Parameter Estimation in Sparse Linear-Gaussian State-Space Models via Reversible Jump Markov Chain Monte Carlo

Benjamin Cox and Víctor Elvira

School of Mathematics, University of Edinburgh

Problem Statement

We have:

- A linear Gaussian state-space model with unknown transition matrix $\mathbf{A}$

$$\begin{aligned} \mathbf{x}_t &= \mathbf{A}\mathbf{x}_{t-1} + \mathbf{q}_t, \\ \mathbf{y}_t &= \mathbf{H}\mathbf{x}_t + \mathbf{r}_t, \end{aligned} \tag{1}$$

where
- $\mathbf{A} \in \mathbb{R}^{d_x \times d_x}$ is the unknown transition matrix
- $\mathbf{H} \in \mathbb{R}^{d_y \times d_x}$ is a known matrix
- $\mathbf{q}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ with $\mathbf{Q}$ known
- $\mathbf{r}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ with $\mathbf{R}$ known
- $\mathbf{x}_0 \sim \mathcal{N}(\bar{\mathbf{x}}_0, \mathbf{P}_0)$ with $\bar{\mathbf{x}}_0$ and $\mathbf{P}_0$ known
- Observations $\mathbf{y}_{1:T} \in \mathbb{R}^{d_y}$ of this SSM

Objective: estimate the value and structure of $\mathbf{A}$.

Kalman Filtering

Kalman filtering allows us to obtain optimal estimates of $\mathbf{x}_{1:T}$ given $\mathbf{y}_{1:T}$ and values for the other parameters. In particular it does this efficiently and gives an analytic distribution for the hidden state.

We can use this distribution to obtain a likelihood over which to optimise our parameter values. This is standard and is the focus of many existing techniques.

Reversible Jump Markov Chain Monte Carlo

Reversible Jump Markov Chain Monte Carlo (RJMCMC) is a method by which one can sample from multiple different models within the same MCMC chain. With some known probability at each step the chain transitions between models in a manner that preserves detailed balance and ergodicity.

We can use different models to explore sparse subspaces of $\mathbf{A}$, thus allowing us to sample structure as well as value.

Proposed Algorithm: SpaRJ

We can combine RJMCMC and Kalman filtering to sample different models of our $\mathbf{A}$ matrix. Using RJMCMC lets us operate in a Bayesian framework, affording additional flexibility.

- We define the sparsity of $\mathbf{A}$ as a per-model property, therefore changing sparsity when we change models. This allows the sampling of exact zeros.
- We include prior information via a function $\Lambda(\cdot, \cdot, \lambda)$. This function compares the accepted sample and the proposed sample.
- We walk over sparsity levels as a modified random walk: we have maximal and minimal sparsity levels, and the ability to remain at the same sparsity/model.
  - We remain at the same model/level of sparsity with probability π_0
  - If we do not remain we jump one level sparser with probability π_{-1}, otherwise we jump one level denser. (See Alg. 2 for details as to how jumps are performed)

Algorithm 1 gives the core method. Algorithm 2 gives the full form of the jumping rules. Note some corrections to the acceptance probability are made when jumping.

The method provides flexibility in the choice of

- $\Lambda(\cdot, \cdot, \lambda)$, the prior penalty function
- $g(\cdot)$, the distribution from which newly dense parameters are drawn
- The method by which samples of $\mathbf{A}$ are generated from the posterior of a retained model

We find that the Lasso, $\mathcal{N}(0, 0.1^2)$, and RWMH work well respectively.

Performance

The method has good performance compared to existing methods due to several properties:

- The algorithm does not sample sparse elements, reducing the dimension of the parameter space
- Penalising complexity (to a specified extent)
- Exploiting the interconnectedness of the generated models when jumping
- Sparse samples allow for easier inference
  - Significance testing is much easier
  - Checking for interrelatedness between series is easier
  - Real systems tend to be sparse
- Sampler for the posterior of $\mathbf{A}$ within each model can be modified using subject knowledge

Furthermore the method is effectively composed of several smaller algorithms that can be changed as long as the output is comparable. Therefore it is very tunable to a given use case.

Overview

We have developed a method, which we call SpaRJ, that combines RJMCMC and Kalman filtering to sparsely sample $\mathbf{A}$. Our method can be used as

- A sampler to obtain sparse samples of $\mathbf{A}$ for inference
- A method to assess the linear relatedness of different time series
- An algorithm to sparsely estimate parameters in an AR(IMA) model

or anywhere else an estimate of $\mathbf{A}$ is required. The method has solid theoretical guarantees and shows excellent performance in challenging numerical examples.

