Bayesian Inference for Switching Linear Dynamical Systems

D. Bacilieri, L. Barbiero, G. Bordin, A. Pitteri

9th March 2024

1 Model description

A switching linear dynamical system – also known as *switching state space model* – is defined, borrowing the notation from Linderman et al. [3], by the set of discrete-time stochastic equations

$$x_t = A_{z_t} x_{t-1} + b_{z_t} + v_t, \tag{1}$$
$$y_t = C_{z_t} x_t + d_{z_t} + w_t, \tag{2}$$

where $v_t \in \mathbb{R}^M$ and $w_t \in \mathbb{R}^N$ are Gaussian-distributed random vectors with mean zero and variance Q_{z_t}, S_{z_t} respectively. The vectors $y_t \in \mathbb{R}^N$, $t = 1, \ldots, T$ may represent a time series of observations, while the $x_t \in \mathbb{R}^M$ are a set of continuous latent states linked together by linear dynamics defined by the matrices $A_k \in \mathbb{R}^{M \times M}$ and bias vectors $b_k \in \mathbb{R}^M$. The transformation between x and y is also linear, through the matrices $C_k \in \mathbb{R}^{N \times M}$ and bias vectors $d_k \in \mathbb{R}^N$.

The linear parameters A_k, b_k, C_k, d_k form a discrete set of K elements, and a discrete latent variable $z_t \in \{1, \ldots, K\}$ sets the specific instances in use at time step t. The variable z evolves over time as a Markov process, meaning that z_t is conditionally independent of all previous states except for its immediate predecessor z_{t-1}:

$$p(z_t \mid z_{t-1}, z_{t-2}, \ldots, z_1) = p(z_t \mid z_{t-1}). \tag{3}$$

We will denote the probability to transition from $z_{t-1} = j$ to $z_t = k$ with π_{jk}. The transition $z_{t-1} \to z_t$ effectively modifies the linear dynamics from x_{t-1} to x_t and the linear transformation from x_t to y_t, *switching* from one regime to another.

Given a set of data points y_t, the goal is then to infer the posterior distribution of the parameter set

$$\vartheta = \{\pi_k, A_k, b_k, Q_k, C_k, d_k, S_k\}, \tag{4}$$

where $x_{1:T}$ denotes the whole sequence $x_1, x_2, \ldots, x_T$, and π_k the kth row of the transition matrix.

2 Model implementation

To set up a Monte Carlo sampling scheme, we chose to work with the Stan [5] programming language and its implementation in R through the package rstan [4]. Sampling in Stan is done by default using a variant of the Hamiltonian Monte Carlo scheme called 'No-U-Turn sampler' or NUTS [1].

The model cannot be implemented directly as it is, in the sense of specifying a categorical likelihood for the transition $z_t \to z_{t+1}$ and Gaussian likelihoods for $x_t \mid x_{t-1}$ and $y_t \mid x_t$, because Stan does not allow the definition of integer parameters: so, one should marginalize over the hidden discrete states. Besides Stan's limitations in this regard, the resulting strategy – known in the literature as *forward algorithm* – is more efficient than the straightforward implementation in sampling low-probability states, and is commonly used in similar inference problems involving hidden Markov models or other state space models [2].

2.1 The forward algorithm

The basic idea behind the forward algorithm is to exploit a recursive relationship to build the full likelihood: indeed, consider the quantity

$$\gamma_t(k) \coloneqq p(z_t = k, x_{1:t}, y_{1:t}). \tag{5}$$

By summing over the z states at $t-1$ first and then using the chain rule repeatedly, we can write

$$\begin{aligned} \gamma_t(k) &= \sum_{j=1}^{K} p(z_t = k, z_{t-1} = j, x_{1:t}, y_{1:t}) \\ &= p(y_t \mid z_t = k, x_t)\, p(x_t \mid z_t = k, x_{t-1}) \\ &\quad \cdot \sum_{j=1}^{K} \pi_{jk}\, p(z_{t-1} = j, x_{1:t-1}, y_{1:t-1}), \end{aligned} \tag{6}$$

where we have recognized the conditional probability $p(z_t = k \mid z_{t-1} = j, x_{1:t-1}, y_{1:t-1}) = p(z_t = k \mid z_{t-1} = j)$ as the element (j, k) of the transition matrix π. The first two terms outside the sum are the likelihoods of y_t and x_t, and because of the model definition they only depend on z_t, x_t and x_{t-1}. Also, they are simply Gaussian densities:

$$\mathcal{L}_k(y_t) \coloneqq p(y_t \mid z_t = k, x_t) = \mathcal{N}(C_k x_t + d_k, S_k), \tag{7}$$
$$\mathcal{L}_k(x_t) \coloneqq p(x_t \mid z_t = k, x_{t-1}) = \mathcal{N}(A_k x_{t-1} + b_k, Q_k). \tag{8}$$

Then, the remaining terms in the sum in (6) are nothing else than $\gamma_{t-1}(j)$, giving us the recursive relation we needed:

$$\gamma_t(k) = \mathcal{L}_k(y_t)\, \mathcal{L}_k(x_t) \sum_{j=1}^{K} \pi_{jk} \gamma_{t-1}(j). \tag{9}$$

