Fitting Jump Models

A. Bemporad†, V. Breschi†, D. Piga‡, and S. Boyd§
†IMT School for Advanced Studies Lucca, Piazza San Francesco 19, 55100 Lucca, Italy. Email: {alberto.bemporad,valentina.breschi}@imtlucca.it
‡Dalle Molle Institute for Artificial Intelligence Research - USI/SUPSI, Galleria 2, Via Cantonale 2c, CH-6928 Manno, Switzerland. Email: dario.piga@supsi.ch
§Department of Electrical Engineering, Stanford University, Stanford CA 94305, USA. Email: boyd@stanford.edu

November 28, 2017

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 determine 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_θ ∥Aθ − b∥_2

each row of A and b is associated with a data-point, but clearly the solution θ^ is the same no matter how the rows of A and b are ordered. In systems identification of dynamical models, 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 change 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 the data 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 [19, 9] to speech recognition [23, 24], asset-price models in finance [25, 14], human action classification [22, 21], 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 data set, where each model is active based on the location of the input sample in a polyhedral partition of the input space [12, 8]. 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 [4, 7]. Hidden Markov models (HMMs) treat instead the mode as a stochastic discrete variable, whose temporal dynamics is described by a Markov chain [23]. Natural extensions of hidden Markov models consider the cases in which each mode is associated with a linear function of the input [13, 10, 20]. Hidden Markov models are usually trained using the Baum-Welch algorithm [1], a forward-backward version of the more general Expectation Maximization (EM) algorithm [11].

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 [16, 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 data sets.

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

\[
L_{\text{true}} = \frac{1}{\tilde{T}} \sum_{t=1}^{\tilde{T}} \ell_{\text{true}}(\hat{y}_t, \tilde{y}_t)
\]

(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

\[
J(X, Y, \theta) = \sum_{t=1}^{T} \ell(x_t, y_t, \theta) + r(\theta)
\]

(2a)

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

\[
\theta^* = \arg \min_{\theta} J(X, Y, \theta)
\]

(2b)

be the optimal model parameter. By fixing \(\theta = \theta^*\) we get the following regression model

\[
\hat{y}_t = \arg \min_{y_t} J(X, Y, \theta^*) = \arg \min_{y_t} \ell(x_t, y_t, \theta^*) =: \varphi(x_t)
\]

(2c)
where we have exploited the separability of the loss $J$ and set $\varphi : X \to Y$ as the regression model, with ties in the arg min broken arbitrarily.

Model $\mathcal{E}$ can be enriched by adding output information sets $Y_t \subseteq Y$ that augment the information that is available about $y_t$,

$$\hat{y}_t = \arg \min_{y \in Y_t} \ell(x, y, \theta^*)$$

where $Y_t = Y$ if no extra information on $y_t$ is given. For example, if we know a priori that $y_t \geq 0$ we can set $Y_t$ 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 $\theta_{s_t}$ that is active at step $t$. Fitting a $K$-model on the training data set $(X, Y)$, entails choosing the $K$ models by minimizing

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

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

$$(\hat{y}_t, \hat{s}_t) = \arg \min_{y_t, s_t} \ell(x_t, y_t, \theta_{s_t}^*)$$

For example, the loss function

$$\ell(x_t, y_t, \theta_{s_t}) = \|y_t - \theta'_{s_t} x_t\|_2^2$$

corresponds to $K$ possible linear models $\hat{y}_t = \theta'_{s_t} x_t$. By setting

$$r(\theta) = 0, \quad \ell(x_t, y_t, \theta_{s_t}) = \|y_t - \theta_{s_t}\|_2^2 + \|x_t - \theta_{s_t}\|_2^2$$

and $y_t = x_t$, model $\mathcal{E}$ is the standard $K$-means model $\mathcal{M}$, where $\theta_s$ are the centroids of the clusters. In this case, $\hat{s}_t$ denotes the index of the centroid $\theta_{s_t}^*$ which is closest to $x_t$, and $\hat{y}_t = \theta_{s_t}^*$ is the best estimate of $x_t$ based on the regression model defined by $\mathcal{M}$. More generally,

$$\ell(x_t, y_t, \theta_{s_t}) = \|y_t - \theta'_{y,s_t} \tau^*_t\|_2^2 + \rho\|x_t - \theta_{x,s_t}\|_2^2$$

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

$$\hat{s}_t = \arg \min_{s_t} \|x_t - \theta_{x,s_t}^*\|_2^2, \quad \hat{y}_t = (\theta_{y,s_t}^*)' \tau^*_t.$$
A more general PWA model can be defined by considering a piecewise linear separation function \( \vartheta : X \to \mathbb{R} \)
\[
\vartheta(x) = \max_{s} \{ \theta'_{x,s}[x_1] \}
\]
and setting
\[
\ell(x_t, y_t, \theta_{s_t}) = \|y_t - \theta'_{y,s_t}[x_1]\|^2 + \rho \sum_{j=1}^{K} \max \{0, (\theta_{x,j} - \theta_{x,s_t})' [x_1] + 1\}^2.
\]
In this case it is immediate to verify that the regression model is
\[
\hat{s}_t = \arg \max_s \{ (\theta_{x,t,s}^{*})' [x_1] \}, \quad \hat{y}_t = (\theta_{y,s_t}^{*})' [x_1].
\]

