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/

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), $t = 1, \dots, T$, 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 $x_1, \dots, x_t$, and possibly past outputs $y_1, \dots, y_{t-1}$. 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 $x_1, \dots, x_t, y_1, \dots, y_{t-1}, y_t$ and other variables and parameters. The chosen shape for J determines the structure of the corresponding regression model.

Given a production data sequence $(\tilde{x}_1, \tilde{y}_1), \dots,$ thought to be generated by a similar process that produced the training data, the quality of the regression model over a time period $t = 1, \dots, \tilde{T}$ 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 $X = (x_1, \dots, x_T)$, $Y = (y_1, \dots, y_T)$. 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 $\theta_s \in \mathbb{R}^d$, $s = 1, \dots, K$, 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 $\Theta = (\theta_1, \dots, \theta_K)$ and $S = (s_1, \dots, s_T)$. The optimal parameters $\theta_1^\star, \dots, \theta_K^\star$ 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 $\theta_1^\star, \dots, \theta_K^\star$ 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 $\theta_1^\star, \dots, \theta_K^\star$ as the centroids of the