = 1, \dots, |\mathcal{F}_j|$, of forest $\mathcal{F}_j$, and $\mathcal{T}_{\tau'}$, $\tau' = 1, \dots, |\mathcal{F}_{j+1}|$, of forest $\mathcal{F}_{j+1}$, the transition probabilities $p_{i_j,i_{j+1},\tau,\tau'}$ using (16). The transition probabilities $\tilde{p}_{i_j,i_{j+1}}$ can be obtained by averaging, for each $j = 1, \dots, N$, the parameters $p_{i_j,i_{j+1},\tau,\tau'}$:

$$p_{i_j,i_{j+1}} = \frac{\sum_{\tau=1}^{|\mathcal{F}_j|} \sum_{\tau'=1}^{|\mathcal{F}_{j+1}|} p_{i_j,i_{j+1},\tau,\tau'}}{|\mathcal{F}_j \parallel \mathcal{F}_{j+1}|} \tag{19}$$

over all the pairs of trees of forests $\mathcal{F}_j$, $\mathcal{F}_{j+1}$. MPC can be directly applied by replacing model (15) with model (18) in Problem 4.

7. Case study

For the comparison of our approach with respect to Static-RT and Static-RF techniques as well as with the k-LinReg method for switching regression of [29], we consider a bilinear building model developed at the Automatic Control Laboratory at ETH Zurich. It captures the essential dynamics governing the zone-level operation while considering the external and the internal thermal disturbances. By Swiss standards, the model used for this study is of a heavyweight construction with a high window area fraction on one facade and high internal gains due to occupancy and equipments [76]. As mentioned above, our methodology fits well for large-scale systems where identifying a physics-based mathematical model can be prohibitive. However, in order to validate our methodology, we provide a comparison with an optimal MPC benchmark considering the bilinear model for which the physics-based dynamics are known a priori. To this end, we build three types of dynamical models — static, Switching Affine and Markov Switching Affine — by generating data using the bilinear model. In the model validation procedure, we also compare the prediction accuracy of such models against a baseline method for Switched ARX identification [29], showing better performance of our methods in the considered case study. In our future work, we will consider more complex building models, for which the data can be obtained using EnergyPlus [77].

Model description. The bilinear model has 12 internal states including the inside zone temperature $\mathsf{T}_{\mathsf{in}} \in \mathbb{R}$, the slab temperatures $\mathsf{T}_{\mathsf{sb}} \in \mathbb{R}^5$, the inner wall $\mathsf{T}_{\mathsf{iw}} \in \mathbb{R}^3$ and the outside wall temperature $\mathsf{T}_{\mathsf{ow}} \in \mathbb{R}^3$. The state vector is defined as $x := [\mathsf{T}_{\mathsf{in}}\ \mathsf{T}_{\mathsf{sb}}^\top\ \mathsf{T}_{\mathsf{iw}}^\top\ \mathsf{T}_{\mathsf{ow}}^\top]^\top$. There are 4 control inputs including the blind position B, the gains due to electric lighting L, the evaporative cooling usage factor C, and the heat from the radiator H such that $u := [\mathsf{B}\ \mathsf{L}\ \mathsf{C}\ \mathsf{H}]^\top$. B and L affect both room illuminance and temperature due to heat transfer, whereas C and H affect only the temperature. The model is subject to 5 weather disturbances (see Fig. 2): solar gains with fully closed blinds Q_{sc} and with open blinds Q_{so}, daylight illuminance with open blinds I_{o}, external dry-bulb temperature T_{db} and external wet-bulb temperature T_{wb}. The hourly weather forecast, provided by MeteoSwiss, was updated every 12 hrs. Therefore, to improve the forecast, an autoregressive model of the uncertainty was considered. Other disturbances come from the internal gains due to occupancy Q_{io} and due to equipments Q_{ie} which were assumed as per the Swiss standards [78]. We define $d := [\mathsf{Q}_{\mathsf{sc}}\ \mathsf{Q}_{\mathsf{so}}\ \mathsf{I}_{\mathsf{o}}\ \mathsf{Q}_{\mathsf{io}}\ \mathsf{Q}_{\mathsf{ie}}\ \mathsf{T}_{\mathsf{db}}\ \mathsf{T}_{\mathsf{wb}}]^\top$. For further details, we refer the reader to [28]. The model dynamics are given below. The bilinearity is present in both input-state, and input-disturbance.

$$x(k+1) = Ax(k) + (B_u + B_{xu}[x_k] + B_{du}[d_k])u(k) + B_d d(k) \tag{20}$$

$$B_{xu}[x_k] = [B_{xu,1}x(k), B_{xu,2}x(k), \dots, B_{xu,4}x(k)] \tag{21}$$

$$B_{du}[d_k] = [B_{du,1}d(k), B_{du,2}d(k), \dots, B_{du,4}d(k)], \tag{22}$$

with $A \in \mathbb{R}^{12\times12}$, $B_{xu,i} \in \mathbb{R}^{12\times12}$ and $B_{du,i} \in \mathbb{R}^{12\times7}$, $\forall i = 1, 2, 3, 4$.

The solution obtained from MPC sets the optimal benchmark, since it uses the exact knowledge of the plant nonlinear dynamics and of the future disturbances. In what follows, we will call this solution the *oracle*.

Objective. We want to minimize the energy usage, i.e. $c^\top u$, while maintaining a desired level of occupant comfort. At time step k, we solve the following continuously linearized MPC problem to determine the optimal sequence of inputs u^*:

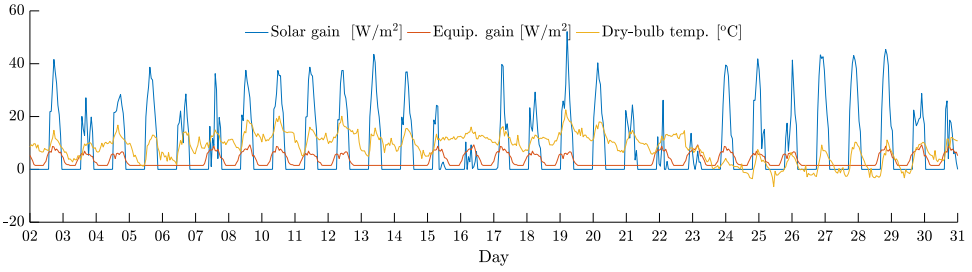


Fig. 2. An example of disturbance signals for the month of January.