### 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
\[
J(X, Y, \Theta, S) = \sum_{t=1}^{T} \ell(x_t, y_t, \theta_{s_t}) + \sum_{k=1}^{K} r(\theta_k) + \mathcal{L}(S),
\]
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
\[
\mathcal{L}(S) = \mathcal{L}_{\text{init}}(s_0) + \sum_{t=1}^{T} \mathcal{L}_{\text{mode}}(s_t) + \sum_{t=1}^{T} \mathcal{L}_{\text{trans}}(s_t, s_{t-1})
\]
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
\[
J(X, Y, \Theta, S) = \ell(X, Y, \Theta, S) + r(\Theta) + \mathcal{L}(S)
\]
where
\[
\ell(X, Y, \Theta, S) = \sum_{t=1}^{T} \ell(x_t, y_t, \theta_{s_t}), \quad r(\Theta) = \sum_{k=1}^{K} r(\theta_k).
\]

As with any model, the choice of the fitting objective \( \mathcal{L} \) 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 (13) 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 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 \( L^{\text{init}}, L^{\text{mode}}, L^{\text{trans}} \) defining the mode sequence loss \( L \) in (13a).

As mentioned above, the case \( L(S) = 0 \) leads to a \( K \)-model. By choosing \( L^{\text{trans}}(i,j) = \lambda \) for all \( i \neq j \), \( L^{\text{mode}}(i) = L^{\text{trans}}(i,i) = 0 \), 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 \( \lambda \) for all transitions makes the fitting problem exhibit multiple solutions, as indexes \( i, j \) can be arbitrarily permuted. The mode loss \( L^{\text{mode}} \) can be used to break such symmetries. For example, smaller values for \( s_t \) will be preferred by making \( L^{\text{mode}}(i) < L^{\text{mode}}(j) \) for \( i < j \). The shape of the increasing finite sequence \( \{L^{\text{mode}}(i)\}_{i=1}^{K} \) can be used to reduce the number of possible modes: larger increasing values of \( L^{\text{mode}}(i) \) will discourage the use of an increasing number of modes.

The initial mode cost \( L^{\text{init}} \) summarizes prior knowledge about the initial mode \( s_0 \). For example, \( L^{\text{init}}(s_0) = 0 \) 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 \( L^{\text{init}}(s_0) = 0 \) for \( s_0 = j \) and \( +\infty \) otherwise.

Next Section 3 suggests criteria for choosing \( L \) in case statistical assumptions about the underlying process that generates \( s_t \) are available. Alternative criteria are discussed in Section 4.4 for choosing \( L \) 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 \( \Theta \) 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 \( p(y_t|x_t, \theta_s) \) is the likelihood of the outcome \( y_t \) given \( x_t \) and \( \theta_s \);

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
\[
p(s_t|s_{t-1}) = \pi_{s_t,s_{t-1}} \quad \text{(Markov property)};
\]

A5. The initial mode \( s_0 \) has probability \( p(s_0) = \pi_{s_0} \).

**Proposition 1** Let Assumptions A1-A5 be satisfied and define

\[
\begin{align*}
\ell(x_t, y_t, \theta_s) &= -\log p(y_t|x_t, \theta_s) \quad (14a) \\
r(\theta_k) &= -\log p(\theta_k) \quad (14b) \\
L_{\text{trans}}(s_t, s_{t-1}) &= -\log \pi_{s_t,s_{t-1}} \quad (14c) \\
L_{\text{init}}(s_0) &= -\log \pi_{s_0} \quad (14d) \\
L_{\text{mode}}(s_t) &= 0. \quad (14e)
\end{align*}
\]

Then minimizing \( J(X,Y,\Theta,S) \) as defined in (12)–(14) with respect to \( \Theta \) 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
\[
p(S) = p(s_0) \prod_{t=1}^{T} p(s_t|s_{t-1}). \quad (15)
\]

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

\[
p(Y,S,\Theta|X) = p(\Theta|X)p(Y,S|X,\Theta) = p(\Theta|X)p(S|X,\Theta)p(Y|S,X,\Theta)
\]
\[
= p(\Theta)p(S)p(Y|S,X,\Theta)
\]
\[
= \prod_{k=1}^{K} p(\theta_k)p(s_0) \prod_{t=1}^{T} p(s_t|s_{t-1})p(y_t|x_t, \theta_s)
\]
whose logarithm is
\[
\log p(Y,S,\Theta|X) = \sum_{k=1}^{K} \log p(\theta_k) + \log p(s_0) + \sum_{t=1}^{T} \log p(s_t|s_{t-1}) + \sum_{t=1}^{T} \log p(y_t|x_t, \theta_s). \quad (16)
\]

