Automatica 96 (2018) 11–21

Contents lists available at ScienceDirect

Automatica

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

Fitting jump models[☆]

Alberto Bemporad[a,*], Valentina Breschi[b], Dario Piga[c], Stephen P. Boyd[d]

[a] IMT School for Advanced Studies Lucca, Piazza San Francesco 19, 55100 Lucca, Italy
[b] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza L.Da Vinci, 32, 20133 Milano, Italy
[c] Dalle Molle Institute for Artificial Intelligence Research - USI/SUPSI, Galleria 2, Via Cantonale 2c, CH-6928 Manno, Switzerland
[d] Department of Electrical Engineering, Stanford University, Stanford, CA 94305, USA

ARTICLE INFO

Article history:
Received 25 November 2017
Received in revised form 23 February 2018
Accepted 21 May 2018

Keywords:
Model regression
Mode estimation
Jump models
Hidden Markov models
Piecewise affine models

ABSTRACT

We describe a new framework for fitting jump models to a sequence of data. The key idea is to alternate between minimizing a loss function to fit multiple model parameters, and minimizing a discrete loss function to determine which set of model parameters is active at each data point. The framework is quite general and encompasses popular classes of models, such as hidden Markov models and piecewise affine models. The shape of the chosen loss functions to minimize determines the shape of the resulting jump model.

1. Introduction

In many regression and classification problems the training dataset is formed by input and output observations with time stamps. However, when fitting the function that maps input data to output data, most algorithms used in supervised learning do not take the temporal order of the data into account. For example, in linear regression problems solved by least squares $\min_\theta \|A\theta - b\|_2^2$ each row of A and b is associated with a data-point, but clearly the solution $\theta^\star$ is the same no matter how the rows of A and b are ordered. In system identification temporal information is often used only to construct the input samples (or regressors) and outputs, but then it is neglected. For example, in estimating autoregressive models with exogenous inputs (ARX), the regressor is a finite collection of current and past signal observations, but the order of the regressor/output pairs is irrelevant when least squares are used. Similarly, in logistic regression and support vector machines the order of the data points does not affect the result. In training forward neural networks using stochastic gradient descent, the samples may be picked up randomly (and more than once) by the solution algorithm, and again their original temporal ordering is neglected.

On the other hand, there are many applications in which relevant information is contained not only in data values but also in their temporal order. In particular, if the time each data-point was collected is taken into account, one can detect changes in the type of regime the data were produced. Examples range from video segmentation (Chan & Vasconcelos, 2008; Oh, Rehg, Balch, & Dellaert, 2008) to speech recognition (Rabiner, 1989; Schuller, Wöllmer, Moosmayr, Ruske, & Rigoll, 2008), asset-price models in finance (Guidolin, 2011; Timmermann, 2015), human action classification (Ozay, Sznaier, & Lagoa, 2010; Pavlovic, Rehg, & MacCormick, 2001), and many others. All these examples are characterized by the need of fitting multiple models and understanding when switches from one model to another occur.

Piecewise affine (PWA) models attempt at fitting multiple affine models to a dataset, where each model is active based on the location of the input sample in a polyhedral partition of the input space (Breschi, Piga, & Bemporad, 2016b; Ferrari-Trecate, Muselli, Liberati, & Morari, 2003). However, as for ARX models, the order of the data is not relevant in computing the model parameters and the polyhedral partition. In some cases, mode transitions are captured by finite state machines, for example in hybrid dynamical models with logical states, where the current mode and the next logical state are generated deterministically by Boolean functions (Bemporad & Giorgetti, 2006; Breschi, Bemporad, & Piga, 2016a). In spite of the difficulty of assessing whether a switched linear dynamical system is identifiable from input/output data (Vidal, Chiuso, & Soatto, 2002), a rich variety of identification methods have been proposed in the literature (Bemporad, Garulli, Paoletti, & Vicino, 2005; Bemporad, Roll, & Ljung, 2001; Breschi et al., 2016b;

[☆] The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Alessandro Chiuso under the direction of Editor Torsten Söderström.
[*] Corresponding author.
E-mail addresses: alberto.bemporad@imtlucca.it (A. Bemporad), valentina.breschi@polimi.it (V. Breschi), dario.piga@supsi.ch (D. Piga), boyd@stanford.edu (S.P. Boyd).

https://doi.org/10.1016/j.automatica.2018.06.022
0005-1098/© 2018 Elsevier Ltd. All rights reserved.

Ferrari-Trecate et al., 2003; Juloski, Heemels, Ferrari-Trecate, Vidal, Paoletti, & Niessen, 2005; Juloski, Weiland, & Heemels, 2004; Pillonetto, 2016).

Hidden Markov models (HMMs) treat instead the mode as a stochastic discrete variable, whose temporal dynamics is described by a Markov chain (Rabiner, 1989). Natural extensions of hidden Markov models consider the cases in which each mode is associated with a linear function of the input (Costa, Fragoso, & Marques, 2006; Fridman, 1994; Ohlsson & Ljung, 2013). Hidden Markov models are usually trained using the Baum–Welch algorithm (Baum, Petrie, Soules, & Weiss, 1970), a forward–backward version of the more general Expectation–Maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977).

In this paper we consider rather general *jump models* to fit a temporal sequence of data that takes the ordering of the data into account. The proposed fitting algorithm alternates two steps: estimate the parameters of multiple models and estimate the temporal sequence of model activation, until convergence. The model fitting step can be carried out exactly when it reduces to a convex optimization problem, which is often the case. The mode-sequence step is always carried out optimally using dynamic programming.

Our jump modeling framework is quite general. The structure of the model depends on the shape of the function that is minimized to obtain the model parameters, the way the model jumps depends on the function that is minimized to get the sequence of model activation. When we impose no constraints or penalty on the model sequence, our method reduces to automatically splitting the dataset in K clusters and fitting one model per cluster, which is a generalization of K-means (Hastie, Tibshirani, & Friedman, 2009 Algorithm 14.1). Hidden Markov models (HMMs) are a special case of jump models, as we will show in the paper. Indeed, jump models have broader descriptive capabilities than HMMs, for example the sequence of discrete states may not be necessarily generated by a Markov chain and could be a deterministic function. Moreover, as stated above, jump models can have rather arbitrary model shapes.

After introducing jump models in Section 2 and giving a statistical interpretation of the loss function in Section 3, we provide algorithms for fitting jump models to data and to estimate output values and hidden modes from available input samples in Section 4, emphasizing differences and analogies with HMMs. Finally, in Section 5 we show four examples of application of our approach for regression and classification, using both synthetic and experimental datasets.

The code implementing the algorithms described in the paper is available at http://cse.lab.imtlucca.it/~bemporad/jump_models/.

1.1. Setting and goal

We are given a training sequence of data pairs (x_t, y_t), $t = 1, \ldots, T$, with $x_t \in \mathcal{X}, y_t \in \mathcal{Y}$. We refer to t as the *time* or *period*, x_t as the *regressor* or *input*, and y_t as the *outcome* or *output* at time t. The training sequence is used to build a regression model that provides a *prediction* $\hat{y}_t$ of y_t given the available inputs $x_1, \ldots, x_t$, and possibly past outputs $y_1, \ldots, y_{t-1}$. We are specifically interested in models where $\hat{y}_t$ is not simply a static function of x_t, but rather we want to exploit the additional information embedded in the temporal ordering of the data. As we will detail later, our regression model is implicitly defined by the minimization of a *fitting loss* J that depends on $x_1, \ldots, x_t, y_1, \ldots, y_{t-1}, y_t$ and other variables and parameters. The chosen shape for J determines the structure of the corresponding regression model.

Given a production data sequence $(\tilde{x}_1, \tilde{y}_1), \ldots,$ thought to be generated by a similar process that produced the training data, the quality of the regression model over a time period $t = 1, \ldots, \tilde{T}$ will be judged by the average *true loss*

$$L^{\text{true}} = \frac{1}{\tilde{T}} \sum_{t=1}^{\tilde{T}} \ell^{\text{true}}(\hat{y}_t, \tilde{y}_t) \tag{1}$$

where $\ell^{\text{true}} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ penalizes the mismatch between $\hat{y}_t$ and $\tilde{y}_t$, with $\ell(y, y) = 0$ for all $y \in \mathcal{Y}$.

2. Regression models

2.1. Single model

A simple form of deriving a regression model is to introduce a *model parameter* $\theta \in \mathbb{R}^d$, a *loss function* $\ell : \mathcal{X} \times \mathcal{Y} \times \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$, and a *regularizer* $r : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ defining the *fitting objective*

$$J(X, Y, \theta) = \sum_{t=1}^{T} \ell(x_t, y_t, \theta) + r(\theta) \tag{2a}$$

where $X = (x_1, \ldots, x_T)$, $Y = (y_1, \ldots, y_T)$. For a given training dataset (X, Y), let

$$\theta^\star = \arg\min_{\theta} J(X, Y, \theta) \tag{2b}$$

be the optimal model parameter. By fixing $\theta = \theta^\star$ and exploiting the separability of the loss J in (2a) we get the following regression model

$$\begin{aligned} \hat{y}_t &= \arg\min_{y} J(X, Y, \theta^\star) = \arg\min_{y} \ell(x_t, y, \theta^\star) \\ &=: \varphi(x_t) \end{aligned} \tag{2c}$$

where $\varphi : \mathcal{X} \to \mathcal{Y}$ as the regression model, with ties in the arg min broken arbitrarily. For example, when $\ell(x_t, y, \theta) = \left\|y - \theta' x_t\right\|_2^2$ we get the standard linear regression model $\hat{y}_t = \theta' x_t$.

Model (2c) can be enriched by adding *output information sets* $\mathcal{Y}_t \subseteq \mathcal{Y}$ that augment the information that is available about y_t,

$$\hat{y}_t = \arg\min_{y \in \mathcal{Y}_t} \ell(x, y, \theta^\star) \tag{3}$$

where $\mathcal{Y}_t = \mathcal{Y}$ if no extra information on y_t is given. For example, if we know a priori that $y_t \geq 0$ we can set $\mathcal{Y}_t$ equal to the nonnegative orthant.