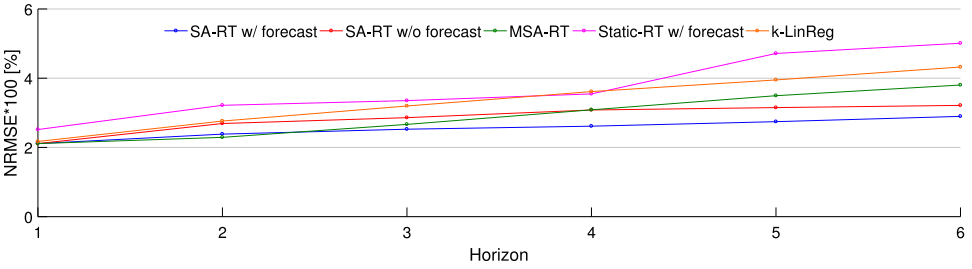


Fig. 3. Model validation of RT models for different horizons over the *testing* dataset.

Problem 5.

$$
\begin{aligned}
\underset{u}{\text{minimize}} \quad & \sum_{j=1}^{N-1} \left(\left(x_{k+j} - x_{\text{ref}}\right)^\top Q \left(x_{k+j} - x_{\text{ref}}\right) + u_{k+j-1}^\top R u_{k+j-1} + c^\top u_{k+j-1} + \lambda \varepsilon_j \right) \\
\text{subject to} \quad & x_{k+j} = A x_{k+j-1} + B u_{k+j-1} + B_d d_{k+j-1} \\
& B = B_u + B_{xu}[x_k] + B_{du}[d_{k+j-1}] \\
& x_{k+j} \in [x_{\min} - \varepsilon_j, x_{\max} + \varepsilon_j] \\
& u_{k+j-1} \in [u_{\min}, u_{\max}] \\
& \varepsilon_j \geq 0, \\
& x_k = x(k), \ j = 1, \ldots, N-1,
\end{aligned}
\tag{23}
$$

where $Q \in \mathbb{R}^{12\times 12}$ has all zero entries except for the element $(1, 1)$ associated to the zone temperature, $R \in \mathbb{R}^{4\times 4}$, $c^\top \in \mathbb{R}^{1\times 4}$ is proportional to the cost of using each actuator and λ penalizes state bound violations ε_j. At each time step, only the first optimal input of the sequence is applied to the system.

Training. The output variable for training is the inside zone temperature i.e. T_{in}. To train the trees $\mathcal{T}_{\iota,j}$ and the forests $\mathcal{F}_{\iota,j}$, we consider weather disturbances, external disturbances due to occupancy and equipments, and autoregressive terms of the inside room temperature, i.e.

$$
\mathcal{X} = \{\mathsf{T}_{\text{in}}(k), \ldots, \mathsf{T}_{\text{in}}(k - \delta_x), u(k), d(k + \bar{\delta}_d), \ldots, d(k - \underline{\delta}_d)\},
\tag{24}
$$

where δ_x and δ_d represent the orders of the auto-regressive terms, and we have chosen $\delta_u = 0$. The training dataset was generated by simulating the bilinear model with rule-based strategies for 10 months in 2007, while the testing dataset was generated for the month of January. For the training we chose $\underline{\delta}_d = N - j + 1$, $\delta_x = 6$, where N is the predictive horizon, and either $\bar{\delta}_d = j-1$ in the case with perfect forecast knowledge or $\bar{\delta}_d = 0$ in the case without forecast knowledge. Moreover, as anticipated in Section 5.1, we exploit in the identification process the linear equality (8) and inequality (9) to take into account the physical constraints that the second and fourth components of the input, i.e. the electric lighting L and the heat from the radiator H cannot be negative, and the third component of the input, i.e. the evaporative cooling usage factor C, cannot be positive. To train random forests we considered a number of trees equal to 30. In the following, for the sake of clarity, we first show simulation results obtained with regression trees, and then the ones with random forests. Finally, we provide our conclusions.

Validation using RT. In Fig. 3 we validate the prediction accuracy in the case of regression trees for horizon $N = 1, \ldots, 6$ (i.e. 6-hour ahead) for the month of January using Static-RT w/ forecast [11,75] (i.e. with perfect knowledge of future disturbance), SA-RT w/ forecast, SA-RT w/o forecast (i.e. without any knowledge of future disturbance) and MSA-RT

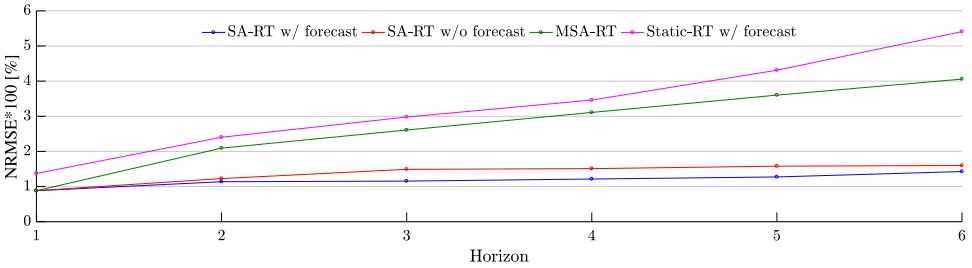


Fig. 4. Model validation of RT based models for different horizons over the *training* dataset.

(of course without any knowledge of future disturbance). We also compare the validation results with a baseline approach to switching regression, i.e. the k-LinReg algorithm proposed in [26].