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

It is also possible to give an inverse result, namely a statistical interpretation of minimizing a given generic $J(X,Y,\Theta,S)$ defined as in (13). Define the following probabilities

$$p(y_t|x_t,\theta_{st}) = \frac{e^{-\ell(x_t,y_t,\theta_{st})}}{\nu(\theta_{st},x_t)}$$ \hspace{1cm} (17a)$$

$$p(\Theta|X) = p(\Theta) = \frac{e^{-r(\Theta)}}{\int_{\mathbb{R}^K} e^{-r(\Theta)}d\Theta}$$ \hspace{1cm} (17b)$$

$$p(S|\Theta, X) = \frac{e^{-\bar{L}(S,\Theta,X)}}{\sum_{S \in K} e^{-\bar{L}(S,\Theta,X)}}$$ \hspace{1cm} (17c)$$

where

$$\nu(\theta_{st},x_t) = \int_Y e^{-\ell(x_t,y,\theta_{st})}dy$$ \hspace{1cm} (18a)$$

$$\bar{L}(S,\Theta,X) = \mathcal{L}(S) - \sum_{t=1}^T \log \nu(\theta_{st},x_t).$$ \hspace{1cm} (18b)$$

Under the assumption that the outputs $y_t$ are independent given $(S,X,\Theta)$, we have that $p(Y|S,X,\Theta) = \prod_{t=1}^T p(y_t|x_t,\theta_{st})$. Furthermore, since

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

we get

$$\arg \max \log p(Y,S,\Theta|X) = \arg \min J(X,Y,\Theta,S).$$ \hspace{1cm} (19)$$

Note that the standard quadratic regularization $r(\Theta) = \sum_{k=1}^K r(\theta_k)$, with $r(\theta_k) = \rho\|\theta_k\|^2_2$, leads to $p(\theta_k) = ce^{-\frac{\|\theta_k\|^2}{2\sigma^2}}$, where $\sigma_\theta = \sqrt{\frac{1}{2\rho}}$, which amounts to assuming $\theta_k \sim N(0,\sigma^2_\theta I)$. Similarly, the quadratic penalty on the prediction error

$$\ell(x_t,y_t,\theta_{st}) = c\|y_t - \theta_{s,t}^\prime x_t\|^2_2$$ \hspace{1cm} (20)$$

leads to assuming that $y_t \sim N(\theta_{s,t}^\prime x_t,\sigma^2_y I)$, where $\sigma_y = \sqrt{\frac{1}{2c}}$.

The special cases highlighted in Section 2.3.1 also have a statistical interpretation. Let us assume for simplicity that $\nu(\theta_{st},x_t) = \frac{1}{T}$ in (18a), so that we can replace $\bar{L}(S,\Theta,X) = \mathcal{L}(S)$ in (18b) without changing the optimizer of $J(X,Y,\Theta,S)$ in (19), and also assume that $\mathcal{L}_{\text{init}}(s) = \mathcal{L}_{\text{mode}}(s)$.

When $\mathcal{L}_{\text{trans}} = 0$ we can set

$$p(S|\Theta, X) = p(S) = \prod_{t=0}^T \frac{e^{-\mathcal{L}_{\text{mode}}(s_t)}}{\sum_{k=1}^K e^{-\mathcal{L}_{\text{mode}}(k)}}$$
Algorithm 1 Jump model fitting

**Input:** Training data set $X = (x_1, \ldots, x_T)$, $Y = (y_1, \ldots, y_T)$, number $K$ of models, initial mode sequence $S^0 = \{s^0_0, \ldots, s^0_T\}$.

1. **iterate for** $k = 1, \ldots$
   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 $\Theta^* = \Theta^k$ and mode sequence $S^* = S^k$.

1 Algorithms

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

4.1 Model fitting

Given a training sequence $X = (x_1, \ldots, x_T)$ of inputs and $Y = (y_1, \ldots, y_T)$ of outputs, for fitting a jump $K$-model we need to attempt minimizing the cost $J(X, Y, \Theta, S)$ with respect to $\Theta$ and $S$. A simple algorithm to solve this problem is Algorithm 1, a coordinate descent algorithm that alternates minimization with respect to $\Theta$ and $S$. If $\ell$ and $r$ are convex functions, Step 1.1 can be solved globally (up to the desired precision) by standard convex programming [6]. Step 1.2 can be solved by standard discrete dynamic programming (DP) [2]. The latter 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