2.2. K-models

Let us add more flexibility and introduce multiple model parameters $\theta_s \in \mathbb{R}^d$, $s = 1, \ldots, K$, and a latent *mode* variable s_t that determines the model parameter θ_{s_t} that is active at step t. Fitting a K-*model* on the training dataset (X, Y), entails choosing the K models by minimizing

$$J(X, Y, \Theta, S) = \sum_{t=1}^{T} \ell(x_t, y_t, \theta_{s_t}) + \sum_{i=1}^{K} r(\theta_i) \tag{4}$$

with respect to $\Theta = (\theta_1, \ldots, \theta_K)$ and $S = (s_1, \ldots, s_T)$. The optimal parameters $\theta_1^\star, \ldots, \theta_K^\star$ define the K-model

$$(\hat{y}_t, \hat{s}_t) = \arg\min_{y,s} \ell(x_t, y, \theta_s^\star). \tag{5}$$

Note that the objective function in (4) is used to estimate the model parameters $\theta_1^\star, \ldots, \theta_K^\star$ based on the entire training dataset, while (5) defines the model used to infer the output $\hat{y}_t$ and discrete state $\hat{s}_t$ given the input x_t, as exemplified in the next section.

2.2.1. K-means and piecewise affine models

The standard K-means model (Hastie et al., 2009) is obtained by setting $y_t = x_t$, $r(\theta) = 0$, and

$$\ell(x_t, y_t, \theta_{s_t}) = \frac{1}{2}\|y_t - \theta_{s_t}\|_2^2 + \frac{1}{2}\|x_t - \theta_{s_t}\|_2^2 = \|x_t - \theta_{s_t}\|_2^2 \tag{6}$$

In this case, minimizing (4) assigns each datapoint x_t to the cluster indexed by $s_t^\star$, and defines $\theta_1^\star, \ \ldots, \theta_K^\star$ as the centroids of the resulting K clusters. Moreover, the regression model defined by (6) returns

$$\hat{s}_t = \arg\min_s \|\tilde{x}_t - \theta_s^\star\|_2^2, \quad \hat{y}_t = \theta_{\hat{s}_t} \tag{7}$$

that is the index $\hat{s}_t$ of the centroid $\theta_{\hat{s}_t}^\star$ which is closest to the given input x_t, and sets $\hat{y}_t = \theta_{\hat{s}_t}^\star$ as the best estimate of x_t.

More generally, by setting

$$\ell(x_t, y_t, \theta_{s_t}) = \left\| y_t - \theta_{y,s_t}' \begin{bmatrix} x_t \\ 1 \end{bmatrix} \right\|_2^2 + \rho \|x_t - \theta_{x,s_t}\|_2^2 \tag{8}$$

with $\theta_{s_t} = (\theta_{y,s_t}, \theta_{x,s_t})$ and $\rho > 0$, we obtain a piecewise affine (PWA) model over the piecewise linear partition generated by the Voronoi diagram of $(\theta_{x,1}^\star, \ldots, \theta_{x,K}^\star)$, i.e., the regression model (5) becomes

$$\hat{s}_t = \arg\min_s \|x_t - \theta_{x,s}^\star\|_2^2, \quad \hat{y}_t = (\theta_{y,\hat{s}_t}^\star)' \begin{bmatrix} x_t \\ 1 \end{bmatrix} \tag{9}$$

The hyper-parameter ρ in (8) trades off between fitting the output y_t and clustering the inputs $(x_1, \ldots, x_t)$ based on their mutual Euclidean distance.

A more general PWA model can be defined by setting

$$\ell(x_t, y_t, \theta_{s_t}) = \left\| y_t - \theta_{y,s_t}' \begin{bmatrix} x_t \\ 1 \end{bmatrix} \right\|_2^2 + \rho \sum_{\substack{j=1 \\ j \neq s_t}}^{K} \max\left\{0, (\theta_{x,j} - \theta_{x,s_t})' \begin{bmatrix} x_t \\ 1 \end{bmatrix} + 1\right\}^2 \tag{10}$$

where $\max_s \left\{\theta_{x,s}' \left[\begin{smallmatrix} x \\ 1 \end{smallmatrix}\right]\right\}$ defines a piecewise linear separation function that induces a polyhedral partition of the input space (Bennett & Mangasarian, 1994; Breschi et al., 2016b). In this case it is immediate to verify that the regression model induced by (5) is

$$\hat{s}_t = \arg\max_s \left\{(\theta_{x,s}^\star)' \begin{bmatrix} x_t \\ 1 \end{bmatrix}\right\}, \quad \hat{y}_t = (\theta_{y,\hat{s}_t}^\star)' \begin{bmatrix} x_t \\ 1 \end{bmatrix}. \tag{11}$$

2.3. Jump model

The models introduced above do not take into account the temporal order in which the samples (x_t, y_t) are generated. To this end, we add a *mode sequence loss* $\mathcal{L}$ in the fitting objective (4)

$$J(X, Y, \Theta, S) = \sum_{t=1}^{T} \ell(x_t, y_t, \theta_{s_t}) + \sum_{k=1}^{K} r(\theta_k) + \mathcal{L}(S), \tag{12}$$

where $S = (s_0, s_1, \ldots, s_T)$ is the mode sequence. We define $\mathcal{L} : \mathcal{K}^{T+1} \to \mathbb{R} \cup \{+\infty\}$ in (12) as

$$\mathcal{L}(S) = \mathcal{L}^{\text{init}}(s_0) + \sum_{t=1}^{T} \mathcal{L}^{\text{mode}}(s_t) + \sum_{t=1}^{T} \mathcal{L}^{\text{trans}}(s_t, s_{t-1}) \tag{13a}$$

where $\mathcal{K} = \{1, \ldots, K\}$, $\mathcal{L}^{\text{init}} : \mathcal{K} \to \mathbb{R} \cup \{+\infty\}$ is the *initial mode cost*, $\mathcal{L}^{\text{mode}} : \mathcal{K} \to \mathbb{R} \cup \{+\infty\}$ is the *mode cost*, and $\mathcal{L}^{\text{trans}} : \mathcal{K}^2 \to \mathbb{R} \cup \{+\infty\}$ is the *mode transition cost*. We discuss possible choices for $\mathcal{L}$ in Sections 2.3.1 and 3.

With a little abuse of notation, we write

$$J(X, Y, \Theta, S) = \ell(X, Y, \Theta, S) + r(\Theta) + \mathcal{L}(S) \tag{13b}$$

where

$$\ell(X, Y, \Theta, S) = \sum_{t=1}^{T} \ell(x_t, y_t, \theta_{s_t}), \quad r(\Theta) = \sum_{k=1}^{K} r(\theta_k). \tag{13c}$$

As with any model, the choice of the fitting objective (13) should trade off between fitting the given data and prior assumptions we have about the models and the mode sequence. In particular, the mode sequence loss $\mathcal{L}$ in (13a) takes into account the temporal structure of the mode sequence, for example that the mode might change (i.e., $s_t \neq s_{t-1}$) rarely.

A jump model can be used for several tasks beyond inferring the values $\hat{y}_t$. In *anomaly identification*, we are interested in determining times t for which the jump model does not fit the data point y_t well. In *model change detection* we are interested in identifying times t for which $\hat{s}_t \neq \hat{s}_{t-1}$. In *control systems* jump models can be used to approximate nonlinear/discontinuous dynamics and design model-based control policies, state estimators, and fault-detection algorithms.

2.3.1. Mode loss functions

We discuss a few options for choosing the mode loss functions $\mathcal{L}^{\text{init}}$, $\mathcal{L}^{\text{mode}}$, $\mathcal{L}^{\text{trans}}$ defining the mode sequence loss $\mathcal{L}$ in (13a). As we assume that the number K of possible modes must be fixed, K must be chosen as a trade off between fitting the model to data (K large) and limit the complexity of the model and avoid overfitting (K small). The best value is usually determined after performing cross-validation.

As mentioned above, the case $\mathcal{L}(S) = 0$ leads to a K-model. By choosing $\mathcal{L}^{\text{trans}}(i,j) = \lambda$ for all $i \neq j$, $\mathcal{L}^{\text{mode}}(i) = \mathcal{L}^{\text{trans}}(i,i) = 0$, one penalizes mode transitions equally by $\lambda \geq 0$, where $\lambda \to \infty$ leads to regression of a single model on the data (that is, $s_t \equiv s_0$), while $\lambda \to 0$ leads again to a K-model. Note that choosing the same constant λ for all transitions makes the fitting problem exhibit multiple solutions, as indexes i, j can be arbitrarily permuted. The mode loss $\mathcal{L}^{\text{mode}}$ can be used to break such symmetries. For example, smaller values for s_t will be preferred by making $\mathcal{L}^{\text{mode}}(i) < \mathcal{L}^{\text{mode}}(j)$ for $i < j$. The shape of the increasing finite sequence $\{\mathcal{L}^{\text{mode}}(i)\}_{i=1}^{K}$ can be used to reduce the number of possible modes: larger increasing values of $\mathcal{L}^{\text{mode}}(i)$ will discourage the use of an increasing number of modes.

The initial mode cost $\mathcal{L}^{\text{init}}$ summarizes prior knowledge about the initial mode s_0. For example, $\mathcal{L}^{\text{init}}(s_0) \equiv 0$ if no prior information on s_0 is available. On the contrary, if the initial mode s_0 is known and say equal to j, then $\mathcal{L}^{\text{init}}(s_0) = 0$ for $s_0 = j$ and $+\infty$ otherwise.

Next Section 3 suggests criteria for choosing $\mathcal{L}$ in case statistical assumptions about the underlying process that generates s_t are available. Alternative criteria are discussed in Section 4.4 for choosing $\mathcal{L}$ directly from the training data.

3. Statistical interpretations

Let $Y = (y_1, \ldots, y_T)$, $X = (x_1, \ldots, x_T)$, $S = (s_0, \ldots, s_T)$, $\Theta = (\theta_1, \ldots, \theta_K)$. We provide a statistical interpretation of the loss functions for the special case in which the following modeling assumptions are satisfied:

A1. The mode sequence S, the model parameters Θ and the input data X are statistically independent, i.e.,

$$p(S|X, \Theta) = p(S), \quad p(\Theta|S, X) = p(\Theta)$$

A2. The conditional likelihood of Y is given by

$$p(Y|X, S, \Theta) = \prod_{t=1}^{T} p(y_t|X, S, \Theta) = \prod_{t=1}^{T} p(y_t|x_t, \theta_{s_t})$$

where $p(y_t|x_t, \theta_{s_t})$ is the likelihood of the outcome y_t given x_t and θ_{s_t};

A3. The priors on the model parameters $\theta_1, \ldots, \theta_K$ are all equal to $p(\theta)$, i.e.,

$$p(\theta_1) = \cdots = p(\theta_K) = p(\theta)$$

and the model parameters are statistically independent, i.e.,

$$p(\Theta)=\prod_{k=1}^{K}p(\theta_k)$$

A4. The probability of being in mode s_t given $s_0,\ldots,s_{t-1}$ is $p(s_t|s_t-1)=\pi_{s_t,s_{t-1}}$ (Markov property);
A5. The initial mode s_0 has probability $p(s_0)=\pi_{s_0}$.