The NRMSE behavior shows that the Static-RT provides less accurate prediction with respect to the SA-RT and MSA-RT techniques proposed in this paper. As expected, SA-RT w/ forecast, SA-RT w/o forecast and MSA-RT provides exactly the same prediction quality at step 1, as the future disturbance has no effect. SA-RT forecast always provides the best prediction, except for step 2 where its prediction is slightly worse than MSA-RT due to model identification uncertainties. MSA-RT works better than SA-RT w/o forecast up to step 4: this shows that the stochastic information introduced by the Markov Chain identification process improves the quality of our prediction. In particular, it is possible to notice that the MSA-RT approach is much more robust with respect to perturbations of the disturbance signal than the deterministic SA-RT approach, without increasing the computational complexity of the model generation and of the MPC solution: this is evident by comparing the validation of our prediction over the *training* (Fig. 4) and *testing* (Fig. 3) datasets. In the training case MSA-RT performs much worse than SA-RT w/o forecast because the disturbances are exactly the same used in the learning process: as a consequence a deterministic model based just on the knowledge of the current disturbance signal $d(k)$ works very well (also note that the dynamics of the system and of the disturbance, i.e. the weather, are very slow). In the testing case, being the dataset different from the one used for learning, the robustness introduced by the Markov model of the disturbance makes the MSA-RT prediction much more accurate. However, after step 4, SA-RT w/o forecast works better than MSA-RT also on the *testing* data-set: our interpretation is that, after 4 h, the prediction error introduced by the Markov Chain model grows significantly making the overall accuracy worse than the case without any knowledge of future disturbance. Also, the proposed models (both deterministic and stochastic) outperform the predictive models obtained by means of the k-LinReg method.

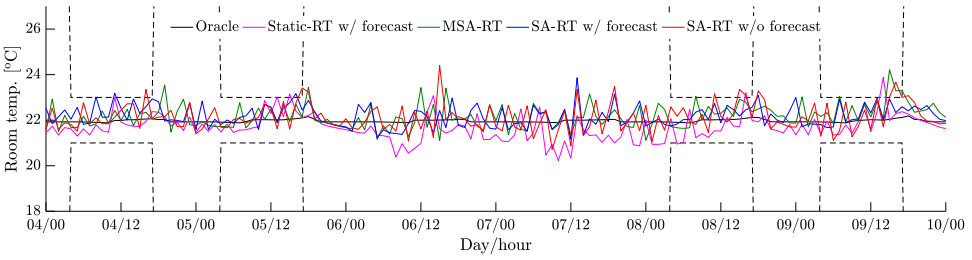
The improvements of the prediction accuracy of the proposed approaches with respect to Static-RT will be even more evident in the closed-loop simulations, as the MSA and SA models keep memory of the past applied inputs due to the presence of an internal state model.

Closed-loop simulations using RT. In Fig. 5, we simulate the closed-loop system where the controller solves the MPC Problem 5 with prediction horizon $N = 6$ (i.e. 6 h) for 5 different cases: Static-RT w/ forecast, SA-RT w/ forecast, SA-RT w/o forecast, MSA-RT, and finally the *oracle*. The performance is compared for the month of January. The cooling usage factor $\mathtt{C}$ is constrained in $[0, 1]$, the heat input in $[0, 23]$ W/m^2, and the room temperature in $[21, 23]$ °C during the day. The optimization is solved in MATLAB using CPLEX. The reference temperature x_{ref} is chosen to be 22 °C. The cost function parameters are setup as $Q(1, 1) = 10^2$, $R = diag(10^{-3})$, $\lambda = 10^3$, and $c^\top = [0,\ 3.32,\ 7.47,\ 1.107]$ as a constant cost of the electricity taken from [76].

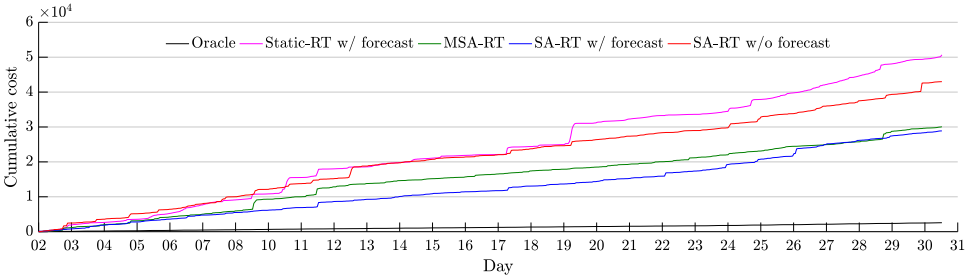
The room temperature profile is shown in Fig. 5(a). For the sake of readability we only show the controlled temperature over 6 days of January, while the cost will be provided for the whole month. The plots show that, except for few spikes, Static-RT w/ forecast generates large oscillations and bound violations, while MSA-RT, SA-RT w/o forecast and SA-RT w/ forecast are closer to the smooth temperature regulation of the *oracle*. We recall that the sampling time of our system is 1 h, thus the *spiky* temperature plot. The optimized cost function is shown in Fig. 5(b). The plots show that, SA-RT w/ forecast provides the best control performance, followed by MSA-RT, SA-RT w/o forecast and finally Static-RT w/ forecast. Note that the MSA-RT is almost as performant as the SA-RT w/ forecast. The *oracle* we compare to shows the best achievable control performance.

Validation using RF. As done for regression trees, in Fig. 6 we validate the prediction accuracy in the case of random forests for horizon $N = 1, \ldots, 6$ for the month of January using Static-RF w/ forecast [11,14], SA-RF w/ forecast, SA-RF w/o forecast and MSA-RF. We trained the forests to generate SA-RF w/ forecast, SA-RF w/o forecast and MSA-RF with 30 trees, and a forest of 75 trees to generate the Static RF model: in particular, Static RF with less than 75 trees provides very inaccurate predictions, while 30 trees are enough for obtaining a good prediction with SA-RF w/ forecast, SA-RF w/o forecast and MSA-RF.

Also in this case, the NRMSE behavior shows that the Static-RF provides (using 75 trees) less accurate prediction with respect to the other techniques (using just 30 trees). The SA-RF w/ forecast shows the best performance. All deterministic models (Static-RF w/ forecast, SA-RF w/ forecast and SA-RF w/o forecast) improve the prediction accuracy w.r.t. the corresponding RT case. On the contrary, the prediction accuracies provided by MSA-RT and MSA-RF are pretty close one to



(a) Controlled temperature from January 4th to January 10th.



(b) Cumulative cost in the objective function over January.

Fig. 5. Comparison of control performance for different modeling approaches using Regression Trees.