$$M(s, T) = \mathcal{L}^{\text{mode}}(s) + \ell(x_T, y_T, \theta_s) \quad (21a)$$

$$U_{s,t} = \arg \min_j \{ M(j, t + 1) + \mathcal{L}^{\text{trans}}(j, s) \}, \ t = 1, \ldots, T - 1 \quad (21b)$$

$$M(s, t) = \mathcal{L}^{\text{mode}}(s) + \ell(x_t, y_t, \theta_s) + M(U_{s,t}, t + 1) + \mathcal{L}^{\text{trans}}(U_{s,t}, s) \quad (21c)$$

$$M(s, 0) = \mathcal{L}^{\text{init}}(s) + \min_j\{M(j, 1) + \mathcal{L}^{\text{trans}}(j, s)\} \quad (21d)$$

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

$$s_0 = \arg \min_j M(j, 0) \quad (21e)$$

$$s_t = U_{s_{t-1}, t}, \ t = 1, \ldots, T. \quad (21f)$$

Note that if the time order of operations in (21) is reversed, the DP iterations (21) become Viterbi algorithm [23, p. 264]:

$$M(s, 0) = \mathcal{L}^{\text{init}}(s) \quad (22a)$$

$$U_{s,t} = \arg \min_j \{ M(j, t - 1) + \mathcal{L}^{\text{trans}}(j, s) \}, \ t = 1, \ldots, T \quad (22b)$$

$$M(s, t) = \mathcal{L}^{\text{mode}}(s) + \ell(x_t, y_t, \theta_s) + M(U_{s,t}, t - 1) + \mathcal{L}^{\text{trans}}(U_{s,t}, s) \quad (22c)$$

followed by the backwards iterations

$$s_T = \arg \min_j M(j, T) \quad (22d)$$

$$s_t = U_{s_{t+1}, t}, \ t = 0, \ldots, T - 1. \quad (22e)$$

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.

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 $\theta_s$, 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 data set. If in addition $\ell$ and $r$ are specified as in (7) 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 $(x_1, \ldots, x_T)$. Step 1.1 computes the collection $\Theta^k$ of cluster centroids at iteration $k$, and Step 1.2 reassigns data points to clusters by updating their labels $s^k_t$. 

10
When again $L(S) = 0$ 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 PWL separation function $\vartheta$ in (9) [8], 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 $\{x \in X : \theta'_{x,s_t}[\bar{1}] \geq \theta'_{x,j}[\bar{1}], \forall j \neq s_t, j \in K\}$ and matching the predicted output $y_t \approx \theta'_{y,s_t}[1]$. Finally, we remark that Algorithm 1 is also applicable to the more general case in which the mode loss $L$ also depends on $\Theta$, by simply replacing Steps 1.1 and 1.2 with

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

(23a)

$$
S^k \leftarrow \arg\min_{S} \ell(X,Y,\Theta^k,S) + L(S,\Theta^k).$$

(23b)

This would cover the case in which $L$ contains parameters to be estimated.

4.2 Inference

4.2.1 One-step ahead prediction

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

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

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

(24)

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

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

(25)

Algorithm 2 attempts at solving problem (24). Step 1 is solved again by the DP iterations (21) over the time span $[0, t]$, with the only difference that in (21a) we set the terminal penalty equal to $M(s,t) = \mathcal{L}^{\text{mode}}(s) + \min_{y} \{\ell(\hat{x}_t, y, \theta_s)\}$, since the last output $y_t$ is determined later at Step 2.

Note that open-loop prediction, that is the task of predicting $\hat{y}_t$ without acquiring $\hat{Y}_{t-1}$, can be simply obtained by replacing $\hat{Y}_{t-1} = (\hat{y}_1, \ldots, \hat{y}_{t-1})$ with $\hat{Y}_{t-1} = (\hat{y}_1, \ldots, \hat{y}_{t-1})$. Arbitrary combinations of one-step ahead and open-loop predictions are possible to handle the more general case of intermittent output data availability.
Algorithm 2 Inference

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

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

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

4.2.2 Recursive inference

When $L_{\text{trans}} = 0$ problem (24) becomes completely separable and simplifies to

$$ (\hat{y}_t, \hat{s}_t) = \arg\min_{y, s} \ell(\tilde{x}_t, y, \theta_s^*) + L_{\text{mode}}(s) \quad \text{s.t. } y \in \mathcal{Y}_t. \quad (26) $$

For example, in the case of $K$-means ($L(s) = 0$), the estimate obtained by (26) is

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

that is $\hat{y}_t$ is the centroid of the cluster which $\tilde{x}_t$ is closest to.