Proposition 1. *Let Assumptions* A1–A5 *be satisfied and define*

$$\ell(x_t,y_t,\theta_{s_t}) = -\log p(y_t|x_t,\theta_{s_t}) \tag{14a}$$
$$r(\theta_k) = -\log p(\theta_k) \tag{14b}$$
$$\mathcal{L}^{\text{trans}}(s_t,s_{t-1}) = -\log \pi_{s_t,s_{t-1}} \tag{14c}$$
$$\mathcal{L}^{\text{init}}(s_0) = -\log \pi_{s_0} \tag{14d}$$
$$\mathcal{L}^{\text{mode}}(s_t) = 0. \tag{14e}$$

Then minimizing $J(X,Y,\Theta,S)$ as defined in (12)–(14) with respect to Θ and S is equivalent to maximizing the joint likelihood $p(Y,S,\Theta|X)$.

Proof. Because of the Markov property (Assumption A4), the likelihood of the mode sequence S is

$$p(S)=p(s_0)\prod_{t=1}^{T}p(s_t|s_{t-1}). \tag{15}$$

From (15) and Assumptions A1–A3, we have:

$$\begin{aligned}p(Y,S,\Theta|X) &= p(\Theta|X)p(Y,S|X,\Theta)\\ &= p(\Theta|X)p(S|X,\Theta)p(Y|S,X,\Theta)\\ &= p(\Theta)p(S)p(Y|S,X,\Theta)\\ &= \prod_{k=1}^{K}p(\theta_k)p(s_0)\prod_{t=1}^{T}p(s_t|s_{t-1})p(y_t|x_t,\theta_{s_t})\end{aligned}$$

whose logarithm is

$$\begin{aligned}\log p(Y,S,\Theta|X) = &\sum_{k=1}^{K}\log p(\theta_k)+\log p(s_0)\\ &+\sum_{t=1}^{T}\log p(s_t|s_{t-1})+\sum_{t=1}^{T}\log p(y_t|x_t,\theta_{s_t}).\end{aligned} \tag{16}$$

By defining the loss functions ℓ, r, $\mathcal{L}^{\text{trans}}$, $\mathcal{L}^{\text{init}}$, and $\mathcal{L}^{\text{mode}}$ as in (14), the minimization of the fitting objective $J(X,Y,\Theta,S)$ as in (12)–(13) with respect to Θ and S is equivalent to maximizing the logarithm of the joint likelihood $p(Y,S,\Theta|X)$, and therefore $p(Y,S,\Theta|X)$. ■

The following proposition provides an inverse result, namely a statistical interpretation of minimizing a given generic $J(X,Y,\Theta,S)$ defined as in (13).

Proposition 2. *Define the probability density functions*

$$p(y_t|x_t,\theta_{s_t}) = \frac{e^{-\ell(x_t,y_t,\theta_{s_t})}}{\nu(\theta_{s_t},x_t)} \tag{17a}$$
$$p(S,\Theta|X) = \frac{\nu(S,\Theta,X)e^{-\mathcal{L}(S)-r(\Theta)}}{\sum_{\bar S\in K^{T+1}}\int_{\mathbb{R}^{d\times K}}\nu(\bar S,\Theta,X)e^{-\mathcal{L}(\bar S)-r(\Theta)}d\Theta} \tag{17b}$$

where

$$\nu(\theta_{s_t},x_t) = \int_{\mathcal{Y}} e^{-\ell(x_t,y,\theta_{s_t})}dy \tag{18a}$$
$$\nu(S,\Theta,X) = \prod_{t=1}^{T}\nu(\theta_{s_t},x_t) \tag{18b}$$

and assume that the outputs Y are conditionally independent given (S,X,Θ), i.e., $p(Y|S,X,\Theta)=\prod_{t=1}^{T}p(y_t|x_t,\theta_{s_t})$. Then the following identity holds

$$\arg\min_{S,\Theta} J(X,Y,\Theta,S) = \arg\max_{S,\Theta}\log p(Y,S,\Theta|X) \tag{19}$$

Proof. Since

$$p(Y,S,\Theta|X)=p(Y|S,X,\Theta)p(S,\Theta|X) \tag{20}$$

by substituting (18) in (20) we get

$$\begin{aligned}&p(Y,S,\Theta|X)=\\ &\frac{\prod_{t=1}^{T}e^{-\ell(x_t,y_t,\theta_{s_t})}}{\prod_{t=1}^{T}\nu(\theta_{s_t},x_t)}\frac{\nu(S,\Theta,X)e^{-\mathcal{L}(S)-r(\Theta)}}{\sum_{\bar S\in K^{T+1}}\int_{\mathbb{R}^{d\times K}}\nu(\bar S,\Theta,X)e^{-\mathcal{L}(\bar S)-r(\Theta)}d\Theta}\\ &=\frac{e^{-\sum_{t=1}^{T}\ell(x_t,y_t,\theta_{s_t})-\mathcal{L}(S)-r(\Theta)}}{\sum_{\bar S\in K^{T+1}}\int_{\mathbb{R}^{d\times K}}\nu(\bar S,\Theta,X)e^{-\mathcal{L}(\bar S)-r(\Theta)}d\Theta}\end{aligned} \tag{21}$$

As the denominator in (21) does not depend on S and Θ, maximize $p(Y,S,\Theta|X)$ is equivalent to maximize

$$e^{-\sum_{t=1}^{T}\ell(x_t,y_t,\theta_{s_t})-\mathcal{L}(S)-r(\Theta)},$$

or, equivalently, to minimize

$$\sum_{t=1}^{T}\ell(x_t,y_t,\theta_{s_t})+\mathcal{L}(S)+r(\Theta)$$

The identity (19) thus follows from the definition of $J(X,Y,\Theta,S)$ in (13). ■

The following corollary provides a set of probabilistic interpretations of the loss function $J(X,Y,\Theta,S)$, some of which are well known in Bayesian estimation.

Corollary 1. *Let $\nu(\theta_{s_t},x_t)$ in (18a) be a constant. Then the following statements hold:*

1. *The quadratic regularization $r(\Theta)=\rho\sum_{k=1}^{K}\|\theta_k\|_2^2$ corresponds to assuming a Gaussian prior on θ_k, namely $p(\theta_k)=ce^{-\frac{\|\theta_k\|_2^2}{2\sigma_\theta^2}}$ with $\sigma_\theta=\sqrt{\frac{1}{2\rho}}$.*
2. *The quadratic penalty on the prediction error*
$$\ell(x_t,y_t,\theta_{s_t})=c\|y_t-\theta_{s_t}'x_t\|_2^2 \tag{22}$$
correspond to assuming the probabilistic model of the output $y_t\sim N(\theta_{s_t}'x_t,\sigma_y^2 I)$, with $\sigma_y=\sqrt{\frac{1}{2c}}$.
3. *Setting $\mathcal{L}^{\text{trans}}=0$ is equivalent to assuming that the modes s_t are i.i.d., with*
$$s_t\sim p(s_t)=\frac{e^{-\mathcal{L}^{\text{mode}}(s_t)}}{\sum_{k=1}^{K}e^{-\mathcal{L}^{\text{mode}}(k)}}$$
Furthermore, setting $\mathcal{L}(S)=0$ corresponds to assuming that $p(s_t)=\frac{1}{K}$ for all $t=0,\ldots,T$, while setting $\mathcal{L}^{\text{init}}(s)=\mathcal{L}^{\text{mode}}(s)=s$, $s=1,\ldots,K$, corresponds to assuming $p(s)=\frac{(e-1)}{1-e^{-K}}e^{-s}$.
4. *Under the assumption $p(S|\Theta,X)=p(S)=p(s_0)\prod_{t=1}^{T}p(s_t|s_{t-1})$, the case $\mathcal{L}^{\text{mode}}=\mathcal{L}^{\text{init}}=0$ and $\mathcal{L}^{\text{trans}}(i,j)=\lambda$ for $i\neq j$ and 0 for $i=j$, corresponds to assume that*
$$p(s_0)=\frac{1}{K},\quad p(s_t|s_{t-1})=\begin{cases}\dfrac{e^{-\lambda}}{1+(K-1)e^{-\lambda}} & \text{if}\quad s_t\neq s_{t-1}\\[2mm] \dfrac{1}{1+(K-1)e^{-\lambda}} & \text{if}\quad s_t=s_{t-1}\end{cases}$$

Proof. As $\nu(\theta_{s_t}, x_t)$ does not depend on θ_{s_t} and X, $p(S, \Theta|X)$ in (17b) can be written as $p(S, \Theta|X) = p(S)p(\Theta)$, where

$$p(S) = \frac{e^{-\mathcal{L}(S)}}{\sum_{\bar{S}\in\mathcal{K}^{T+1}} e^{-\mathcal{L}(\bar{S})}}, \quad p(\Theta) = \frac{e^{-r(\Theta)}d\Theta}{\int_{\mathbb{R}^{d\times K}} e^{-r(\Theta)}d\Theta} \tag{23}$$

The results follow straightforwardly from the above expressions of $p(S)$ and $p(\Theta)$ and the definition of $\mathcal{L}(s)$ in (13a). ■

4. Algorithms

We provide now algorithms for fitting a jump model to a given dataset and to infer predictions $\hat{y}_t, \hat{s}_t$ from it.