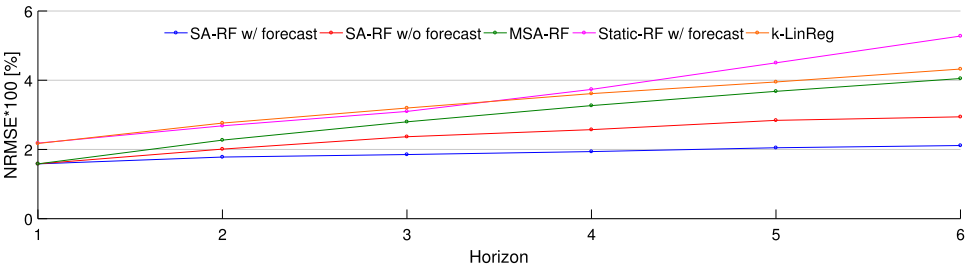


Fig. 6. Model validation of RF models for different horizons over the *testing* dataset.

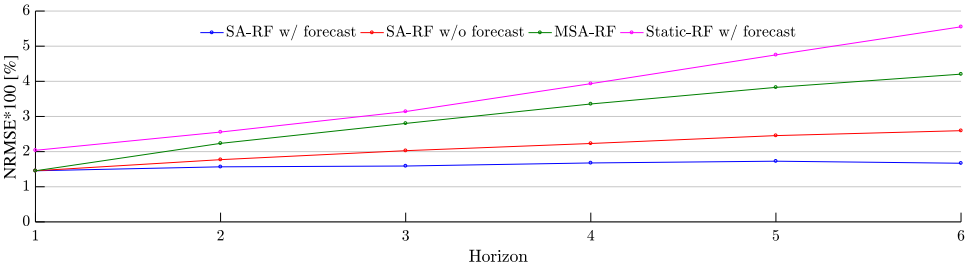
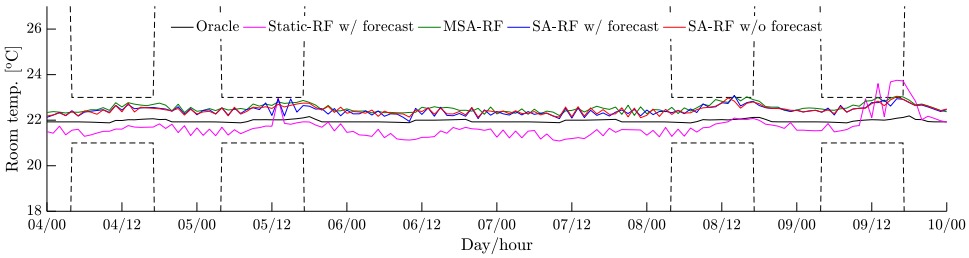
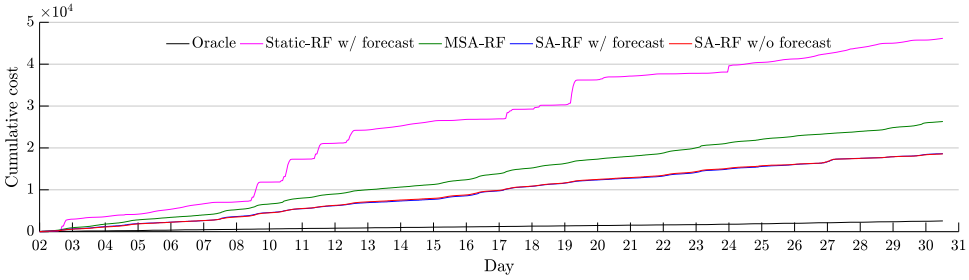


Fig. 7. Model validation of RF models for different horizons over the *training* dataset.

each other, and in particular they are both slightly worse than the accuracy provided by SA-RF w/o forecast. In our opinion, the reason behind this interesting behavior is that the robustness introduced by the random forests mitigates the effect of the Markov Chain through the 3 averaging steps in (14) and (19). Also, the fact that both temperature and disturbance dynamics are slow makes the robustness introduced by the forest a key factor to compensate the uncertainties of the disturbance. This is also the reason why, differently from the RT case, the validation of RF made on the training dataset, shown in Fig. 7, keeps the same shape as in the validation using the testing set. Also, the proposed random forests-based models outperform the k-LinReg approach in terms of model accuracy.



(a) Controlled temperature from January 4th to January 10th.



(b) Cumulative cost in the objective function over January.

Fig. 8. Comparison of control performance for different modeling approaches using Random Forests.

We can conclude that the MSA-RT approach represents, in applications where the disturbance forecast is unavailable, a good compromise with respect to RF-based approaches as it provides good prediction accuracy without the drawback due to the increase of computational complexity typical of RF.

Closed-loop simulations using RF. The closed-loop simulations using RF have been ran using same parameters as in the RT case. In Fig. 8, we show results obtained from the simulations of the closed-loop system where the controller solves the MPC Problem 5 with prediction horizon $N = 6$ (i.e. 6 hrs) for 5 different cases — Static-RF w/ forecast, SA-RF w/ forecast, SA-RF w/o forecast, MSA-RF, and finally the *oracle*. As already mentioned in the validation paragraph, we used 75 trees to train the Static-RF, and 30 trees to train the SA-RF w/ forecast, SA-RF w/o forecast and MSA-RF.

The room temperature profile is shown in Fig. 8(a), and also in this case shows a reduced period of 6 days. The first remarks are that, as expected, the robustness introduced by the randomized Random Forest algorithms generates smoother temperature plots with respect to the Regression Trees case. The plots show that Static-RF w/ forecast generates bound violations, while MSA-RF, SA-RF w/o forecast and SA-RF w/ forecast are closer to the smooth temperature regulation of the *oracle*. The optimized cost function is shown in Fig. 5(b), where as expected all RF models perform better than the corresponding RT model. The plots show that SA-RF w/ forecast and SA-RF w/o forecast perform almost equally, and provide the best control performance: our interpretation is that the robustness introduced by the forest compensates the effect of errors in disturbance prediction. The accuracy of MSA-RF follows, confirming the considerations discussed in the RF validation section. Finally, Static-RF w/ forecast provides the worst prediction accuracy. Again, the *oracle* we compare to shows the best achievable control performance.

8. Conclusions and future work

This paper provides a novel technique to identify a switching affine model from a dataset, combining regression trees and random forests with ARX system identification: this represents a further step towards bridging machine learning to control theory. Our novel modeling framework based on Switching Affine and Markov Switching Affine dynamical models allows, due to the introduction of an internal state model, to efficiently improve the prediction accuracy and control performance with respect to existing related techniques, as shown on a benchmark case study. As an additional contribution, our framework allows to formally define and characterize important properties, e.g. such as stability and stabilizability, which was not possible using the static models of our previous techniques.