When the mode transition loss function $L_{\text{trans}} \neq 0$, the simplification in (26) does not hold anymore. Nonetheless, an incremental version of (24), producing exactly the same results, can be still derived in the following form

$$ (\hat{y}_t, \hat{s}_t) = \arg\min_{y, s} \ell(\tilde{x}_t, y, \theta_s) + L_t(s) \quad \text{s.t. } y \in \mathcal{Y}_t \quad (27a) $$

where $L_t : \mathcal{K} \to \mathbb{R}$ is the arrival cost recursively defined as

$$ L_t(s_t) = L_{\text{mode}}(s_t) + \min_{s_{t-1}} \left\{ \ell(\tilde{x}_{t-1}, \tilde{y}_{t-1}, \theta_{s_{t-1}}) + L_{t-1}(s_{t-1}) + \right.$$ \phantom{.}

$$ + L_{\text{trans}}(s_t, s_{t-1}) \right\}, \quad t = 1, 2, \ldots \quad (27b) $$

$$ L_0(s_0) = L_{\text{init}}(s_0). \quad (27c) $$

Clearly, while producing exactly the same results, the formulation in (27) is much more efficient than Algorithm 2 as the number of computations does not increase with $t$.

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 $\hat{y}_k$ is only known at steps $k \in T_t \subseteq \{1, \ldots, t\}$. Steps 1-2 are replaced
by
\[
\hat{S}_t \leftarrow \arg \min_{S_t} \left\{ \mathcal{L}(S_t) + \sum_{j \in \mathcal{T}_t} \ell(\hat{x}_j, \hat{y}_j, \theta^*_s) + \sum_{j \in \mathcal{T}_t} \min_{y_j \in Y_j} \ell(\hat{x}_j, y_j, \theta^*_s) \right\}
\]
\[
y_j \leftarrow \arg \min_{y \in Y_j} \ell(\hat{x}_j, y, \theta^*_s), \quad \forall j \in \mathcal{T}_t
\]
where \(\mathcal{T}_t = \{1, \ldots, t\} \setminus \mathcal{T}_t\).

4.2.4 Pure mode estimation

In case we are interested in estimating only the latent mode \(\hat{s}_t\) given \(\hat{x}_1, \ldots, \hat{x}_t, \hat{y}_1, \ldots, \hat{y}_{t-1}\) and also \(\hat{y}_t\), we can keep using (27) by simply changing (27a) to
\[
\hat{s}_t = \arg \min_x \ell(\hat{x}_t, \hat{y}_t, \theta_s) + \mathcal{L}_t(s).
\]
This allows reconstructing the mode sequence \(\hat{s}_1, \ldots, \hat{s}_t\) recursively from the available data set, which may be useful for example to detect changes in the relation between the input \(\hat{x}_t\) and the output \(\hat{y}_t\).

4.3 Relation with Hidden Markov Models

Jump models have several common features with hidden Markov models (HMMs) \[23\]. First, both models consider the presence of discrete latent states \(s_t\). While HMMs assume that the sequence \(S\) of such states satisfy the Markov property
\[
p(s_t | s_{t-1}, \ldots, s_0) = p(s_t | s_{t-1})
\]
in jump models the particular form chosen in (13a) for the mode sequence loss \(\mathcal{L}\) makes estimating \(\hat{s}_t\) incrementally as in (27) possible.

Second, in HMMs the observed outputs are such that
\[
p(y_t | x_1, \ldots, x_{t-1}, y_{t-1}, \ldots, y_1, s_t, \ldots, s_0) = p(y_t | x_t, s_t).
\]
Similarly, in jump models \(\hat{y}_t\) is a unique function of a given pair \((x_t, s_t)\), as (27a) becomes
\[
\hat{y}_t = \arg \min_{y \in Y_t} \ell(x_t, y, \theta_{s_t}).
\]
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 \(\mathcal{Y} = \{1, \ldots, L\}\). An HMM is characterized by the set of discrete probabilities
\[
p(s_{t+1} = i | s_t = j) = \pi_{i,j}, \quad i, j \in \mathcal{K} \quad (29a)
\]
\[
p(s_0) = \pi_{s_0} \quad (29b)
\]
\[
p(y_t = v | s_t = j) = \beta_{j,v}, \quad v \in \mathcal{Y}. \quad (29c)
\]
Let us set \(x_t = 1\), \(\theta_s = s\), and define the loss function \(\ell\) as
\[
\ell(x, y, \theta_s) = -\log(\beta_{\theta_s,y}).
\]
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 (24)–(25) returns the mode sequence \( \tilde{S}_t \) that best matches the observed sequence \( \tilde{Y}_t \) of outputs and that sets the output \( \tilde{y}_t \) equal to the value \( v \in \mathcal{Y} \) that maximizes the probability \( \beta_{s,v} \). An extension to HMMs with continuous observation densities can be obtained by properly redefining the loss function \( \ell \) in (30).

