Backward Filtering Forward Guiding

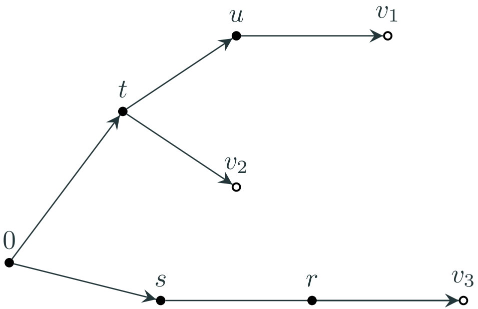
Frank van der Meulen (Vrije Universiteit Amsterdam)
Moritz Schauer (Chalmers University of Technology & University of Gothenburg)

Algorithms and Computationally Intensive Inference Seminar (ACIIS)
Warwick, December 2 (2022)

General problem setting

Problem setting

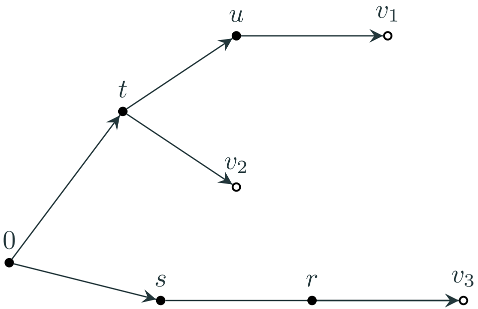
Consider a directed *Markovian* tree:



$\bullet$ latent vertices, $\circ$ leaf/observation-vertices.

Problem setting

Consider a directed *Markovian* tree:



• latent vertices, ∘ leaf/observation-vertices.

To each edge corresponds a Markov kernel $\kappa_{\to t}(x_{\mathrm{pa}(t)}, \mathrm{d}x_t)$.

Problem setting

Think of either a discrete- or continuous-time Markov process evolving over edges.

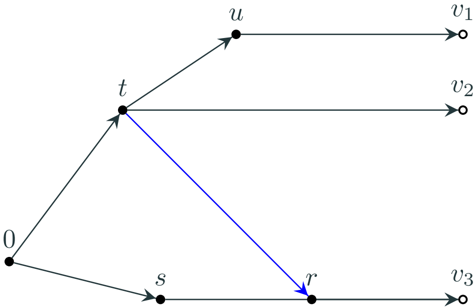
Problem setting

Think of either a discrete- or continuous-time Markov process evolving over edges.

We aim for

1. sampling values at $\bullet$, conditional on values at $\circ$;
2. estimating parameters in kernels;
3. not just on a tree, but on a general Directed Acyclic Graph (DAG).

Example of a DAG



Example 1: interacting particle process

Setup:

- population of n individuals;
- each individual is either **S**usceptible, **I**nfected or **R**ecovered;
- each individual has a known (possibly time varying) set of neighbours .

Example 1: interacting particle process

Setup:

- population of n individuals;
- each individual is either **S**usceptible, **I**nfected or **R**ecovered;
- each individual has a known (possibly time varying) set of neighbours .

Dynamics:

- If $x_i = \mathbf{S}$, then it transitions to $\mathbf{I}$ with intensity $\lambda N_i(t, x)$, with $N_i(t, x)$ number of infected neighbours of individual i at time t.

Example 1: interacting particle process

Setup:

- population of n individuals;
- each individual is either **S**usceptible, **I**nfected or **R**ecovered;
- each individual has a known (possibly time varying) set of neighbours .

Dynamics:

- If $x_i = \mathbf{S}$, then it transitions to $\mathbf{I}$ with intensity $\lambda N_i(t, x)$, with $N_i(t, x)$ number of infected neighbours of individual i at time t.
- If $x_i = \mathbf{I}$, then it transitions to $\mathbf{R}$ with intensity μ.

Example 1: interacting particle process

Setup:

- population of n individuals;
- each individual is either **S**usceptible, **I**nfected or **R**ecovered;
- each individual has a known (possibly time varying) set of neighbours .