4.1. Model fitting

Given a training sequence $X = (x_1, \dots, x_T)$ of inputs and $Y = (y_1, \dots, y_T)$ of outputs, for fitting a jump K-model we need to attempt minimizing the cost $J(X, Y, \Theta, S)$ with respect to Θ and S. A simple algorithm to solve this problem is Algorithm 1, a coordinate descent algorithm that alternates minimization with respect to Θ and S. If ℓ and r are convex functions, Step 1.1 can be solved globally (up to the desired precision) by standard convex programming (Boyd & Vandenberghe, 2004). Step 1.2 can be solved to global optimality by standard discrete *dynamic programming* (DP) (Bellman, 1957) with complexity $O\left(TK^2\right)$. This is achieved by computing the following matrices $M \in \mathbb{R}^{K\times(T+1)}$ of costs and $U \in \mathcal{K} \times \mathbb{R}^T$ of indexes

$$M(s,T) = \mathcal{L}^{\text{mode}}(s) + \ell(x_T, y_T, \theta_s) \tag{24a}$$

$$U_{s,t} = \arg\min_j \{M(j, t+1) + \mathcal{L}^{\text{trans}}(j, s)\}, \quad t = 1, \dots, T-1 \tag{24b}$$

$$M(s,t) = \mathcal{L}^{\text{mode}}(s) + \ell(x_t, y_t, \theta_s) + M(U_{s,t}, t+1) + \mathcal{L}^{\text{trans}}(U_{s,t}, s) \tag{24c}$$

$$M(s,0) = \mathcal{L}^{\text{init}}(s) + \min_j\{M(j,1) + \mathcal{L}^{\text{trans}}(j,s)\} \tag{24d}$$

backwards in time, and then reconstructing the minimum cost sequence S forward in time by setting

$$s_0 = \arg\min_j M(j, 0) \tag{24e}$$

$$s_t = U_{s_{t-1},t}, \ t = 1, \dots, T. \tag{24f}$$

Note that if the time order of operations in (24) is reversed, the DP iterations (24) become Viterbi algorithm (Rabiner, 1989, p. 264):

$$M(s,0) = \mathcal{L}^{\text{init}}(s) \tag{25a}$$

$$U_{s,t} = \arg\min_j \{M(j, t-1) + \mathcal{L}^{\text{trans}}(j, s)\}, \quad t = 1, \dots, T \tag{25b}$$

$$M(s,t) = \mathcal{L}^{\text{mode}}(s) + \ell(x_t, y_t, \theta_s) + M(U_{s,t}, t-1) + \mathcal{L}^{\text{trans}}(U_{s,t}, s) \tag{25c}$$

followed by the backwards iterations

$$s_T = \arg\min_j M(j, T) \tag{25d}$$

$$s_t = U_{s_{t+1},t}, \ t = 0, \dots, T-1. \tag{25e}$$

Since at each iteration the cost $J(X, Y, \Theta, S)$ is non-increasing and the number of sequences S is finite, Algorithm 1 always terminates in a finite number of steps, assuming that in case of multiple optima one selects the optimizers in Steps 1.1 and 1.2 according to some predefined criterion. However, there is no guarantee that the solution found is the global one, as it depends on the initial guess S^0. To improve the quality of the solution, we may run Algorithm 1 N times from different random initial sequences S^0 and select the best result. Our experience is that a small N, say $N = 5$, is usually enough.

Algorithm 1 Jump model fitting

```
Input: Training dataset $X = (x_1, \dots, x_T)$, $Y = (y_1, \dots, y_T)$,
number $K$ of models, initial mode sequence $S^0 = \{s_0^0, \dots, s_T^0\}$.

1. iterate for $k = 1, \dots$
    1.1. $\Theta^k \leftarrow \arg\min_\Theta \ell(X, Y, \Theta, S^{k-1}) + r(\Theta)$;
                                                        (model fitting)
    1.2. $S^k \leftarrow \arg\min_S \ell(X, Y, \Theta^k, S) + \mathcal{L}(S)$;
                                               (mode sequence fitting)
2. until $S^k = S^{k-1}$.

Output: Estimated model parameters $\Theta^\star = \Theta^k$ and mode
sequence $S^\star = S^k$.
```

During the execution of Algorithm 1 it may happen that a mode s does not appear in the sequence S^{k-1}. In this case, the *fitting loss* $\ell(X, Y, \Theta, S^{k-1})$ does not depend on θ_s, and the latter will be determined in Step 1.1 based only on the regularizer $r(\Theta)$.

In case $\mathcal{L}(S) = 0$, the ordering of the training data becomes irrelevant and Algorithm 1 reduces to fitting K models to the dataset. If in addition ℓ and r are specified as in (6) and $Y = X$, Algorithm 1 is the standard K-means algorithm, where the starting sequence S^0 is the initial clustering of the data points $(x_1, \dots, x_T)$, Step 1.1 computes the collection Θ^k of cluster centroids at iteration k, and Step 1.2 reassigns data points to clusters by updating their labels s_t^k.

When again $\mathcal{L}(S) = 0$ and the mode loss in (10) is used for getting a PWA model, the cost function minimized in Step 1.1 of Algorithm 1 is separable with respect to $\theta_{y,s}$, $\theta_{x,s}$. Then the minimization with respect to $\theta_{x,s}$ produces the piecewise linear separation function $\max_s \left\{\theta_{x,s}' \left[\begin{smallmatrix} x \\ 1 \end{smallmatrix}\right]\right\}$ that defines the polyhedral partition of the input space (Breschi et al., 2016b), while Step 1.2 looks for the optimal latent variables s_t that best trade off between assigning the corresponding data point x_t to the polyhedron $\{x \in \mathcal{X} : \theta_{x,s_t}' \left[\begin{smallmatrix} x \\ 1 \end{smallmatrix}\right] \geq \theta_{x,j}' \left[\begin{smallmatrix} x \\ 1 \end{smallmatrix}\right], \ \forall j \neq s_t, j \in \mathcal{K}\}$and matching the predicted output $y_t \approx \theta_{y,s_t}' \left[\begin{smallmatrix} x_t \\ 1 \end{smallmatrix}\right]$.

Finally, we remark that Algorithm 1 is also applicable to the more general case in which the mode loss $\mathcal{L}$ also depends on Θ, by simply replacing Steps 1.1 and 1.2 with

$$\Theta^k \leftarrow \arg\min_\Theta \ell(X, Y, \Theta, S^{k-1}) + r(\Theta) + \mathcal{L}(S^{k-1}, \Theta) \tag{26a}$$

$$S^k \leftarrow \arg\min_S \ell(X, Y, \Theta^k, S) + \mathcal{L}(S, \Theta^k). \tag{26b}$$

This would cover the case in which $\mathcal{L}$ contains parameters to be estimated.

4.2. Inference

4.2.1. One-step ahead prediction

Assume that the model parameters $\Theta^\star$ have been estimated and that new production data $\tilde{X}_t = (\tilde{x}_1, \dots, \tilde{x}_t)$ and outputs $\tilde{Y}_{t-1} = (\tilde{y}_1, \dots, \tilde{y}_{t-1})$ are given. Because of the structure of the mode loss function $\mathcal{L}$ defined in (13a), the estimates $\hat{y}_t$ and $\hat{s}_0, \dots, \hat{s}_t$ do not depend on future inputs $\tilde{x}_j$ and modes $\hat{s}_j$ for $j > t$.

The same fitting objective (12) can be used to estimate $\hat{y}_t$ and $\hat{S}_t = (\hat{s}_0, \dots, \hat{s}_t)$,

$$(\hat{y}_t, \hat{S}_t) = \arg\min_{y,S_t} J_t(\tilde{X}_t, \tilde{Y}_{t-1}, y, \Theta^\star, S_t) \quad \text{s.t. } y \in \mathcal{Y}_t \tag{27}$$

Algorithm 2 Inference

Input: Model set $\Theta^\star$, production dataset $\tilde{X}_t = (\tilde{x}_1, \dots, \tilde{x}_t)$, past outputs $\tilde{Y}_{t-1} = (\tilde{y}_1, \dots, \tilde{y}_{t-1})$.

```
1. $\hat{S}_t \leftarrow \arg\min_{S_t} \left\{ \mathcal{L}(S_t) + \sum_{j=1}^{t-1} \ell(\tilde{x}_j, \tilde{y}_j, \theta^\star_{s_j}) + \min_{y\in\mathcal{Y}_t} \ell(\tilde{x}_t, y, \theta^\star_{s_t}) \right\}$;
2. $\hat{y}_t \leftarrow \arg\min_{y\in\mathcal{Y}_t} \ell(\tilde{x}_t, y, \theta^\star_{\hat{s}_t})$;
```

Output: Estimated output $\hat{y}_t$ and mode sequence $\hat{S}_t$.

where $\mathcal{Y}_t \subseteq \mathcal{Y}$ is a possible additional output information set and

$$J_t(\tilde{X}_t, \tilde{Y}_{t-1}, y, \Theta^\star, S_t) = \ell(\tilde{x}_t, y, \theta^\star_{s_t}) + \sum_{j=1}^{t-1} \ell(\tilde{x}_j, \tilde{y}_j, \theta^\star_{s_j}) + \mathcal{L}^{\text{init}}(s_0) + \sum_{j=1}^{t} \mathcal{L}^{\text{mode}}(s_j) + \sum_{j=1}^{t} \mathcal{L}^{\text{trans}}(s_j, s_{j-1}).$$

Algorithm 2 attempts at solving problem (27) at every t of interest. Step 1 is solved again by the DP iterations (24) over the time span $[0, t]$, with the only difference that in (24a) we set the terminal penalty equal to $M(s,t) = \mathcal{L}^{\text{mode}}(s) + \min_y\{\ell(\tilde{x}_t, y, \theta_s)\}$, since the last output y_t is determined later at Step 2.

Note that *open-loop prediction*, that is the task of predicting $\hat{y}_t$ and $\hat{s}_t$ without acquiring $\tilde{Y}_{t-1}$, can be simply obtained by replacing $\tilde{Y}_{t-1} = (\tilde{y}_1, \dots, \tilde{y}_{t-1})$ with $\hat{Y}_{t-1} = (\hat{y}_1, \dots, \hat{y}_{t-1})$. Arbitrary combinations of one-step ahead and open-loop predictions are possible to handle the more general case of intermittent output data availability.

4.2.2. Recursive inference

When $\mathcal{L}^{\text{trans}} = 0$, problem (27) becomes completely separable and simplifies to

$$(\hat{y}_t, \hat{s}_t) = \arg\min_{y,s} \ell(\tilde{x}_t, y, \theta^\star_s) + \mathcal{L}^{\text{mode}}(s) \quad \text{s.t. } y \in \mathcal{Y}_t. \tag{28}$$

For example, in the case of K-means (6) ($\mathcal{L}(s) = 0$), the estimate obtained by (28) is given by (7).

When the mode transition loss function $\mathcal{L}^{\text{trans}} \neq 0$, the simplification in (28) does not hold anymore. Nonetheless, an incremental version of (27) can be still derived as described in Algorithm 3, where $\mathcal{L}_t : \mathcal{K} \to \mathbb{R}$ is the *arrival cost* recursively computed by the algorithm from the initial condition $\mathcal{L}_0(s_0) = \mathcal{L}^{\text{init}}(s_0)$, for all $s_0 \in \mathcal{K}$. Clearly, while producing exactly the same results, the formulation in Algorithm 3 is much more efficient than Algorithm 2, as the number of computations does not increase with t and thus can be used for online inference.