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

\[
\ell(y, \theta_s) = -\log(e_y' \theta_s)
\]

where \( e_y \) is the \( y \)th column of the identity matrix of size \( L \). If the initial probability distribution \( \pi_{s_0} \) 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 \( \Theta \) should also include \( \pi_{s_0} \) and \( \pi_{i,j} \), leading to the general case of having the mode sequence loss \( \mathcal{L} \) also dependent on \( \Theta \) as in (23).

The well-known Expectation-Maximization (EM) algorithm [11] 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 \( \Theta \). Instead, as shown by Proposition 1, our approach maximizes \( \log p(Y, S, \Theta|X) \) with respect to \( \Theta \) 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 [13], 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 \( \ell \) as in (20), \( \mathcal{L}^{\text{trans}} \) 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 [13], 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 $S^*$ found so far, and run Algorithm 1 again, executing the algorithm $N$ times in total.

Assuming $L_{\text{mode}} = 0$, we update $L_{\text{trans}}$, $L_{\text{init}}$ from one run of Algorithm 1 to another by setting

$$
\mu_j \leftarrow \frac{\#\{t \in \{1, \ldots, T\} : s^*_t = j\}}{T} \quad (33a)
$$
$$
\mu_{ij} \leftarrow \frac{\#\{t \in \{1, \ldots, T\} : s^*_t = i, s^*_t-1 = j\}}{T} \quad (33b)
$$
$$
L_{\text{trans}}(i,j) \leftarrow -\tau_i \log \left( \frac{\mu_{ij}}{\mu_j} \right), \quad i, j = 1, \ldots, K \quad (33c)
$$
$$
\mu^0_j \leftarrow \frac{\#\{t \in \{0, \ldots, T\} : s^*_t = j\}}{T} \quad (33d)
$$
$$
L_{\text{init}}(j) \leftarrow -\tau_0 \log (\mu^0_j) \quad (33e)
$$

where $\#$ denotes the cardinality (number of elements) of a set, $S^* = (s^*_0, \ldots, s^*_T)$, and we set

$$
\tau_i \leftarrow \frac{\sum_{j=1}^K L_{\text{trans}}(i,j)}{\sum_{j=1}^K \log \left( \frac{\mu_{ij}}{\mu_j} \right)}, \quad i = 1, \ldots, K
$$
$$
\tau_0 \leftarrow \frac{\sum_{j=1}^K L_{\text{init}}(j)}{\sum_{j=1}^K \log (\mu^0_j)}
$$

to preserve the relative weight of $L$ with respect to the loss $\ell$ and regularization $r$.

Choosing $L$ as in (33) is motivated by the statistical interpretation (14c)–(14d) and used routinely for estimating state probabilities in HMMs [23]. Clearly, (33) are well defined only if $\mu_{ij}, \mu_j, \mu^0_j > 0$ for all $i, j = 1, \ldots, K$. If the latter condition is not satisfied, one may consider adding the following Laplace smoothing

$$
\mu_j \leftarrow \frac{1 + \#\{t \in \{1, \ldots, T\} : s^*_t = j\}}{T + K} \quad (34a)
$$
$$
\mu_{ij} \leftarrow \frac{1 + \#\{t \in \{1, \ldots, T\} : s^*_t = i, s^*_t-1 = j\}}{T + K^2} \quad (34b)
$$
$$
\mu^0_j \leftarrow \frac{1 + \#\{t \in \{0, \ldots, T\} : s^*_t = j\}}{T + K} \quad (34c)
$$

when estimating $\mu_j, \mu_{ij}$ and $\mu^0_j$.

Estimating $L$ as in (33) is especially useful for improving the quality of inference, both when using (25) or (27).

5 Examples

We test the algorithms proposed in the previous sections on various problems of regression and classification using jump models. As the DP computation in
Step 1.2 of Algorithm 1 also provides the optimal cost $V^k = J(X, Y, \Theta^k, S^k)$, when running the tests we replace the termination criterion in Step 2 with

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

where $\epsilon_V$ is a small tolerance.

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/](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 $y_t \in \mathbb{R}, x_t \in \mathbb{R}^{20}, \zeta_t \sim \mathcal{N}(0, \sigma_y^2)$. The coefficients of the parameter vectors $\theta_i$ 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 $s_0 = 1$.

As loss functions we consider

$$\ell(x_t, y_t, \theta_{s_t}) = \|y_t - \theta_{s_t}' x_t\|_2^2$$

$$r(\theta_k) = 10^{-5} \|\theta_k\|_2^2$$

$$L_{\text{trans}}(s_t, s_{t-1}) = \begin{cases} -\tau \log(1 - (K - 1)\pi) & \text{if } s_t = s_{t-1} \\ -\tau \log \pi & \text{if } s_t \neq s_{t-1} \end{cases}$$

$$L_{\text{init}}(s_0) = 0$$

$$L_{\text{mode}}(s_t) = 0$$

where $\tau$ is treated as a hyper-parameter to be tuned. The tolerance $\epsilon_V = 10^{-8}$ is used in (35) as a stopping criterion for Algorithm 1, which is executed $N = 5$ times from different random initial guesses. Each execution is limited to $k_{\text{max}} = 1000$ iterations.