Dynamics:

- If $x_i = \mathbf{S}$, then it transitions to $\mathbf{I}$ with intensity $\lambda N_i(t, x)$, with $N_i(t, x)$ number of infected neighbours of individual i at time t.
- If $x_i = \mathbf{I}$, then it transitions to $\mathbf{R}$ with intensity μ.
- If $x_i = \mathbf{R}$, it transitions to $\mathbf{S}$ with intensity ν.

Example 1: Dynamics of each particle

- Time-discretised version of the problem, where time steps are multiples of $\tau > 0$.

Example 1: Dynamics of each particle

- Time-discretised version of the problem, where time steps are multiples of $\tau > 0$.
- In each time interval of length τ, conditional on the the present state of *all* individuals, each individual independently transitions.

Example 1: Dynamics of each particle

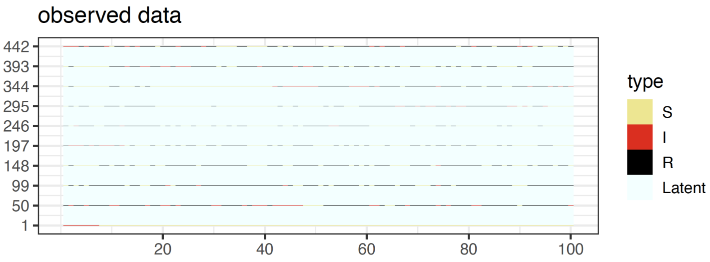
- Time-discretised version of the problem, where time steps are multiples of $\tau > 0$.
- In each time interval of length τ, conditional on the the present state of *all* individuals, each individual independently transitions.

The transition matrix for individual i at time t, given "full state" x:

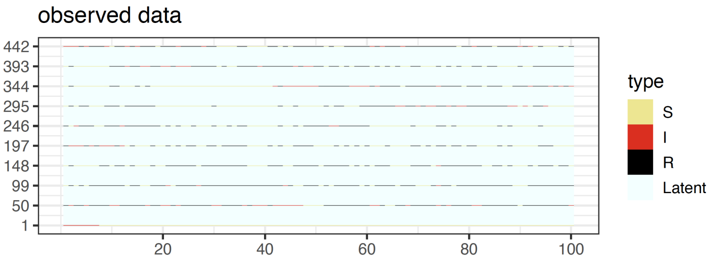
$$\kappa_i(t,x) = \begin{bmatrix} \psi\left(\lambda N_i(t,x)\right) & 1 - \psi\left(\lambda N_i(t,x)\right) & 0 \\ 0 & \psi(\mu) & 1-\psi(\mu) \\ 1-\psi(\nu) & 0 & \psi(\nu) \end{bmatrix},$$

where $\psi(u) = \exp(-\tau u)$

Example 1: data and challenges



Example 1: data and challenges



Observe state of each individual at times $t_0 < t_1 < \cdots t_n$.

Goals:

- identify most probable latent states (partial observations...);
- estimate rate parameters λ, μ and ν.

Example 1: data and challenges

⚠ Dimension of state-space is 3^n.

Example 1: data and challenges

⚠ Dimension of state-space is 3^n.

Variation: observe

$$V_{t_i} \sim \mathsf{Bin}(I(X_{t_i}), \rho),$$

with $I(X_{t_i})$ the number of infected individuals at time t_i,.

Example 1: data and challenges

⚠ Dimension of state-space is 3^n.

Variation: observe

$$V_{t_i} \sim \mathsf{Bin}(I(X_{t_i}), \rho),$$

with $I(X_{t_i})$ the number of infected individuals at time t_i,.

Ju, Heng, Jacob – Sequential Monte Carlo algorithms for agent-based models of disease transmission

Example 2: Wright-Fisher diffusion on a tree

Diffusion approximation to Wright-Fisher model (with mutation) for N diploid individuals.