In future work we plan to validate our techniques to real experimental setups. We also plan to improve the accuracy of the Markov model generation by applying optimization techniques, and validate our models on control systems where modeling a disturbance characterized by fast dynamics with a Markov Chain can be even more effective, e.g. when the disturbance is a communication channel in a networked control system.

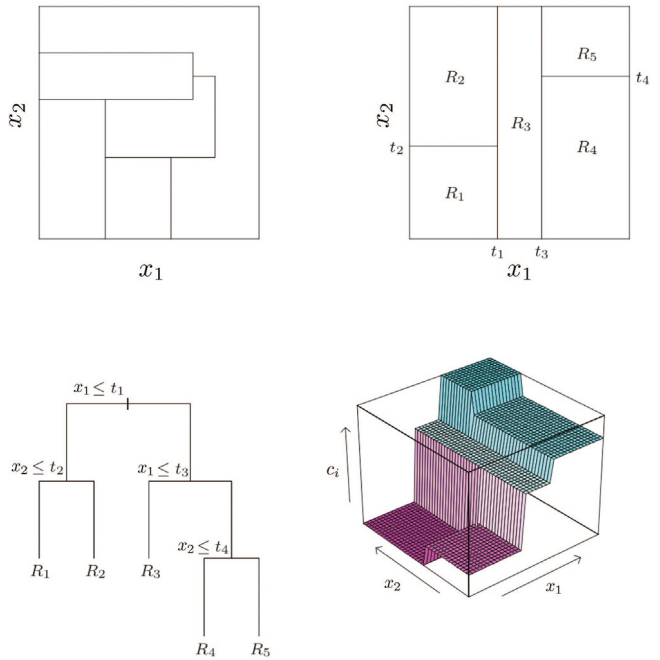


Fig. 9. Top right: dataset by recursive binary splitting. Top left: partition that cannot be obtained from recursive binary splitting. Bottom left: tree corresponding to the partition. Bottom right: perspective plot of the prediction surface.

Acknowledgments

This work was supported by the Italian Government under Cipe resolution n. 135 (Dec. 21, 2012), project *INnovating City Planning through Information and Communication Technologies* (INCIPICT), and under PON Ricerca & Innovazione, Italy 2014-2020 (AIM1877124-Attività 1); and by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO, USA and DARPA, USA.

The authors would like to thank Xiaojing Zhang for providing the bilinear building model.

Appendix

In this appendix we explain how Regression Trees are built using an example adapted from [79]. Tree-based methods partition the dataset, for example $\mathcal{D} = \{y(k), x_1(k), x_2(k)\}_{k=1}^{\ell}$, with $y, x_1, x_2 \in \mathbb{R}$, into a set of rectangles (more formally, hyperrectangles) and then fit a simple model in each one. They are conceptually simple yet powerful.

Let us consider a regression problem, where we wish to estimate the (response) variable $y(k)$ using the previous values of the (predictor) variables $x_1(k)$ and $x_2(k)$, each taking values in the unit interval.

The top left plot of Fig. 9 shows a partition of the dataset by lines that are parallel to the coordinate axes. In each partition element, we can model $y(k)$ with a different constant. However, there is a problem: although each partitioning line has a simple description like $x_1 = c \in \mathbb{R}$, some of the resulting regions are complicated to describe. To simplify things, we can restrict ourselves to only consider recursive binary partitions, like the ones shown in the top right plot of Fig. 9. We first split the set into two regions, and model the response by the mean of y in each region. We choose the variable and split-point to achieve the best prediction for y. Then one or both of these regions are split into two more regions, and this process is continued, until some stopping rule is applied. This is the "recursive partitioning" step of the algorithm. For example, in the top right plot of Fig. 9, we first split at $x_1 = t_1$. Then the region $x_1 \leq t_1$ is split at $x_2 = t_2$ and the region $x_1 > t_1$ is split at $x_1 = t_3$. Finally, the region $x_1 > t_3$ is split at $x_2 = t_4$. The result of this process is a partition of the dataset into the five regions (or leaves) $R_1, R_2, \ldots, R_5$. The corresponding regression tree model, $\mathcal{T}$, predicts y with a constant, c_i, in region R_i i.e.,

$$f_{\mathcal{T}}(x_1, x_2) = \sum_{i=1}^{5} c_i I\{(x_1, x_2) \in R_i\}, \tag{25}$$

where $I\{x \in R\}$ is the indicator function that is equal to 1 if $x \in R$, and 0 otherwise. This same model can be represented by the binary tree shown in the bottom left of Fig. 9. The full dataset sits at the top of the tree. Observations satisfying the condition at each node are assigned to the left branch, and the others to the right branch. The terminal nodes or leaves

of the tree correspond to the regions $R_1, R_2, \ldots, R_5$. The bottom right plot of Fig. 9, shows the perspective plot of the regression surface obtained as a result of building a regression tree with the 5 constants $c_i,\ i = 1, 2, 3, 4, 5$.

Node splitting criteria. Now, a first question is: *how to grow a regression tree?* Suppose our dataset $\mathcal{D} = \{y(k), x_1(k), x_2(k), \ldots, x_p(k)\}_{k=1}^{\ell}$, consisting of p predictor variables, i.e. $x_1, x_2, \ldots, x_p$, and one response variable, i.e. y. We have $|\mathcal{D}| = \ell$ observations (samples): $(x(k), y(k))$, $k = 1, 2, \ldots, \ell$, with $x(k) = [x_1(k)\, x_2(k)\ \cdots\ x_p(k)]^\top$. For regression trees we adopt the sum of squares as our splitting criteria, i.e. a variable at a node will be split if it minimizes the following sum of squares between the predicted response and the actual output variable:

$$\sum_k (y(k) - f_{\mathcal{T}}(x(k)))^2. \tag{26}$$

The best response c_i (from Eq. (25) for the partition R_i), is just the average of output samples in the region R_i, i.e.

$$c_i = avg(y(k)|x(k) \in R_i). \tag{27}$$

Finding the best binary partition in terms of minimum sum of squares is generally computationally infeasible. A greedy algorithm is used instead. Starting with all of the data, consider a splitting variable j and split point s, and define the following pair of left (R_L) and right (R_R) half-planes

$$\begin{aligned} R_L(j, s) &= \left\{x | x_j \le s\right\}, \\ R_R(j, s) &= \left\{x | x_j > s\right\} \end{aligned} \tag{28}$$