4.2.3. Smoothing

The same approach described in Section 4.2.1 can be generalized to other inference tasks than one-step ahead or open-loop prediction, such as *smoothing*. Assume $\tilde{y}_k$ is only known at steps $k \in \mathcal{T}_t \subseteq \{1, \dots, t\}$. Steps 1–2 of Algorithm 2 are replaced by

$$\hat{S}_t \leftarrow \arg\min_{S_t} \Big\{ \mathcal{L}(S_t) + \sum_{j\in\mathcal{T}_t} \ell(\tilde{x}_j, \tilde{y}_j, \theta^\star_{s_j}) + \sum_{j\in\bar{\mathcal{T}}_t} \min_{y_j\in\mathcal{Y}_j} \ell(\tilde{x}_j, y_j, \theta^\star_{s_j}) \Big\} \tag{30a}$$

Algorithm 3 Recursive inference

Input: Model $\Theta^\star$, current input $\tilde{x}_t$, past input/output pair $(\tilde{x}_{t-1}, \tilde{y}_{t-1})$, arrival cost $\mathcal{L}_{t-1}$.

```
1. Update
   $\mathcal{L}_t(s_t) \leftarrow \mathcal{L}^{\text{mode}}(s_t) + \min_{s_{t-1}} \{ \ell(\tilde{x}_{t-1}, \tilde{y}_{t-1}, \theta_{s_{t-1}}) + \mathcal{L}_{t-1}(s_{t-1}) + \mathcal{L}^{\text{trans}}(s_t, s_{t-1}) \}$   (29a)
2. Compute
   $(\hat{y}_t, \hat{s}_t) \leftarrow \arg\min_{y,s} \ell(\tilde{x}_t, y, \theta_s) + \mathcal{L}_t(s)$  s.t. $y \in \mathcal{Y}_t$   (29b)
```

Output: Estimated output $\hat{y}_t$ and mode $\hat{s}_t$, updated arrival cost $\mathcal{L}_t$.

$$y_j \leftarrow \arg\min_{y\in\mathcal{Y}_j} \ell(\tilde{x}_j, y, \theta^\star_{\hat{s}_j}), \ \forall j \in \bar{\mathcal{T}}_t \tag{30b}$$

where $\bar{\mathcal{T}}_t = \{1, \dots, t\} \setminus \mathcal{T}_t$. Note that complexity of the inner minimization in (30a) depends on the shape of the loss function ℓ. In the quadratic case, the minimum can be expressed analytically.

4.2.4. Pure mode estimation

In case we are interested in estimating only the latent mode $\hat{s}_t$ given $\tilde{x}_1, \dots, \tilde{x}_t, \tilde{y}_1, \dots, \tilde{y}_{t-1}$ and also $\tilde{y}_t$, we can keep using (29) by simply changing (29b) to

$$\hat{s}_t = \arg\min_s \ell(\tilde{x}_t, \tilde{y}_t, \theta_s) + \mathcal{L}_t(s) \tag{31}$$

This allows reconstructing the mode sequence $\hat{s}_1, \dots, \hat{s}_{\bar{T}}$ recursively from the available dataset, which may be useful for example to detect changes in the relation between the input $\tilde{x}_t$ and the output $\tilde{y}_t$.

4.3. Relation with hidden Markov models

Jump models have several common features with hidden Markov models (HMMs) (Rabiner, 1989). First, both models consider the presence of discrete latent states s_t. While HMMs assume that the sequence S of such states satisfies the Markov property

$$p(s_t|s_{t-1}, \dots, s_0) = p(s_t|s_{t-1})$$

in jump models the particular form chosen in (13a) for the mode sequence loss $\mathcal{L}$ makes estimating $\hat{s}_t$ incrementally as in (29) possible.

Second, in HMMs the observed outputs are such that

$$p(y_t|x_t, \dots, x_1, y_{t-1}, \dots, y_1, s_t, \dots, s_0) = p(y_t|x_t, s_t).$$

Similarly, in jump models $\hat{y}_t$ is a unique function of a given pair (x_t, s_t), as (29b) becomes

$$\hat{y}_t = \arg\min_{y\in\mathcal{Y}_t} \ell(x_t, y, \theta_{s_t}).$$

Indeed, an HMM is a special case of a jump model. Consider the case in which the output observation y_t is discrete, that is $\mathcal{Y} = \{1, \dots, L\}$. An HMM is characterized by the set of discrete probabilities

$$p(s_{t+1} = i|s_t = j) = \pi_{i,j}, \quad i,j \in \mathcal{K} \tag{32a}$$
$$p(s_0) = \pi_{s_0} \tag{32b}$$
$$p(y_t = v|s_t = j) = \beta_{j,v}, \ v \in \mathcal{Y}. \tag{32c}$$

Let us set $x_t = 1$, $\theta_s = s$, and define the loss function ℓ as

$$\ell(x, y, \theta_s) = -\log(\beta_{\theta_s,y}). \tag{33}$$

Similarly to (14), by also setting $r(\theta) = 0$, $\mathcal{L}^{\text{trans}}(s_t, s_{t-1}) = -\log \pi_{s_t,s_{t-1}}$, $\mathcal{L}^{\text{init}}(s_0) = -\log \pi_{s_0}$, and $\mathcal{L}^{\text{mode}}(s) = 0$, the jump model defined by the inference rule (27)–(28) returns the mode sequence $\hat{S}_t$ that best matches the observed sequence $\tilde{Y}_t$ of outputs and that sets the output $\hat{y}_t$ equal to the value $v \in \mathcal{Y}$ that maximizes the probability $\beta_{\hat{s}_t,v}$. An extension to HMMs with continuous observation densities can be obtained by properly redefining the loss function ℓ in (33).

In case the probabilities $\beta_{s,y}$ are not given, but rather must be estimated from a training dataset, we can set instead $\theta_s = [\beta_{s,1} \ \dots \ \beta_{s,L}]'$ along with the loss function ℓ

$$\ell(y, \theta_s) = -\log(e_y' \theta_s) \tag{34}$$

where e_y is the yth column of the identity matrix of size L. If the initial probability distribution π_{s_0} and the state transition probabilities $\pi_{i,j}$ are unknown, they can be estimated by minimizing $J(X, Y, \Theta, S)$ in (12) with $\mathcal{L}^{\text{trans}}(s_t, s_{t-1})$ and $\mathcal{L}^{\text{init}}(s_0)$ as in (14c) and (14d), respectively. This implies that the unknown model parameter Θ should also include π_{s_0} and $\pi_{i,j}$, leading to the general case of having the mode sequence loss $\mathcal{L}$ also dependent on Θ as in (26).

The well-known *Expectation–Maximization* (EM) algorithm (Dempster et al., 1977) determines the parameters of an HMM by maximizing the log-likelihood

$$L_{\text{HMM}}(\Theta|X, Y) = \log p(Y|X, \Theta) = \log \sum_{S \in \mathcal{K}^{T+1}} p(Y, S|X, \Theta)$$

with respect to Θ. Instead, as shown by Proposition 1, our approach maximizes $\log p(Y, S, \Theta|X)$ with respect to Θ and S.

The case of HMMs in which the observations y are a mode-dependent linear function of x rather than discrete has been dealt with for example in Fridman (1994), under the assumption that such a linear relation between input and output samples is perturbed by Gaussian noise. This is a special case of our jump model framework, obtained by setting ℓ as in (22), $\mathcal{L}^{\text{trans}}$ as in (14c), $\mathcal{L}^{\text{init}}$ as in (14d), $\mathcal{L}^{\text{mode}}(s) = 0$, and $r(\theta) = 0$. The training algorithm described in Fridman (1994), however, completely relies on the probabilistic assumptions made about the normal distribution of output noise and the Markovian nature of mode transitions.

In conclusion, jump models are more descriptive than HMMs. The sequence of modes may not be generated by a Markov chain, such as in the case of PWA models (10) and (11), where the mode s_t is a deterministic function of x_t. In addition, the loss and mode loss functions can have rather arbitrary shapes. For example, we may choose $\ell(x, y, \theta_s)$ as the Huber function of $y - \theta_s' x$ for robust regression, which is still a convex loss.

4.4. Selecting the mode sequence loss from data

Selecting the right mode sequence loss $\mathcal{L}$ may not be obvious and require several attempts that involve fitting and cross-validation. A simple approach to choose $\mathcal{L}$ directly from the training data is to update the mode loss function $\mathcal{L}$ after executing Algorithm 1 based on the best sequence $S^\star$ found so far, and run Algorithm 1 again, executing the algorithm N times in total.

Assuming $\mathcal{L}^{\text{mode}} = 0$ and given a set of relative weights $\tau_0, \tau_1, \dots, \tau_K$, we update $\mathcal{L}^{\text{trans}}$, $\mathcal{L}^{\text{init}}$ from one run of Algorithm 1 to another by setting

$$\mu_j \leftarrow \frac{\#\{t \in \{1, \dots, T\} : s_{t-1}^\star = j\}}{T} \tag{35a}$$

$$\mu_{ij} \leftarrow \frac{\#\{t \in \{1, \dots, T\} : s_t^\star = i, \ s_{t-1}^\star = j\}}{T} \tag{35b}$$

$$\mathcal{L}^{\text{trans}}(i, j) \leftarrow -\tau_i \frac{\log\left(\frac{\mu_{ij}}{\mu_j}\right)}{\sum_{j=1}^K \log\left(\frac{\mu_{ij}}{\mu_j}\right)}$$

$$i, j = 1, \dots, K \tag{35c}$$

$$\mu_j^0 \leftarrow \frac{\#\{t \in \{0, \dots, T\} : s_t^\star = j\}}{T} \tag{35d}$$

$$\mathcal{L}^{\text{init}}(j) \leftarrow -\tau_0 \frac{\log\left(\mu_j^0\right)}{\sum_{j=1}^K \log\left(\mu_j^0\right)} \tag{35e}$$

where # denotes the cardinality (number of elements) of a set and $S^\star = (s_0^\star, \dots, s_T^\star)$. The choice in (35) preserves the relative weight between the losses $\mathcal{L}$, ℓ, and r, as

$$\tau_i = \sum_{j=1}^K \mathcal{L}^{\text{trans}}(i, j), \quad i = 1, \dots, K$$

$$\tau_0 = \sum_{j=1}^K \mathcal{L}^{\text{init}}(j)$$

remains the same each time $\mathcal{L}^{\text{init}}(j)$ and $\mathcal{L}^{\text{trans}}(i, j)$ are updated as in (35). Choosing $\mathcal{L}$ as in (35) is motivated by the statistical interpretation (14c)–(14d) and used routinely for estimating state probabilities in HMMs (Rabiner, 1989). Clearly, (35) are well defined only if μ_{ij}, μ_j, $\mu_j^0 > 0$ for all $i, j = 1, \dots, K$. If the latter condition is not satisfied, one may consider adding the following *Laplace smoothing* (Manning, Raghavan, & Schütze, 2008, Ch. 13):

$$\mu_j \leftarrow \frac{1 + \#\{t \in \{1, \dots, T\} : s_{t-1}^\star = j\}}{T + K} \tag{36a}$$

$$\mu_{ij} \leftarrow \frac{1 + \#\{t \in \{1, \dots, T\} : s_t^\star = i, \ s_{t-1}^\star = j\}}{T + K^2} \tag{36b}$$

$$\mu_j^0 \leftarrow \frac{1 + \#\{t \in \{0, \dots, T\} : s_t^\star = j\}}{T + K} \tag{36c}$$

when estimating μ_j, μ_{ij} and μ_j^0.