Consider a directed tree where along each branch

$$\mathrm{d}X_t = (\beta_1(1 - X_t) + \beta_2 X_t)\,\mathrm{d}t + \sqrt{X_t(1 - X_t)}\,\mathrm{d}W_t.$$

At a leaf vertex observe $V \sim \mathit{Bin}(2n, x)$, where x is the state at the parent vertex.

Example 2: Wright-Fisher diffusion on a tree

Diffusion approximation to Wright-Fisher model (with mutation) for N diploid individuals.

Consider a directed tree where along each branch

$$\mathrm{d}X_t = (\beta_1(1 - X_t) + \beta_2 X_t)\,\mathrm{d}t + \sqrt{X_t(1 - X_t)}\,\mathrm{d}W_t.$$

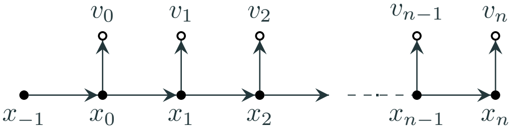
At a leaf vertex observe $V \sim \mathit{Bin}(2n, x)$, where x is the state at the parent vertex.

Wish to

- reconstruct X_t along edges;
- estimate parameters (β_1, β_2).

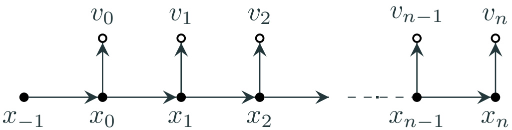
Stoltz et al. – Bayesian inference of species trees using diffusion models

State-space models / hidden Markov models



Well-known filtering, smoothing algorithms dating back to 1960-1970.

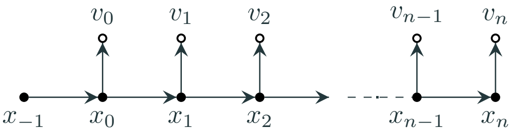
State-space models / hidden Markov models



Well-known filtering, smoothing algorithms dating back to 1960-1970.

- Finite state space: Baum-Welch, Viterbi, forward-backward algorithm.
- Linear Gaussian models: Kalman filter, Rauch-Tung-Striebel smoother.
- Linear stochastic differential equations: Kalman-Bucy filter & smoother.

State-space models / hidden Markov models



Well-known filtering, smoothing algorithms dating back to 1960-1970.

- Finite state space: Baum-Welch, Viterbi, forward-backward algorithm.
- Linear Gaussian models: Kalman filter, Rauch-Tung-Striebel smoother.
- Linear stochastic differential equations: Kalman-Bucy filter & smoother.

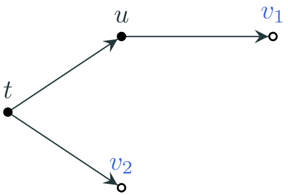
More recently much work on SMC
(twisted particle samplers, controlled SMC).

Conditioning, Doob's h-transform and the Backward Information Filter

Conditioning on a tree

Define

- $\mathcal{V}_t$: all leaf descendants of vertex t.
- $\mathcal{V}_t = \{v_1, v_2\}$.



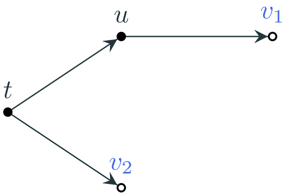
Conditioning on a tree

Define

- $\mathcal{V}_t$: all leaf descendants of vertex t.
- $\mathcal{V}_t = \{v_1, v_2\}$.

Key identity (Bayesian notation):

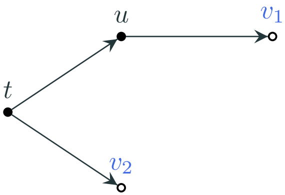
$$p(x_t \mid x_{\mathrm{pa}(t)}, x_{\mathcal{V}_t})$$



Conditioning on a tree

Define

- $\mathcal{V}_t$: all leaf descendants of vertex t.
- $\mathcal{V}_t = \{v_1, v_2\}$.