The splitting variable j and the split point s is obtained by solving the following minimization:

$$\min_{j,s} \left[\min_{c_L} \sum_{x(k)\in R_L(j,s)} (y(k) - c_L)^2 + \min_{c_R} \sum_{x(k)\in R_R(j,s)} (y(k) - c_R)^2 \right] \tag{29}$$

where, for any choice of j and s, the inner minimization in Eq. (29) is solved using

$$\begin{aligned} c_L &= \text{avg}(y(k)|x(k) \in R_L(j, s)), \\ c_R &= \text{avg}(y(k)|x(k) \in R_R(j, s)). \end{aligned} \tag{30}$$

For each splitting variable x_j, the determination of the split point s can be done very quickly and hence by scanning through all of the inputs (x_i's), the determination of the best pair (j, s) is feasible. Having found the best split, we partition the data into the two resulting regions and repeat the splitting process on each of the two regions. Then this process is repeated on all of the resulting regions.

Rather than splitting each node into just two regions at each stage, we might consider multiway splits into more than two groups. While this can sometimes be useful, it is not a good general strategy. The problem is that multiway splits fragment the data too quickly, leaving insufficient data at the next level down. Hence we would want to use such splits only when needed. Also multiway splits can be achieved by a series of binary splits.

Stopping criteria and pruning. At this point, the second question is: *How large should we grow the tree?* Every recursive algorithm needs to know when it is done, i.e. it requires a stopping criteria. For regression trees this means when to stop splitting the nodes. A very large tree might overfit the data, while a small tree might not capture the important structure. Tree size is a tuning parameter governing the model's complexity, and the optimal tree size should be adaptively chosen from the data. One approach is to split tree nodes only if the decrease in sum-of-squares due to the split exceeds some threshold. However, this strategy is myopic, since a seemingly worthless split might lead to a very good split below it. A preferred strategy is to grow a large tree, stopping the splitting process only when some minimum number of data points at a node (`MinLeaf`) is reached. Then this large tree is pruned using cost-complexity pruning methods.

Define a subtree $\mathcal{T}_{sub} \subset \mathcal{T}$ to be any tree that can be obtained by pruning $\mathcal{T}$, i.e. collapsing any number of its non-terminal nodes. Let node i corresponding to the partition R_i. $|\mathcal{T}_{sub}|$ denotes the number of terminal nodes in $\mathcal{T}_{sub}$ Define,

$$\begin{aligned} N_i &= \#\left\{x(k) \in R_i\right\}, \\ c_i &= \frac{1}{N_i} \sum_{x(k)\in R_i} y(k), \\ Q_i(T) &= \frac{1}{N_i} \sum_{x(k)\in R_i} (y(k) - c_i)^2 \end{aligned} \tag{31}$$

where N_i is the number of samples in the partition R_i, c_i is the estimate of y within R_i and $Q_i(T)$ is the mean square error of the estimate c_i. The cost complexity criteria is then defined as:

$$C_\alpha(\mathcal{T}_{sub}) = \sum_{i=1}^{|\mathcal{T}_{sub}|} N_i Q_i(T) + \alpha|\mathcal{T}_{sub}| \tag{32}$$

The goal is to find, for each α, the subtree $\mathcal{T}_\alpha \subset \mathcal{T}$ to minimize $C_\alpha(\mathcal{T}_{sub})$. The tuning parameter $\alpha \geq 0$ governs the trade off between tree size and its goodness of fit to the data. For each α one can show that there is a unique smallest subtree $\mathcal{T}_\alpha$ that minimizes $C_\alpha(\mathcal{T}_{sub})$ [80]. Estimation of α is achieved by cross-validation.