Algorithm Pseudocode

Parameters

- $\mathbf{A}_0$: initial value for sampling $\mathbf{A}$
- N: number of iterations/samples
- $g(\cdot)$: completion distribution
- π_0, π_{-1}: jump probabilities
- $\Lambda(\cdot, \cdot, \lambda)$: prior function
- All SSM parameters except $\mathbf{A}$

The primary algorithm is detailed here.

Algorithm 1: SpaRJ algorithm

Input: $\mathbf{y}_{1:T}, \mathbf{A}_0, g(\cdot), \pi_0, \pi_{-1}, N, \Lambda(\cdot, \cdot, \lambda)$, all other SSM parameters
Output: Set of N samples $\{\mathbf{A}_n\}_{n=1}^N$
begin
  Initialisation
  Initialise M_0 as fully dense
  Run a Kalman filter, obtaining $l = \log(p(\mathbf{y}_{1:T}|\mathbf{A}_0))$.
  for $n = 1, ..., N$ **do**
    Step 1: Propose model and $\mathbf{A}^*$
    Jump models with probability $1 - \pi_0$.
    if *Jump* **then**
      Run Algorithm 2 to generate M^*, $\mathbf{A}^*$, and c.
    else
      Set $c = 0$, $M^* = M_{n-1}$.
      Sample $\mathbf{A}^*$ from the posterior of M^*.
    end
    Step 2: Modified MH accept-reject
    Run a Kalman filter with $\mathbf{A}^*$ and obtain $l^* = \log(p(\mathbf{y}_{1:T}|\mathbf{A}^*))$.
    Construct $a_r = l^* - l + \Lambda(\mathbf{A}_{n-1}, \mathbf{A}^*, \lambda) + c$.
    Accept with probability $\exp(a_r)$.
    if *accept* **then**
      Set $M_n = M^*$, $\mathbf{A}_n = \mathbf{A}^*$, $l = \log(p(\mathbf{y}_{1:T}|\mathbf{A}_n))$
    end
  end
end

For simplicity we provide the jump algorithm separately.

Algorithm 2: Jumping algorithm

Input: SSM model matrix $\mathbf{A}_{n-1}$, $g(\cdot)$, π_{-1}
Output: Adjusted SSM model matrix $\mathbf{A}^*$, correction c
begin
  Step 1: Determine jump direction
  if $\mathbf{A}_{n-1}$ *Densest OR Sparsest* **then**
    Jump sparser or denser respectively
  else
    Jump sparser with probability π_{-1}
  end
  Step 2: Perform jump
  if *Jump sparser* **then**
    Step 2.1: Jump sparser
    Uniformly select a dense element s of $\mathbf{A}_{n-1}$ to become sparse.
    Set $\mathbf{A}^*$ to $\mathbf{A}_{n-1}$ with s set to 0.
    Set $c = \text{logpdf}(g, s)$.
  else
    Step 2.2: Jump denser
    Uniformly select a sparse element d of $\mathbf{A}_{n-1}$ to become dense.
    Draw $u \sim g(\cdot)$.
    Set $\mathbf{A}^*$ to $\mathbf{A}_{n-1}$ with d set to u.
    Set $c = -\text{logpdf}(g, u)$.
  end
end

Numerical Examples

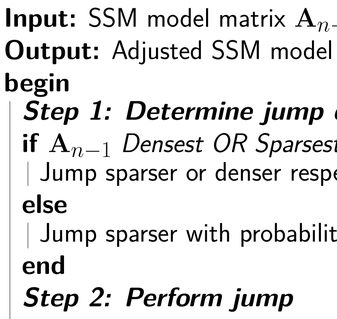

Figure: Comparing GraphEM and SpaRJ over variable series length on the 3x3 system

These systems were derived from random matrices with the given structure and isotropic covariance terms. Results are averaged over 200 independent runs.

transition matrix structure	method	spec.	recall	prec.	F1
3×3 matrix	GraphEM	0.86	0.98	0.79	0.88
	SpaRJ	**0.96**	**0.99**	**0.95**	**0.97**
6×6 block diagonal	GraphEM	0.83	0.90	0.91	0.91
	SpaRJ	**0.92**	**0.96**	**0.95**	**0.95**
12×12 block diagonal	GraphEM	**0.85**	0.77	**0.96**	0.85
	SpaRJ	0.83	**0.89**	0.91	**0.90**

Table: Sparsity statistics over variable systems.

Contact Information

- Web: https://blogs.ed.ac.uk/benjamincox/
- Email: Benjamin.Cox@ed.ac.uk

THE UNIVERSITY of EDINBURGH
School of Mathematics

Centre for Satellite Data in Environmental Science