Computing $\mathcal{L}$ according to (35) after the training step has been found especially useful for improving the quality of inference, both when using (28) or (29b).

5. Examples

We test the algorithms proposed in the previous sections on various problems of regression and classification using jump models. In all the examples, convex optimization methods are used to solve the problem at Step 1.1 of Algorithm 1, while dynamic programming is used to compute the global optimum at Step 1.2. As the DP computation also provides the optimal cost $V^k \triangleq J(X, Y, \Theta^k, S^k)$, when running the tests we replace the termination criterion in Step 2 with

$$V^{k-1} - V^k \leq \epsilon_V \tag{37}$$

where ϵ_V is a small tolerance. In all the examples we set $\epsilon_V = 10^{-8}$.

Furthermore, after the end of the training step, the loss $\mathcal{L}$ is updated as in (35) before making inference.

All tests were run on a MacBook Pro 3 GHz-Intel i7 in MATLAB R2016b. The test code is available for download at http://cse.lab.imtlucca.it/~bemporad/jump_models/.

5.1. Jump linear model regression

We consider a dataset of $T = 10000$ training data and $\tilde{T} = 10000$ production data generated by the following jump linear model with $K = 3$ modes

$$y_t = \theta_{s_t} x_t + \zeta_t$$

with $y_t \in \mathbb{R}$, $x_t \in \mathbb{R}^{20}$, $x_{t,i} \sim \mathcal{N}(0, 1)$ for all $i = 1, \dots, 20$, $\zeta_t \sim \mathcal{N}(0, \sigma_y^2)$. The coefficients of the parameter vectors θ_i are randomly selected from the normal distribution $\mathcal{N}(0, 1)$. The true mode s_t

has probability $\pi = 5\%$ of being different from s_{t-1}, starting from $s_0 = 1$.

We consider the loss functions

$$\begin{aligned}\ell(x_t, y_t, \theta_{s_t}) &= \|y_t - \theta'_{s_t} x_t\|_2^2\\ r(\theta_k) &= 10^{-5}\|\theta_k\|_2^2\\ \mathcal{L}^{\text{trans}}(s_t, s_{t-1}) &= \begin{cases}-\tau\log(1-(K-1)\pi) & \text{if} \quad s_t = s_{t-1}\\ -\tau\log\pi & \text{if} \quad s_t \neq s_{t-1}\end{cases}\\ \mathcal{L}^{\text{init}}(s_0) &= 0\\ \mathcal{L}^{\text{mode}}(s_t) &= 0\end{aligned}$$

where τ is treated as a hyper-parameter to be tuned. Algorithm 1 is executed $N = 5$ times from different random initial guesses. Each execution is limited to $k_{\max} = 1000$ iterations.

We run Algorithm 1 on the training data for different magnitudes σ_y of output noise and values of the hyper-parameter τ. The resulting model coefficients $\Theta^\star$ are then used in Algorithm 3 for recursive inference on the production data. For assessing the quality of inference we use the true loss L^{true} defined in (1) with $\ell^{\text{true}}(\hat{y}_t, \tilde{y}_t) = \|\hat{y}_t - \tilde{y}_t\|_2^2$. In addition, assuming the latent modes $\tilde{s}_t$ are available only for validation purposes, we consider the following mode-mismatch figure

$$\ell_s^{\text{true}} = \frac{100}{\tilde{T}}\sum_{t=1}^{\tilde{T}} \delta_{\hat{s}_t, \tilde{s}_t} \tag{38}$$

where $\delta_{i,j}$ is the Kronecker delta function. The results are summarized in Fig. 1.

By recalling (22) and (14c), in order to minimize $-\log p(y_t|x_t, \theta)$ $-\log \pi_{s_t, s_{t-1}}$ one should set $\ell(x_t, y_t, \theta_{s_t}) = \frac{1}{2\sigma_y^2}$ and $\mathcal{L}^{\text{trans}}(s_t, s_{t-1}) = -\log \pi_{s_t,s_{t-1}}$, or equivalently $\ell(x_t, y_t, \theta_{s_t}) = 1$, $\mathcal{L}^{\text{trans}}(s_t, s_{t-1}) = -\tau^\star \log \pi_{s_t,s_{t-1}}$ with $\tau^\star = 2\sigma_y^2$. Fig. 1 also reports the value of $\tau^\star$ (dashed line) corresponding to different values of σ_y. As expected, the best value for τ obtained by cross validation, corresponding to the minimum of the plotted curves, corresponds to the theoretical one $\tau^\star$ that would be obtained if σ_y were known. For large values of τ the percentage of mode mismatch becomes close to $\frac{K-1}{K} \approx 66\%$ (not shown in the figure), that is the value one gets when the mode $\hat{s}_t$ is assigned randomly. The average CPU time for executing Algorithm 1 is 342 ms, with the longest execution requiring 93 iterations. Algorithm 3 requires 0.89 µs per data point on average to make one-step ahead inference.

Fig. 2 shows the percentage of misclassified modes when pure mode estimation, as presented in Section 4.2.4, is employed instead of one-step ahead prediction. In this case, the latent mode $\hat{s}_t$ is reconstructed based not only on the observations $\tilde{x}_1, \ldots, \tilde{x}_t$, $\tilde{y}_1, \ldots, \tilde{y}_{t-1}$ but also $\tilde{y}_t$, using Algorithm 1 with (29b) replaced by (31). As expected, compared to Fig. 1, taking into account the current observation $\tilde{y}_t$ in estimating $\hat{s}_t$ reduces the number of misclassified modes.

Finally, the *Expectation–Maximization* algorithm for HMM regression in Fridman (1994) is implemented and compared with our method, with the hyper-parameter τ chosen, for each different σ_y, as the best value observed in cross-validation. In EM the sequence of latent modes is inferred in a batch way from the production dataset by using Viterbi algorithm (Viterbi, 2010). In our approach, the mode sequence is estimated using Algorithm 1 with (29b) replaced by (31). Table 1 summarizes the results of the comparison, showing that our approach provides a slightly better, although very similar, mode mismatch figure ℓ_s^{true} (38).

5.2. Jump binary classification

We consider $T = 10000$ training data and $\tilde{T} = 10000$ production data generated by the following jump linear model with $K = 3$ modes

$$y_t = \text{sign}(\theta_{s_t} x_t + \zeta_t)$$

with

$$\begin{bmatrix}\theta_1 & \theta_2 & \theta_3\end{bmatrix} = \begin{bmatrix} -1 & -1 & -1\\ 1.1812 & -0.5587 & 0.8003\\ -0.7585 & 0.1784 & -1.5094\\ -1.1096 & -0.1969 & 0.8759\\ -0.8456 & 0.5864 & -0.2428\\ -0.5727 & 0.8759 & 0.6037\\ -0.5587 & -0.2428 & 1.7813\\ 0.1784 & 0.1668 & 1.7737\end{bmatrix}$$

and $y_t \in \{-1, 1\}$, $x_t \in \mathbb{R}^8$, $x_{t,i} \sim \mathcal{N}(0, \sigma_x^2)$ for all $i = 1, \ldots, 8$ with $\sigma_x = 10$, $\zeta_t \sim \mathcal{N}(0, \sigma_y^2)$, $\sigma_y = 0.1$. The true mode s_t changes every 500 samples during the generation of the data, covering all modes.

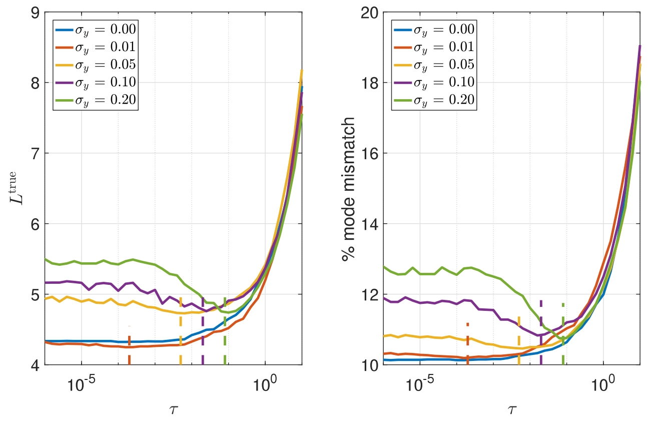


Fig. 1. Jump linear model fit and validation using recursive one-step ahead prediction: true loss $\tilde{L}^{\text{true}}$ (left) and mode mismatch ℓ_s^{true} (right), optimal theoretical value $\tau^\star = 2\sigma_y^2$ (dashed line).

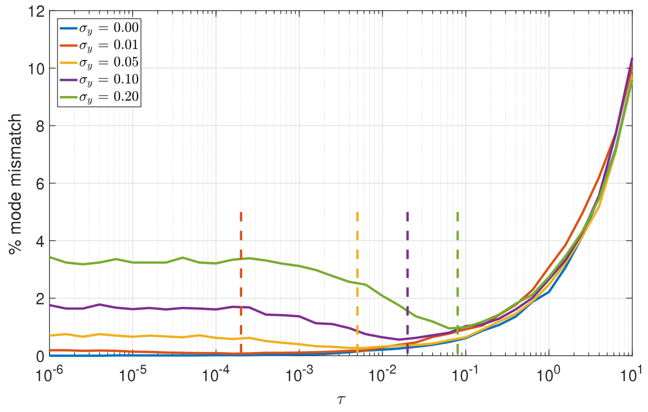


Fig. 2. Jump linear model fit and validation using pure mode estimation: mode mismatch ℓ_s^{true} (right), optimal theoretical value $\tau^\star = 2\sigma_y^2$ (dashed line).