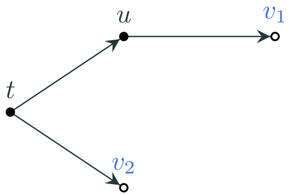
Key identity (Bayesian notation):

$$
\begin{aligned}
p(x_t \mid x_{\mathrm{pa}(t)}, x_{\mathcal{V}_t}) \quad &\propto \quad p(x_t, x_{\mathcal{V}_t} \mid x_{\mathrm{pa}(t)}) \\
&=
\end{aligned}
$$

Conditioning on a tree

Define

- $\mathcal{V}_t$: all leaf descendants of vertex t.
- $\mathcal{V}_t = \{v_1, v_2\}$.



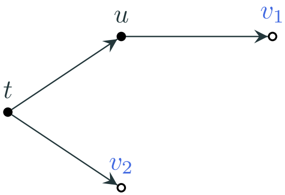
Key identity (Bayesian notation):

$$\begin{aligned} p(x_t \mid x_{\mathrm{pa}(t)}, x_{\mathcal{V}_t}) &\propto p(x_t, x_{\mathcal{V}_t} \mid x_{\mathrm{pa}(t)}) \\ &= p(x_t \mid x_{\mathrm{pa}(t)})\, \underbrace{p(x_{\mathcal{V}_t} \mid x_t, \cancel{x_{\mathrm{pa}(t)}})}_{h_t(x_t)} \end{aligned}$$

Conditioning on a tree

Define

- $\mathcal{V}_t$: all leaf descendants of vertex t.
- $\mathcal{V}_t = \{v_1, v_2\}$.



Key identity (Bayesian notation):

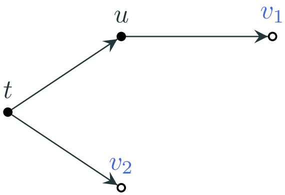
$$\begin{aligned} p(x_t \mid x_{\mathrm{pa}(t)}, x_{\mathcal{V}_t}) &\propto p(x_t, x_{\mathcal{V}_t} \mid x_{\mathrm{pa}(t)}) \\ &= p(x_t \mid x_{\mathrm{pa}(t)}) \underbrace{p(x_{\mathcal{V}_t} \mid x_t, \cancel{x_{\mathrm{pa}(t)}})}_{h_t(x_t)} \end{aligned}$$

Rewrite to $\boxed{\kappa^{\star}_{\to t}(x; \mathrm{d}y) \propto \kappa_{\to t}(x; \mathrm{d}y) h_t(y)}$.

Conditioning on a tree

Define

- $\mathcal{V}_t$: all leaf descendants of vertex t.
- $\mathcal{V}_t = \{v_1, v_2\}$.



Key identity (Bayesian notation):

$$
\begin{aligned}
p(x_t \mid x_{\mathrm{pa}(t)}, x_{\mathcal{V}_t}) &\propto p(x_t, x_{\mathcal{V}_t} \mid x_{\mathrm{pa}(t)}) \\
&= p(x_t \mid x_{\mathrm{pa}(t)}) \underbrace{p(x_{\mathcal{V}_t} \mid x_t, \cancel{x_{\mathrm{pa}(t)}})}_{h_t(x_t)}
\end{aligned}
$$

Rewrite to $\boxed{\kappa^{\star}_{\to t}(x; \mathrm{d}y) \propto \kappa_{\to t}(x; \mathrm{d}y) h_t(y)}$.

⚠ Within the subtree, $h_t(x_t)$ is the likelihood of x_t.

Doob's h-transform

- *Doob's h-transform*: Transformation of each κ_s with h_s to $\kappa_s^\star$:

$$\kappa_{\to s}^\star(x,\, \mathrm{d}y) = \frac{\kappa_{\to s}(x,\, \mathrm{d}y) h_s(y)}{\int \kappa_{\to s}(x,\, \mathrm{d}y) h_s(y)}, \quad s \in \mathcal{S}.$$

A forward pass: Needs $\kappa_{\to s}$ and h_s.