References

[1] Y.Z. Lun, A. D'Innocenzo, F. Smarra, I. Malavolta, M.D. Di Benedetto, State of the art of cyber-physical systems security: An automatic control perspective, J. Syst. Softw. 149 (2019) 174–216.
[2] Y. Ma, J. Matusko, F. Borrelli, Stochastic model predictive control for building HVAC systems: complexity and conservatism, IEEE Trans. Control Syst. Technol. 23 (1) (2015) 101–116.
[3] F. Oldewurtel, A. Parisio, C.N. Jones, D. Gyalistras, M. Gwerder, V. Stauch, B. Lehmann, M. Morari, Use of model predictive control and weather forecasts for energy efficient building climate control, Energy Build. 45 (2012) 15–27.
[4] M. Maasoumy, M. Razmara, M. Shahbakhti, A.S. Vincentelli, Handling model uncertainty in model predictive control for energy efficient buildings, Energy Build. 77 (2014) 377–392.
[5] A. Iovine, T. Rigaut, G. Damm, E. De Santis, M.D. Di Benedetto, Power management for a DC microgrid integrating renewables and storages, Control Eng. Pract. 85 (2019) 59–79.
[6] A.N. Venkat, I.A. Hiskens, J.B. Rawlings, S.J. Wright, Distributed MPC strategies with application to power system automatic generation control, IEEE Trans. Control Syst. Technol. 16 (6) (2008) 1192–1206.
[7] F. Kennel, D. Gorges, S. Liu, Energy management for smart grids with electric vehicles based on hierarchical MPC, IEEE Trans. Ind. Inform. 9 (3) (2013) 1528–1537.
[8] D. Sturzenegger, D. Gyalistras, M. Morari, R.S. Smith, Model predictive climate control of a swiss office building: Implementation, results, and cost–benefit analysis, IEEE Trans. Control Syst. Technol. 24 (1) (2016) 1–12.
[9] E. Žáčeková, Z. Váňa, J. Cigler, Towards the real-life implementation of MPC for an office building: Identification issues, Appl. Energy 135 (2014) 53–62.
[10] M. Behl, F. Smarra, R. Mangharam, DR-Advisor: A data-driven demand response recommender system, Appl. Energy 170 (2016) 30–46.
[11] A. Jain, F. Smarra, R. Mangharam, Data predictive control using regression trees and ensemble learning, in: Decision and Control (CDC), 2017 IEEE 56th Annual Conference on, IEEE, 2017, pp. 4446–4451.
[12] G. Di Girolamo, F. Smarra, V. Gattulli, F. Potenza, F. Graziosi, A. D'Innocenzo, Data-driven optimal predictive control of seismic induced vibrations in frame structures, Struct. Control Health Monit. (2020).
[13] F. Smarra, A. Jain, R. Mangharam, A. D'Innocenzo, Data-driven switched affine modeling for model predictive control, in: IFAC Conference on Analysis and Design of Hybrid Systems, ADHS'18, IFAC, 2018, pp. 199–204.
[14] F. Smarra, A. Jain, T. de Rubeis, D. Ambrosini, A. D'Innocenzo, R. Mangharam, Data-driven model predictive control using random forests for building energy optimization and climate control, Appl. Energy (2018).
[15] O.L.V. Costa, M.D. Fragoso, R.P. Marques, Discrete-Time Markov Jump Linear Systems, Springer Science & Business Media, 2006.
[16] D. Bernardini, A. Bemporad, Stabilizing model predictive control of stochastic constrained linear systems, IEEE Trans. Automat. Control 57 (6) (2012) 1468–1480.
[17] P. Patrinos, P. Sopasakis, H. Sarimveis, A. Bemporad, Stochastic model predictive control for constrained discrete-time Markovian switching systems, Automatica 50 (10) (2014) 2504–2514.
[18] M.A. Müller, P. Martius, F. Allgöwer, Model predictive control of switched nonlinear systems under average dwell-time, J. Process Control 22 (9) (2012) 1702–1710.
[19] L. Zhang, S. Zhuang, R.D. Braatz, Switched model predictive control of switched linear systems: Feasibility, stability and robustness, Automatica 67 (2016) 8–21.
[20] L. Bridgeman, C. Danielson, S. Di Cairano, Stability and feasibility of MPC for switched linear systems with dwell-time constraints, in: American Control Conference, ACC, 2016.
[21] J.P. Hespanha, Uniform stability of switched linear systems: Extensions of LaSalle's invariance principle, IEEE Trans. Automat. Control 49 (4) (2004) 470–482.
[22] D. Liberzon, J.P. Hespanha, A.S. Morse, Stability of switched systems: a Lie-algebraic condition, Systems Control Lett. 37 (3) (1999) 117–122.
[23] G. Xie, D. Zheng, L. Wang, Controllability of switched linear systems, IEEE Trans. Automat. Control 47 (8) (2002) 1401–1405.
[24] Z. Sun, S.S. Ge, T.H. Lee, Controllability and reachability criteria for switched linear systems, Automatica 38 (5) (2002) 775–786.
[25] M. Lazar, W. Heemels, S. Weiland, A. Bemporad, Stabilizing model predictive control of hybrid systems, IEEE Trans. Automat. Control 51 (11) (2006) 1813–1818.
[26] A. Bemporad, M. Morari, Control of systems integrating logic, dynamics, and constraints, Automatica 35 (3) (1999) 407–427.
[27] J. Kenanian, A. Balkan, R.M. Jungers, P. Tabuada, Data driven stability analysis of black-box switched linear systems, Automatica 109 (2019) 108533.
[28] F. Oldewurtel, Stochastic Model Predictive Control for Energy Efficient Building Climate Control (Ph.D. thesis), ETH Zurich, 2011.
[29] F. Lauer, Estimating the probability of success of a simple algorithm for switched linear regression, Nonlinear Anal. Hybrid Syst. 8 (2013) 31–47.
[30] L. Ljung, Perspectives on system identification, Annu. Rev. Control 34 (1) (2010) 1–12.
[31] L. Zadeh, On the identification problem, IRE Trans. Circuit Theory 3 (4) (1956) 277–281.
[32] L. Ljung, System Identification: Theory for the User, Pearson Education, 1998.
[33] T. Katayama, Subspace Methods for System Identification, Springer Science & Business Media, 2006.
[34] O. Nelles, Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models, Springer Science & Business Media, 2013.
[35] F. Lauer, G. Bloch, Hybrid System Identification: Theory and Algorithms for Learning Switching Models, vol. 478, Springer, 2018.
[36] S. Chen, S. Billings, P. Grant, Non-linear system identification using neural networks, Int. J. Control 51 (6) (1990) 1191–1214.
[37] P.M.L. Drezet, R.F. Harrison, Support vector machines for system identification, in: UKACC International Conference on Control '98 (Conf. Publ. No. 455), vol. 1, 1998, pp. 688–692.
[38] M. Martínez-Ramón, J.L. Rojo-Alvarez, G. Camps-Valls, J. Muñoz-Marí, E. Soria-Olivas, A.R. Figueiras-Vidal, et al., Support vector machines for nonlinear kernel ARMA system identification, IEEE Trans. Neural Netw. 17 (6) (2006) 1617–1622.
[39] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, L. Ljung, Kernel methods in system identification, machine learning and function estimation: A survey, Automatica 50 (3) (2014) 657–682.
[40] F. Lauer, On the complexity of piecewise affine system identification, Automatica 62 (2015) 148–153.
[41] R. Vidal, S. Soatto, Y. Ma, S. Sastry, An algebraic geometric approach to the identification of a class of linear hybrid systems, in: 42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475), vol. 1, IEEE, 2003, pp. 167–172.
[42] F. Lauer, G. Bloch, R. Vidal, A continuous optimization framework for hybrid system identification, Automatica 47 (3) (2011) 608–613.