Table 1
Jump linear model validation, smoothing results: mode mismatch ℓ_s^{true} achieved by the Expectation–Maximization (EM) algorithm for HMM regression (Fridman, 1994) and by the approach discussed in this paper (Algorithms 1 and 2).

	ℓ_s^{true} %	
	EM	Algorithms 1–2
$\sigma_y = 0.00$	0.00	0.00
$\sigma_y = 0.01$	0.12	0.06
$\sigma_y = 0.05$	0.40	0.23
$\sigma_y = 0.10$	1.24	0.59
$\sigma_y = 0.20$	1.84	0.88

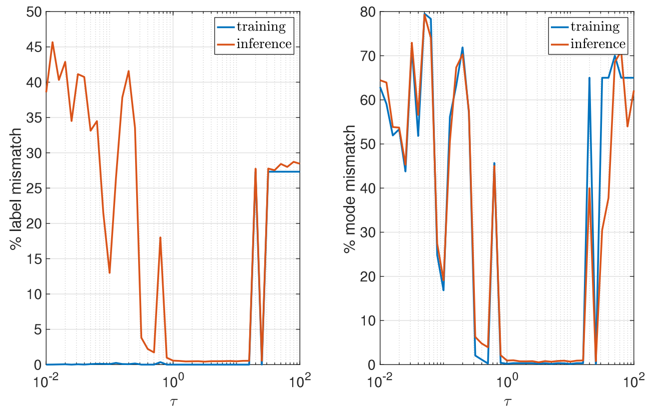


Fig. 3. Jump binary classifier: percentage of misclassified labels (left) and mode mismatch (right) on training and production data.

We want to train a binary classifier defined by the following losses

$$\begin{aligned}\ell(x_t, y_t, \theta_{s_t}) &= \max(1 - y_t\theta_{s_t}' x_t, 0)\\ r(\theta_k) &= 10^{-5}\|\theta_k\|_2^2\\ \mathcal{L}^{\text{trans}}(s_t, s_{t-1}) &= \tau(1-\delta_{i,j})\\ \mathcal{L}^{\text{init}}(s_0) &= 0\\ \mathcal{L}^{\text{mode}}(s_t) &= 0.\end{aligned}$$

Fig. 3 shows the results obtained for different values of the hyper-parameter τ. We consider the mismatch between the true labels y_t and the estimated labels $y_t^\star = \text{sign}((\theta_{s_t^\star}^\star)' x_t)$ returned by Algorithm 1 on the training data, and also between the true labels $\tilde{y}_t$ and the labels $\hat{y}_t = \text{sign}((\theta_{\hat{s}_t}^\star)'\tilde{x}_t)$ returned by Algorithm 3 on the production data. In addition, we consider the detection of model changes, comparing the true modes s_t, $\tilde{s}_t$ and their corresponding estimates $s_t^\star$, $\hat{s}_t$. Good values for τ are in the range 1–10, for which model changes are correctly detected on both training and production data.

The CPU time for executing Algorithm 1 ranges between 4.24 and 80.76 s, with Step 1 computed using the QP solver of GUROBI 7.02 (Gurobi Optimization, Inc., 2017). Algorithm 1 requires between 15 and 199 iterations. Algorithm 3 takes an average of 0.57 μs per data point for inference.

5.3. Markov jump linear dynamical system

We consider the Markov jump linear dynamical system with $K = 4$ modes

$$x_{t+1} = A_{s_t}x_t + B_{s_t}u_t + \zeta_t$$

where $x_t, \zeta_t \in \mathbb{R}^8$, $u_t \in \mathbb{R}^2$ takes random values in $\{-1, 1\}^2$, $\zeta_t^j \sim \mathcal{N}(0, \sigma_y)$ for all $j = 1, \dots, 8$, the matrix pairs (A_i, B_i) are random stable systems for all $i = 1, \dots, K$. The modes s_t are randomly generated according to an (unknown) transition probability matrix $\Pi \in \mathbb{R}^{4\times4}$. The goal is to estimate the system matrices (A_i, B_i), $i = 1, \dots, K$, and the transition probability Π from $T = 50000$ data pairs (x_t, u_t) available for training, and validate the results on $\tilde{T} = 50000$ new samples.

Algorithm 1 is executed $N = 5$ times on the training data with loss function $\|x_{t+1} - A_{s_t}x_t - B_{s_t}u_t\|_2^2$, uniform mode transition loss $\mathcal{L}^{\text{trans}}(i, j) = \tau$, zero losses $\mathcal{L}^{\text{init}}$, $\mathcal{L}^{\text{mode}}$, and regularization $r(\theta_k) = 10^{-5}\|\theta_k\|_2^2$. Note that, since the output sample $y_t = x_{t+1}$ is multidimensional, we cannot train a model for each component of y independently, as they are linked by the common mode s_t.

After training and before performing inference via (29), the transition probability matrix $\hat{\Pi}$ is reconstructed using (35) on the estimated mode sequence $S^\star$ returned by Algorithm 1.

The results are reported in Fig. 4. The coefficients of the models (A_i, B_i) are estimated with an error of 10^{-8} ($\sigma_y = 0$), 10^{-3} ($\sigma_y = 0.01$), and 10^{-2} ($\sigma_y = 0.05$), respectively, while the transition probability matrix with error $\|\Pi - \hat{\Pi}\|_2$ of 0.01 for all values of σ_y. The average CPU time for executing Algorithm 1 is 68 ms (the longest execution takes 134 iterations), while Algorithm 3 takes 0.57 μs per data point on average for inference.

5.4. Experimental example: PWA dynamical model

We consider the problem of modeling the dynamics of a placement process of electronic components in a pick-and-place machine described in Juloski, Heemels, and Ferrari-Trecate (2004). The process consists of a mounting head carrying the electronic component which is placed on a printed circuit board, and then released. This process is characterized by two main operating modes, the *free* and the *impact mode*. In free mode the machine carries the electronic component in an unconstrained environment, i.e., without being in contact with the circuit board. In impact mode the mounting head moves in contact with the circuit board. Because of its switching behavior, this process has been used as a benchmark to assess the performance of several identification algorithms for hybrid dynamical systems (Bemporad et al., 2005; Juloski Heemels, Ferrari-Trecate, Vidal et al., 2005; Ohlsson & Ljung, 2013).

A data record over an interval of 15 s is gathered from an experimental bench (see Juloski, Heemels, & Ferrari-Trecate, 2004 for details), with a sampling frequency of 400 Hz. We denote by u the voltage applied to the motor driving the mounting head and by y the vertical position of the mounting head. The data record is split in two disjoint subsets: a training set with $T = 4800$ samples, which consist of the observations gathered in the first 12 s of the experiments, and a test set with $\tilde{T} = 1200$ samples, which consist of the observations gathered in the last 3 s.

We want to fit a PWA model as defined in (10)–(11) with $K = 2$ discrete modes. Each regression model is given by $y_t = \theta_{y,s_t}' \begin{bmatrix} x_t \\ 1 \end{bmatrix}$, where $x_t = [y_{t-1}\ y_{t-2}\ u_{t-1}\ u_{t-2}]'$.

Algorithm 1 is executed $N = 5$ times on the first 4400 samples of the training set with loss function $\ell(x_t, y_t, \theta_{s_t})$ as in (10), mode sequence loss $\mathcal{L} = 0$ and regularization $r(\Theta) = \sum_{k=1}^{K} r(\theta_{y,k})$, with $r(\theta_{y,k}) = 10^{-5}\|\theta_{y,k}\|_2^2$. The remaining 400 samples are used to tune the hyper-parameter ρ in (10), leading to an optimal value $\rho = 2.15 \cdot 10^{-4}$. The average CPU time for executing Algorithm 1 for a fixed value of ρ is 156 ms. In the worst case, Algorithm 1 terminates after 25 iterations.

Fig. 5 shows the outputs $\tilde{y}_t$ collected from the production dataset, the open-loop prediction $\hat{y}_t$ of the output reconstructed by feeding the same inputs $\tilde{u}_t$ to the estimated PWA model, and the sequence of estimated modes $\hat{s}_t$. The resulting *best fit rate* $\text{BFR} = 100\left(1 - \sqrt{\frac{\sum_{t=1}^{\tilde{T}}\|\tilde{y}_t - \hat{y}_t\|^2}{\sum_{t=1}^{\tilde{T}}\|\tilde{y}_t - \bar{y}\|^2}}\right)\%$ is equal to 83%, where $\bar{y}$ denotes the average of the outputs $\tilde{y}_1, \dots, \tilde{y}_{\tilde{T}}$. The evolution of the reconstructed mode sequence shows that mode 1 is active at, roughly, $y \geq 15$. From the physical knowledge of the system and of the experimental setup, we can associate modes 1 and 2 to the impact and to the free mode, respectively.

For comparison, the same fitting problem is solved by using the cluster-based algorithm for PWA regression in Ferrari-Trecate et al. (2003), using the *Hybrid Identification Toolbox* (HIT) toolbox (Ferrari-Trecate, 2005). The *Proximal Support Vector Classifier* (PSVC) (Fung & Mangasarian, 2005) is employed to compute the polyhedral partition of the regressor space. The same training and production datasets are considered, with the hyper-parameters

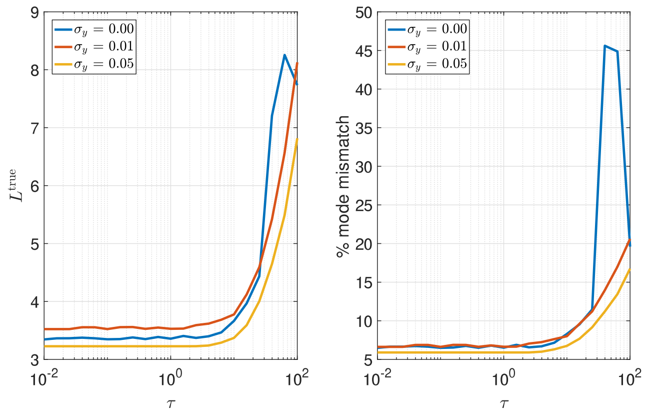


Fig. 4. Markov jump linear dynamical system: true loss $\tilde{L}^{\text{true}}$ (left) and mode mismatch ℓ_s^{true} (right).