We run Algorithm 1 on the training data for different magnitudes $\sigma_y$ of output noise and values of the hyper-parameter $\tau$. The resulting model coefficients $\Theta^*$ are then used in Algorithm 2 for recursive inference on the production data.

For assessing the quality of inference we use the true loss

$$\hat{L}_{\text{true}} = \frac{1}{\tilde{T}} \sum_{i=1}^{\tilde{T}} \|\hat{y}_i - \hat{y}_i\|_2^2.$$ 

In addition, assuming the latent modes $\tilde{s}_t$ are available only for validation purposes, we consider the following mode-mismatch figure

$$\ell_{s_{\text{true}}} = \frac{100}{T} \sum_{i=1}^{T} \delta_{\tilde{s}_t, \tilde{s}_t}$$

(36)
Figure 1: Jump linear model fit and validation: true loss $\tilde{L}_{\text{true}}$ (left) and mode mismatch $\ell_{\text{true}}$ (right), optimal theoretical value $\tau^* = 2\sigma_y^2$ (dashed line)

where $\delta_{i,j}$ is the Kronecker delta function. The results are summarized in Figure 5.1

By recalling (20) 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 $\ell(x_t, y_t, \theta_{s_t}) = 1$, $\mathcal{L}^{\text{trans}}(s_t, s_{t-1}) = -\tau^* \log \pi_{s_t,s_{t-1}}$ with $\tau^* = 2\sigma_y^2$. Figure 5.1 also reports the value of $\tau^*$ (dashed line) corresponding to different values of $\sigma_y$. As expected, the best value for $\tau$ obtained by cross validation, corresponding to the minimum of the plotted curves, corresponds to the theoretical one $\tau^*$ that would be obtained if $\sigma_y$ were known. For large values of $\tau$ 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, for Algorithm 2 is 0.89 $\mu$s per data point.

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

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

$$\ell(x_t, y_t, s_t) = \max(1 - y_t \theta'_{s_t} x_t, 0)$$

$$r(\theta_k) = 10^{-5} \|\theta_k\|_2^2$$

$$L^{\text{trans}}(s_t, s_{t-1}) = \tau (1 - \delta_{i,j})$$

$$L^{\text{init}}(s_0) = 0$$

$$L^{\text{mode}}(s_t) = 0.$$ 

Figure 5.2 shows the results obtained for different values of the hyper-parameter $\tau$. We consider the mismatch between the true labels $y_t$ and the estimated labels $y^*_t = \text{sign}((\theta^*_s)' x_t)$ returned by Algorithm 1 on the training data, and also between the true labels $\tilde{y}_t$ and the labels $\hat{y}_t = \text{sign}((\theta^*_s)' \tilde{x}_t)$ returned by Algorithm 2 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 $s^*_t, \hat{s}_t$. Good values for $\tau$ are in the range $1 \div 10$, 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 

Algorithm 2 takes an average of 0.57 $\mu$s 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 $x_t, \zeta_t \in \mathbb{R}^8$, $u_t \in \mathbb{R}^2$, $\zeta^j_t \sim \mathcal{N}(0, \sigma_y^2)$ for all $j = 1, \ldots, 8$, the matrix pairs $(A_i, B_i)$ are random stable systems for all $i = 1, \ldots, K$. The modes $s_t$ are randomly generated according to an (unknown) transition probability matrix $\Pi \in \mathbb{R}^{4 \times 4}$. The goal is to estimate the system matrices $(A_i, B_i)$, $i = 1, \ldots, K$, and the transition probability $\Pi$ 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 $L^{\text{trans}}(i, j) = \tau$, and zero
Figure 2: Jump binary classifier: percentage of misclassified labels (left) and mode mismatch (right) on training and production data

losses $\mathcal{L}^{\text{init}}$, $\mathcal{L}^{\text{mode}}$. 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 (27), the transition probability matrix $\hat{\Pi}$ is reconstructed using (33) on the estimated mode sequence $S^*$ returned by Algorithm 1.

The results are reported in Figure 3. 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 $\sigma_y = 0$. The average CPU time for executing Algorithm 1 is 68 ms (corresponding to about 15 iterations), while Algorithm 2 takes of 0.57 $\mu$s 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 [17]. 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 [3, 18, 20].

A data record over an interval of 15 s is gathered from an experimental bench (see [17] for details), with a sampling frequency of 2 kHz. 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 \( T = 24000 \) samples, which consist of the observations gathered in the first 12 s of the experiments, and a test set with \( T' = 6000 \) samples, which consist of the observations gathered in the last 3 s.