Doob's h-transform

- *Doob's h-transform*: Transformation of each κ_s with h_s to $\kappa_s^\star$:

$$\kappa^\star_{\to s}(x,\,\mathrm{d}y) = \frac{\kappa_{\to s}(x,\,\mathrm{d}y)h_s(y)}{\int \kappa_{\to s}(x,\,\mathrm{d}y)h_s(y)}, \quad s \in \mathcal{S}.$$

  A forward pass: Needs $\kappa_{\to s}$ and h_s.

- Recursive computation of h_s in a backward pass: (Backward Information Filter):
  - Compute h_s from the leaves back to the roots.
  - Acyclic belief propagation, sum-product algorithm, Felsenstein algorithm...

Doob's h-transform

- *Doob's h-transform*: Transformation of each κ_s with h_s to $\kappa_s^\star$:

$$\kappa^\star_{\to s}(x,\, \mathrm{d}y) = \frac{\kappa_{\to s}(x,\, \mathrm{d}y) h_s(y)}{\int \kappa_{\to s}(x,\, \mathrm{d}y) h_s(y)}, \quad s \in \mathcal{S}.$$

  A forward pass: Needs $\kappa_{\to s}$ and h_s.

- Recursive computation of h_s in a backward pass: (Backward Information Filter):
  - Compute h_s from the leaves back to the roots.
  - Acyclic belief propagation, sum-product algorithm, Felsenstein algorithm...
  - ⚠ Only in very specific models tractable.

Doob's h-transform

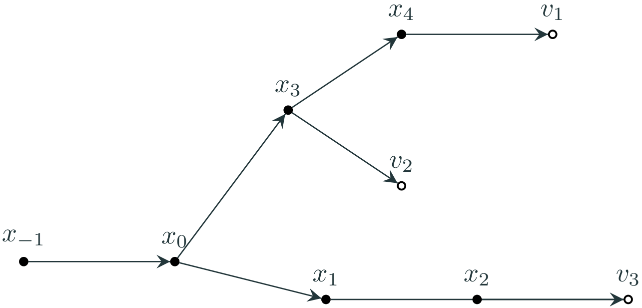
- *Doob's h-transform*: Transformation of each κ_s with h_s to $\kappa_s^\star$:

$$\kappa^\star_{\to s}(x,\,\mathrm{d}y) = \frac{\kappa_{\to s}(x,\,\mathrm{d}y)h_s(y)}{\int \kappa_{\to s}(x,\,\mathrm{d}y)h_s(y)}, \quad s \in \mathcal{S}.$$

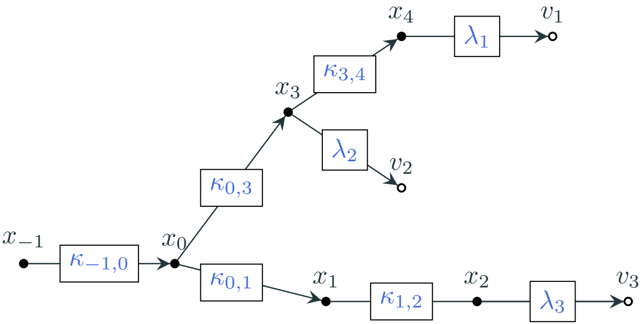
  A forward pass: Needs $\kappa_{\to s}$ and h_s.

- Recursive computation of h_s in a backward pass: (Backward Information Filter):
  - Compute h_s from the leaves back to the roots.
  - Acyclic belief propagation, sum-product algorithm, Felsenstein algorithm...
  - ⚠ Only in very specific models tractable.
- ⚠ On a DAG conditioning changes the dependency structure. There are no conditional kernels $\kappa^\star_{\to s}$ from $\mathrm{pa}(s)$ to s.

Backward Information Filter



Make kernels explicit



Example: finite state space

- Suppose $x_t \in \{1, 2, 3\}$ and $v_t \in \{1{,}2, 3\}$.

  Idea: in observations we cannot distinguish 1 and 2.