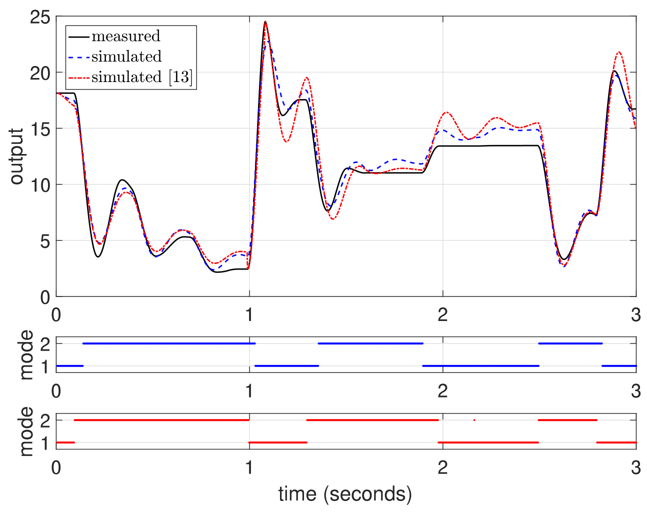


Fig. 5. Pick-and-place machine: simulated and actual output (top), mode sequence estimated using our approach (middle), and using the cluster-based algorithm (Ferrari-Trecate et al., 2003) (bottom).

characterizing the PWA regression algorithm (Ferrari-Trecate et al., 2003) tuned via cross-validation on the last 400 samples of the training set. The open-loop predicted output $\hat{y}_t$ is shown in Fig. 5, along with the estimated mode sequence. The achieved BFR is 75%, which is slightly worse than what we obtained using our approach (83%), although very similar. The average CPU time required by the HIT toolbox to train the PWA model for fixed hyper-parameters is 159 s, which is about 1000×slower than the method proposed in this paper.

6. Conclusions

We have presented a new framework for fitting a jump model to a temporal sequence of data. Overall, the approach is able to fit models with latent discrete variables and provides an efficient (and more general) alternative to existing methods, such as the expectation–maximization algorithm for the estimation of hidden Markov models and cluster-based heuristics for the identification of switching and PWA models.

A main strength of the approach is its versatility in describing a large class of parametric models, as the shape of the model and the way it jumps depends on the shape of the loss functions used for fitting the model parameters and for inference. Such a generality of the approach stimulates future research to address auto-tuning strategies, where the loss functions are chosen automatically from data. We expect that several instances of our approach will be investigated, using different loss functions and in various applications.

Another strength of the proposed approach is its numerical efficiency, due to using a simple coordinate-descent optimization algorithm for fitting model parameters and a recursive formulation for inferring outputs and latent modes. Although there is no guarantee of converging to the global optimum, numerical evidence shows the effectiveness of the method.

Future research will also address an incremental version of the fitting algorithm, so to update models and infer output/mode pairs when data are streaming on-line.

References

Baum, L. E., Petrie, T., Soules, G., & Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, *41*(1), 164–171.

Bellman, R. (1957). *Dynamic programming*. Princeton, NJ, USA: Princeton University Press.

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

Bemporad, A., & Giorgetti, N. (2006). Logic-based methods for optimal control of hybrid systems. *IEEE Transactions on Automatic Control*, *51*(6), 963–976.

Bemporad, A., Roll, J., Ljung, L. (2001). Identification of hybrid systems via mixed-integer programming. In *Proc. 40th IEEE conf. on decision and control* (pp. 786–792).

Bennett, K. P., & Mangasarian, O. L. (1994). Multicategory discrimination via linear programming. *Optimization Methods & Software*, *3*, 27–39.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. New York, NY, USA: Cambridge University Press, http://www.stanford.edu/~boyd/cvxbook.html.

Breschi, V., Bemporad, A., Piga, D., (2016a) Identification of hybrid and linear parameter varying models via recursive piecewise affine regression and discrimination. In *European control conference* (pp. 2632–2637).

Breschi, V., Piga, D., & Bemporad, A. (2016b). Piecewise affine regression via recursive multiple least squares and multicategory discrimination. *Automatica*, *73*, 155–162.

Chan, A. B., & Vasconcelos, N. (2008). Modeling, clustering, and segmenting video with mixtures of dynamic textures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(5), 909–926.

Costa, O. L. V., Fragoso, M. D., & Marques, R. P. (2006). *Discrete-time Markov jump linear systems*. Springer Science & Business Media.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, *39*(1), 1–38.

Ferrari-Trecate, G. (2005). Hybrid identification toolbox (hit).

Ferrari-Trecate, G., Muselli, M., Liberati, D., & Morari, M. (2003). A clustering technique for the identification of piecewise affine systems. *Automatica*, *39*(2), 205–217.

Fridman, M. (1994). *Hidden Markov model regression, Technical report*. Minneapolis, MN: Institute of Mathematics, University of Minnesota.

Fung, G. M., & Mangasarian, O. L. (2005). Multicategory proximal support vector machine classifiers. *Machine Learning*, *59*, 77–97.

Guidolin, M. (2011). Markov switching models in empirical finance. In *Missing data methods: Time-series methods and applications* (pp. 1–86). Emerald Group Publishing Limited.

Gurobi Optimization, Inc. (2017) Gurobi optimizer reference manual. http://www.gurobi.com.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning* (2nd ed.). New York: Springer.

Juloski, A. L., Heemels, W. P. M. H., & Ferrari-Trecate, G. (2004). Data-based hybrid modelling of the component placement process in pick-and-place machines. *Control Engineering Practice*, *12*(10), 1241–1252.

Juloski, A. L., Heemels, W. P. M. H., Ferrari-Trecate, G., Vidal, R., Paoletti, S., & Niessen, J. H. G. (2005). Comparison of four procedures for the identification of hybrid systems. *Lecture Notes in Computer Science*, *3414*, 354–369.

Juloski, A., Weiland, S., Heemels, M. (2004). A Bayesian approach to identification of hybrid systems. In *Proc. 43th IEEE conf. on decision and control.*

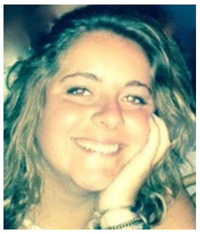
Manning, C. D., Raghavan, P., Schütze, H. (2008). *Introduction to information retrieval. Vol. 1*.

Oh, S. M., Rehg, J. M., Balch, T., & Dellaert, F. (2008). Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *International Journal of Computer Vision*, *77*(1), 103–124.

Ohlsson, H., & Ljung, L. (2013). Identification of switched linear regression models using sum-of-norms regularization. *Automatica*, *49*(4), 1045–1050.

Ozay, N., Sznaier, M. M., Lagoa, C. (2010) Model (in)validation of switched arx systems with unknown switches and its application to activity monitoring. In *49th IEEE conference on decision and control* (pp. 7624–7630).
Pavlovic, V., Rehg, J. M., & MacCormick, J. (2001). Learning switching linear models of human motion. In *Advances in neural information processing systems* (pp. 981–987).
Pillonetto, G. (2016). A new kernel-based approach to hybrid system identification. *Automatica*, *70*, 21–31.
Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.
Schuller, B., Wöllmer, M., Moosmayr, T., Ruske, G., & Rigoll, G. (2008). Switching linear dynamic models for noise robust in-car speech recognition. *Pattern Recognition*, 244–253.
Timmermann, A. (2015). Markov switching models in finance. In *Wiley encyclopedia of management. Vol. 4*. John Wiley & Sons, Ltd.
Vidal, R., Chiuso, A., Soatto, S. (2002) Observability and identifiability of jump linear systems. In *Proc. 41st IEEE conference on decision and control. Vol. 4* (pp. 3614–3619).
Viterbi, A. J. (2010). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. In *The foundations of the digital wireless world: Selected works of AJ Viterbi* (pp. 41–50).

Alberto Bemporad received his master's degree in Electrical Engineering in 1993 and his Ph.D. in Control Engineering in 1997 from the University of Florence, Italy. In 1996/97 he was with the Center for Robotics and Automation, Department of Systems Science & Mathematics, Washington University, St. Louis. In 1997–1999 he held a postdoctoral position at the Automatic Control Laboratory, ETH Zurich, Switzerland, where he collaborated as a senior researcher until 2002. In 1999–2009 he was with the Department of Information Engineering of the University of Siena, Italy, becoming an associate professor in 2005. In 2010–2011 he was with the Department of Mechanical and Structural Engineering of the University of Trento, Italy. Since 2011 he is full professor at the IMT School for Advanced Studies Lucca. He spent visiting periods at Stanford University, University of Michigan, and Zhejiang University. In 2011 he cofounded ODYS S.r.l., a company specialized in developing model predictive control systems for industrial production. His current research interests are in the areas of model predictive control, quadratic optimization, system identification, and automotive control.


Valentina Breschi received her Bachelor's degree in Electronics/Telecommunications Engineering and her Master's degree in Electric/Automation Engineering both from the University of Florence (Italy) in 2011 and 2014, respectively. She received her Ph.D. in Control Systems from IMT School for Advanced Studies Lucca (Italy) in 2018. She is currently a Postdoctoral researcher at Politecnico di Milano (Italy). Her research interests include hybrid and stochastic systems identification.

Dario Piga received his Ph.D. in Systems Engineering from the Politecnico di Torino (Italy) in 2012. He was a Postdoctoral Researcher at the Delft University of Technology (The Netherlands) in 2012 and at the Eindhoven University of Technology (The Netherlands) in 2013. From 2014 to early 2017 he was Assistant Professorat the IMT School for Advanced Studies Lucca (Italy) and since March 2017 he has been tenure-track Researcher at the IDSIA Dalle Molle Institute for Artificial Intelligence in Lugano (Switzerland). His main research interests include system identification, robust control, Bayesian filtering and non-convex optimization, with applications to process control and smart manufacturing.

Stephen P. Boyd is the Samsung Professor of Engineering, and Professor of Electrical Engineering in the Information Systems Laboratory at Stanford University, with courtesy appointments in Computer Science and Management Science and Engineering. He received the A.B. degree in Mathematics from Harvard University in 1980, and the Ph.D. in Electrical Engineering and Computer Science from the University of California, Berkeley, in 1985, and then joined the faculty at Stanford. His current research focus is on convex optimization applications in control, signal processing, machine learning, and finance.