[43] A. Bemporad, A. Garulli, S. Paoletti, A. Vicino, A bounded-error approach to piecewise affine system identification, IEEE Trans. Automat. Control 50 (10) (2005) 1567–1580.
[44] A. Garulli, S. Paoletti, A. Vicino, A survey on switched and piecewise affine system identification, IFAC Proc. Vol. 45 (16) (2012) 344–355.
[45] K.M. Pekpe, G. Mourot, K. Gasso, J. Ragot, Identification of switching systems using change detection technique in the subspace framework, in: 2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No. 04CH37601) vol. 4, IEEE, 2004, pp. 3720–3725.
[46] V. Verdult, M. Verhaegen, Subspace identification of piecewise linear systems, in: 2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No. 04CH37601), vol. 4, IEEE, 2004, pp. 3838–3843.
[47] J. Borges, V. Verdult, M. Verhaegen, M.A. Botto, A switching detection method based on projected subspace classification, in: Proceedings of the 44th IEEE Conference on Decision and Control, IEEE, 2005, pp. 344–349.
[48] R.V. Lopes, G.A. Borges, J.Y. Ishihara, New algorithm for identification of discrete-time switched linear systems, in: 2013 American Control Conference, IEEE, 2013, pp. 6219–6224.
[49] F. Lauer, G. Bloch, Piecewise smooth system identification in reproducing kernel Hilbert space, in: 53rd IEEE Conference on Decision and Control, IEEE, 2014, pp. 6498–6503.
[50] J. Tugnait, Adaptive estimation and identification for discrete systems with Markov jump parameters, IEEE Trans. Automat. Control 27 (5) (1982) 1054–1065.
[51] X. Jin, B. Huang, Identification of switched Markov autoregressive exogenous systems with hidden switching state, Automatica 48 (2) (2012) 436–441.
[52] V. Krishnamurthy, J.B. Moore, On-line estimation of hidden Markov model parameters based on the Kullback-Leibler information measure, IEEE Trans. Signal Process. 41 (8) (1993) 2557–2573.
[53] U. Orguner, M. Demirekler, An online sequential algorithm for the estimation of transition probabilities for jump Markov linear systems, Automatica 42 (10) (2006) 1735–1744.
[54] A. Bemporad, V. Breschi, D. Piga, S.P. Boyd, Fitting jump models, Automatica 96 (2018) 11–21.
[55] D. Ernst, M. Glavic, F. Capitanescu, L. Wehenkel, Reinforcement learning versus model predictive control: A comparison on a power system problem, IEEE Trans. Syst. Man Cybern. B 39 (2) (2009) 517–529.
[56] F.L. Lewis, D. Vrabie, Reinforcement learning and adaptive dynamic programming for feedback control, IEEE Circuits Syst. Mag. 9 (3) (2009) 32–50.
[57] B. Recht, A tour of reinforcement learning: The view from continuous control, Ann. Rev. Control Robot. Auton. Syst. 2 (1) (2019) 253–279.
[58] D. Gorges, Relations between model predictive control and reinforcement learning, IFAC-PapersOnLine 50 (1) (2017) 4920–4928, 20th IFAC World Congress.
[59] R.S. Sutton, A.G. Barto, Introduction to Reinforcement Learning, first ed., MIT Press, Cambridge, MA, USA, 1998.
[60] M. Gheshlaghi Azar, R. Munos, B. Kappen, On the sample complexity of reinforcement learning with a generative model, 2012, arXiv e-prints arXiv:1206.6461.
[61] R.S. Sutton, Dyna, an integrated architecture for learning, planning, and reacting, SIGART Bull. 2 (4) (1991) 160–163.
[62] B. Peng, X. Li, J. Gao, J. Liu, K.-F. Wong, S.-Y. Su, Deep dyna-Q: Integrating planning for task-completion dialogue policy learning, 2018, arXiv e-prints arXiv:1801.06176.
[63] T. Weber, S. Racanière, D.P. Reichert, L. Buesing, A. Guez, D. Jimenez Rezende, A. Puigdomènech Badia, O. Vinyals, N. Heess, Y. Li, R. Pascanu, P. Battaglia, D. Hassabis, D. Silver, D. Wierstra, Imagination-augmented agents for deep reinforcement learning, 2017, arXiv e-prints arXiv:1707.06203.
[64] A. Nagabandi, G. Kahn, R.S. Fearing, S. Levine, Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning, 2017, arXiv e-prints arXiv:1708.02596.
[65] L.D. Pyeatt, Reinforcement learning with decision trees, in: Applied Informatics, 2003.
[66] D. Ernst, P. Geurts, L. Wehenkel, Tree-based batch mode reinforcement learning, J. Mach. Learn. Res. (2005).
[67] M. Kuss, C.E. Rasmussen, GaussIan processes in reinforcement learning, in: Advances in Neural Information Processing Systems, 2004, pp. 751–758.
[68] T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, J. Ba, Benchmarking model-based reinforcement learning, 2019, arXiv e-prints arXiv:1907.02057.
[69] G. Costanzo, S. Iacovella, F. Ruelens, T. Leurs, B. Claessens, Experimental analysis of data-driven control for a building heating system, Sustain. Energy Grids Netw. 6 (2016) 81–90.
[70] L. Eller, L.C. Siafara, T. Sauter, Adaptive control for building energy management using reinforcement learning, in: 2018 IEEE International Conference on Industrial Technology, ICIT, 2018, pp. 1562–1567.
[71] P. Ferreira, A. Ruano, S. Silva, E. Conceicao, Neural networks based predictive control for thermal comfort and energy savings in public buildings, Energy Build. 55 (2012) 238–251.
[72] A. Afram, F. Janabi-Sharifi, A.S. Fung, K. Raahemifar, Artificial neural network (ANN) based model predictive control (MPC) and optimization of HVAC systems: A state of the art review and case study of a residential HVAC system, Energy Build. 141 (2017) 96–113.
[73] L. Breiman, Classification and Regression Trees, Routledge, 2017.
[74] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[75] A. Jain, F. Smarra, M. Behl, R. Mangharam, Data-driven model predictive control with regression trees-An application to building energy management, ACM Trans. Cyber-Phys. Syst. 2 (1) (2018) 4.
[76] D. Gyalistras, M. Gwerder, Use of weather and occupancy forecasts for optimal building climate control (opticontrol): Two years progress report main report, in: Terrestrial Systems Ecology, ETH Zurich R&D HVAC Products, Building Technologies Division, Siemens Switzerland Ltd, Zug, Switzerland, 2010, p. 83.
[77] D.B. Crawley, L.K. Lawrie, F.C. Winkelmann, W.F. Buhl, Y.J. Huang, C.O. Pedersen, R.K. Strand, R.J. Liesen, D.E. Fisher, M.J. Witte, et al., Energyplus: creating a new-generation building energy simulation program, Energy Build. 33 (4) (2001) 319–331.
[78] S. Merkblatt, 2024: Standard-Nutzungsbedingungen für die Energie-und Gebäudetechnik, Swiss Society of Engineers and Architects, Zürich, 2006.
[79] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, R. Tibshirani, The Elements of Statistical Learning, vol. 2, no. 1, Springer, 2009.
[80] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, Cambridge, 1996.