Indeed, to retrieve the full joint likelihood of the sequences $x_{1:T}$ and $y_{1:T}$ we only need to marginalize γ at the last time step T over the discrete states $k = 1, 2, \ldots, K$:

$$p(x_{1:T}, y_{1:T}) = \sum_{k=1}^{K} p(z_T = k, x_{1:T}, y_{1:T}) = \sum_{k=1}^{K} \gamma_T(k). \tag{10}$$

To recursively build γ_t up to time T we need $\mathcal{O}(TK^2)$ operations, because of the double marginalization over z_t and z_{t-1}. To initialize the recursion,

$$\begin{aligned}\gamma_1(k) &= p(z_1 = k, x_1, y_1) \\ &= \mathcal{L}_1(y_1)\, p(x_1 \mid z_1 = k)\, p(z_1 = k).\end{aligned} \tag{11}$$

The last two terms are the prior distributions on x_1 and z_1. We chose a multivariate Gaussian for the first and a uniform distribution over the K states for the second.

At this point, we also need the prior distributions for the dynamical parameters. Following the suggestion from Linderman et al. [3], we chose matrix-normal-inverse-Wishart priors:

$$(A_k, b_k), Q_k \sim \mathrm{MNIW}(M_x, \Omega_x, \Psi_x, \nu_x) \tag{12}$$

$$(C_k, d_k), S_k \sim \mathrm{MNIW}(M_y, \Omega_y, \Psi_y, \nu_y). \tag{13}$$

Here $M_x \in \mathbb{R}^{M\times(M+1)}$ and $M_y \in \mathbb{R}^{N\times(M+1)}$ are the mean matrices of the matrix normals, $\Omega_x, \Omega_y \in \mathbb{R}^{(M+1)\times(M+1)}$ their between-column covariance matrices, while $\Psi_x \in \mathbb{R}^{M\times M}$ and $\Psi_y \in \mathbb{R}^{N\times N}$ are the scale matrices of the inverse Wisharts and ν_x, ν_y their degrees of freedom. The returned random matrices with $M+1$ columns are then split between the matrices A_k and C_k and their corresponding bias vectors b_k and d_k.

2.2 Reconstructing the hidden states

Since we marginalize out the z sequence during the sampling procedure, we need a way to recover them probabilistically. One way to do it is to search *a posteriori* for the most likely hidden sequence of z states conditioned to the observed $y_{1:T}$ and inferred $x_{1:T}$. This is done through the so-called 'Viterbi algorithm' [5], which is based on a recursive relation much like the forward algorithm.

Indeed, consider the quantity

$$\eta_t(k) \coloneqq \underset{z_{1:t-1}}{\arg\max}\, p(z_{1:t-1}, z_t = k, x_{1:t}, y_{1:t}). \tag{14}$$

If we proceed similarly to (6), and using also the fact that

$$\max_{a,b}[f(a)g(a,b)] = \max_a[f(a)\max_b g(a,b)], \tag{15}$$

we can expand the definition as

$$\begin{aligned}\eta_t(k) = &\underset{j\in\{1,\dots,K\}}{\arg\max}\ [\, p(y_t \mid z_t = k, x_t)\, p(x_t \mid z_t = k, x_{t-1}) \\ &\cdot \pi_{jk}\, \underset{z_{1:t-2}}{\arg\max}\, p(z_{1:t-2}, z_{t-1} = j, x_{1:t-1}, y_{1:t-1})].\end{aligned} \tag{16}$$

We then recognize the last factor as η at the previous time step $t-1$, giving us the recursive relation

$$\eta_t(k) = \mathcal{L}_k(y_t)\mathcal{L}_k(x_t)\ \underset{j\in\{1,\dots,K\}}{\arg\max}\ [\pi_{jk}\,\eta_{t-1}(j)]. \tag{17}$$

Regarding the initial value, there is no z_{t-1} to maximize over, so we get

$$\eta_1(k) = p(z_1 = k, x_1, y_1), \tag{18}$$

which is the same initialization of $\gamma_1(k)$ as shown in the previous section, Eq. (11). Once we have the value of η at time T we can maximize over z_T and recover the maximum-probability sequence $\hat{z}_{1:T}$:

$$\hat{z}_{1:T} = \underset{z_{1:T}}{\arg\max}\, p(z_{1:T}, x_{1:T}, y_{1:T}) = \underset{k\in\{1,\dots,K\}}{\arg\max}\ \eta_T(k). \tag{19}$$

The procedure is thus substantially the same as the forward algorithm, but with maximization replacing summation.

References

[1] Bob Carpenter et al. 'Stan: A Probabilistic Programming Language'. In: *Journal of Statistical Software* 76.1 (2017), pp. 1–32. DOI: 10.18637/jss.v076.i01.

[2] Luis Damiano, Brian Peterson and Michael Weylandt. 'A tutorial on hidden Markov models using Stan'. In: StanCon 2018 (Asilomar Conference Center, California, 10th Jan. 2018). Asilomar. Zenodo, 10th Jan. 2018. DOI: 10.5281/zenodo.1284341.

[3] Scott W. Linderman et al. 'Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems'. In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by Aarti Singh and Jerry Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, Apr. 2017, pp. 914–922.

[4] Stan Development Team. *RStan: the R interface to Stan*. R package version 2.32.6. 2024.

[5] Stan Development Team. *Stan Modeling Language Users Guide and Reference Manual*. Version 2.34. 2024.