Example: finite state space

- Suppose $x_t \in \{①, ②, ③\}$ and $v_t \in \{(1,2), ③\}$.
  Idea: in observations we cannot distinguish ① and ②.
- Finite state space $\Longrightarrow$ Markov kernels can be identified with matrices

$$\lambda_i = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \kappa_{s,t} = \begin{bmatrix} 1-\theta & \theta & 0 \\ 0.25 & 0.5 & 0.25 \\ 0.4 & 0.3 & 0.3 \end{bmatrix},$$

for $i \in \{1, 2, 3\}$, $s \in \{0, 1, 3\}$ and $t \in \mathrm{ch}(s)$.

Example: finite state space

- Suppose $x_t \in \{1, 2, 3\}$ and $v_t \in \{1{,}2,\ 3\}$.
  Idea: in observations we cannot distinguish 1 and 2.
- Finite state space $\Longrightarrow$ Markov kernels can be identified with matrices

$$\lambda_i = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \kappa_{s,t} = \begin{bmatrix} 1-\theta & \theta & 0 \\ 0.25 & 0.5 & 0.25 \\ 0.4 & 0.3 & 0.3 \end{bmatrix},$$

  for $i \in \{1, 2, 3\}$, $s \in \{0, 1, 3\}$ and $t \in \mathrm{ch}(s)$.
- Prior on initial state: set $x_{-1} = 0$ and

$$\kappa_{-1,0} = [\pi_1,\ \pi_2,\ \pi_3] =: \boldsymbol{\pi}.$$

Backward Information Filter (BIF)

- **BIF**: efficient way to compute $x \mapsto h_t(x)$.

Backward Information Filter (BIF)

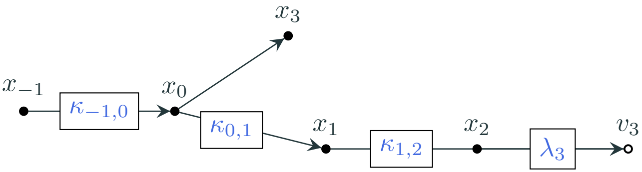
- **BIF**: efficient way to compute $x \mapsto h_t(x)$.
- ⚠ For finite state space this map can be identified with a vector h_t.

Backward Information Filter (BIF)

- **BIF**: efficient way to compute $x \mapsto h_t(x)$.
- ⚠ For finite state space this map can be identified with a vector h_t.
- Initialise from observations: for $t = 1, 2, 3$

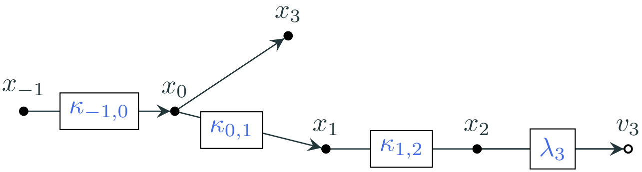
$$h_t^{\text{obs}} := \begin{bmatrix} 1 \\ 0 \end{bmatrix} \mathbf{1}\{v_t = 1{,}2\} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{1}\{v_t = 3\}.$$