We want to fit a PWA model (9)–(10) with \( K = 2 \) discrete modes. Each regression model is given by
\[
y_t = \theta_s^\prime x_t,
\]
where
\[
x_t = [y_t - 1 \ y_t - 2 \ u_t - 1 \ u_t - 2]'.
\]
A piecewise linear separation function \( \vartheta \) defined in (9) is used to describe the partition of \( \mathcal{X} = \mathbb{R}^4 \).

Algorithm 1 is executed \( N = 5 \) times on the first 22000 samples of the training set with loss function \( \ell(x_t, y_t, \theta_{s_t}) \) as in (10) and mode sequence loss \( \mathcal{L} = 0 \). The remaining 2000 samples are used to tune the hyper-parameter \( \rho \), whose best value is \( \rho = 1.23 \cdot 10^{-5} \). The average CPU time for executing Algorithm 1 for a fixed value of \( \rho \) is 200 ms.

Figure 4 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 \( \text{BFR} = 100 \left( 1 - \sqrt{\frac{\sum_{t=1}^{\tilde{T}} \|\tilde{y}_t - \bar{y}\|^2}{\sum_{t=1}^{\tilde{T}} \|\tilde{y}_t - \bar{y}\|^2}} \right) \) % is equal to 81%, where \( \bar{y} \) denotes the average of the outputs \( \tilde{y}_1, \ldots, \tilde{y}_{\tilde{T}} \). It is interesting to note that the reconstructed mode correctly captures the two different operating conditions of the process (mode 1 = free mode, mode 2 = impact mode).

6 Conclusions

We have presented a new framework for fitting a jump model to a temporal sequence of data. 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. Another strength of the 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.

Future research will address an incremental version of the fitting algorithm, so to update models and infer output/mode pairs when data are streaming on-line.
Figure 3: Markov jump linear dynamical system: true loss $\tilde{L}_{true}$ (left) and mode mismatch $\ell^s_{true}$ (right).

References

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

[2] R. Bellman. Dynamic Programming. Princeton University Press, Princeton, NJ, USA, 1957.

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

[4] A. Bemporad and N. Giorgetti. Logic-based methods for optimal control of hybrid systems. IEEE Transaction on Automatic Control, 51(6):963–976, 2006.

[5] K.P. Bennett and O.L. Mangasarian. Multicategory discrimination via linear programming. Optimization Methods and Software, 3:27–39, 1994.

[6] S. Boyd and L. Vandenberghe. Convex Optimization. Cambridge University Press, New York, NY, USA, 2004. [http://www.stanford.edu/~boyd/cvxbook.html](http://www.stanford.edu/~boyd/cvxbook.html)
Figure 4: Pick-and-place machine: simulated and actual output (top) and estimated mode sequence (bottom).
[7] V. Breschi, A. Bemporad, and D. Piga. Identification of hybrid and linear parameter varying models via recursive piecewise affine regression and discrimination. In European Control Conference, pages 2632–2637, Aalborg, Denmark, 2016.

[8] V. Breschi, D. Piga, and A. Bemporad. Piecewise affine regression via recursive multiple least squares and multicategory discrimination. *Automatica*, 73:155–162, November 2016.

[9] A.B. Chan and N. Vasconcelos. Modeling, clustering, and segmenting video with mixtures of dynamic textures. *IEEE transactions on pattern analysis and machine intelligence*, 30(5):909–926, 2008.

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

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

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

[13] M. Fridman. Hidden Markov model regression. Technical report, Institute of Mathematics, University of Minnesota, Minneapolis, MN, 1994.

[14] M. Guidolin. Markov switching models in empirical finance. In *Missing Data Methods: Time-Series Methods and Applications*, pages 1–86. Emerald Group Publishing Limited, 2011.

[15] Gurobi Optimization, Inc. *Gurobi Optimizer Reference Manual*, 2017.

[16] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer, New York, 2nd edition, 2009.

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

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

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

[20] H. Ohlsson and L. Ljung. Identification of switched linear regression models using sum-of-norms regularization. *Automatica*, 49(4):1045–1050, 2013.
[21] N. Ozay, Mario M. Sznaier, and C. Lagoa. Model (in)validation of switched arx systems with unknown switches and its application to activity monitoring. In 49th IEEE Conference on Decision and Control, pages 7624–7630, Atlanta, GA, 2010.

[22] V. Pavlovic, J.M. Rehg, and J. MacCormick. Learning switching linear models of human motion. In Advances in neural information processing systems, pages 981–987, 2001.

[23] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2):257–286, 1989.

[24] B. Schuller, M. Wöllmer, T. Moosmayr, G. Ruske, and G. Rigoll. Switching linear dynamic models for noise robust in-car speech recognition. Pattern Recognition, pages 244–253, 2008.

[25] A. Timmermann. Markov Switching Models in Finance, volume 4. John Wiley & Sons, Ltd, 2015.