Pullback along edges



$$h_2 = \lambda_3 h_3^{\mathrm{obs}} \qquad\qquad h_1 = \kappa_{1,2} h_2.$$

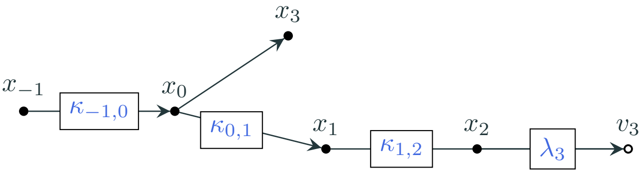
Pullback along edges



$h_2 = \lambda_3 h_3^{\text{obs}}$ $\qquad\qquad$ $h_1 = \kappa_{1,2} h_2.$

$$h_1(x_1) = p(v_3 \mid x_1) \quad =$$

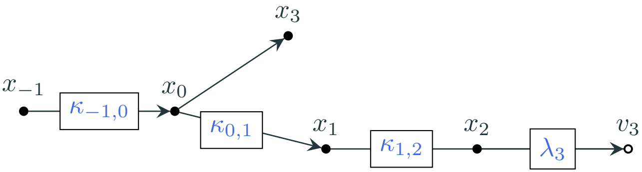
Pullback along edges



$$h_2 = \lambda_3 h_3^{\text{obs}} \qquad\qquad h_1 = \kappa_{1,2} h_2.$$

$$h_1(x_1) = p(v_3 \mid x_1) \quad = \quad \sum_{x_2} p(v_3, x_2 \mid x_1)$$

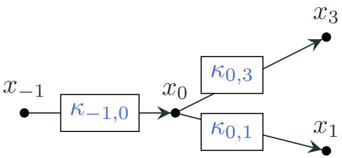
Pullback along edges



$$h_2 = \lambda_3 h_3^{\text{obs}} \qquad\qquad h_1 = \kappa_{1,2} h_2.$$

$$\begin{aligned} h_1(x_1) = p(v_3 \mid x_1) &= \sum_{x_2} p(v_3, x_2 \mid x_1) \\ &= \sum_{x_2} \underbrace{p(v_3 \mid \cancel{x_1}, x_2)}_{h_2(x_2)} p(x_2 \mid x_1). \end{aligned}$$

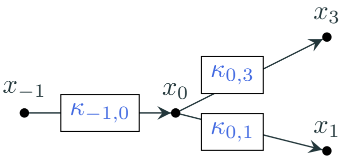
Backward Information Filter (BIF)



Get

$$h_{0\to 3} = \kappa_{0,3} h_3 \quad \text{and} \quad h_{0\to 1} = \kappa_{0,1} h_1$$

Backward Information Filter (BIF)



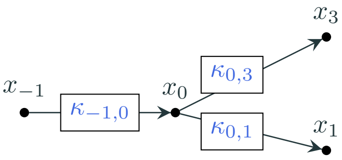
Get

$$h_{0\to 3} = \kappa_{0,3} h_3 \quad \text{and} \quad h_{0\to 1} = \kappa_{0,1} h_1$$

Fusion: by conditional independence of children we have

$$h_0(x) = h_{0\to 1}(x) h_{0\to 3}(x).$$

Backward Information Filter (BIF)



Get

$$h_{0\to 3} = \kappa_{0,3} h_3 \quad \text{and} \quad h_{0\to 1} = \kappa_{0,1} h_1$$

Fusion: by conditional independence of children we have

$$h_0(x) = h_{0\to 1}(x) h_{0\to 3}(x).$$

By identifying with vectors

$$h_0 = h_{0\to 1} \odot h_{0\to 3}.$$

Backward Information Filter (BIF)

- Likelihood: $L(\theta) := h_{-1} = \kappa_{-1,0} h_0$.

Backward Information Filter (BIF)

- Likelihood: $L(\theta) := h_{-1} = \kappa_{-1,0} h_0$.
- Forward simulate:

$$x_t^\star \mid x_s^\star = i \sim \mathsf{Cat}(\kappa_{s,t}[i,] \odot h_t), \qquad t \in \mathrm{ch}(s).$$

Backward Information Filter (BIF)

- Likelihood: $L(\theta) := h_{-1} = \kappa_{-1,0} h_0$.
- Forward simulate:

$$x_t^\star \mid x_s^\star = i \sim \mathsf{Cat}(\kappa_{s,t}[i,] \odot h_t), \qquad t \in \mathrm{ch}(s).$$

⚠ This is all tractable because

1. the DAG is a directed tree;
2. the state space is finite.

There is essentially just one operation...

- Along an edge compute

$$h(x) = \int \kappa(x, \mathrm{d}y) h(y).$$

There is essentially just one operation...

- Along an edge compute

$$h(x) = \int \kappa(x,\, \mathrm{d}y) h(y).$$

- At splits, make k-fold copy using kernel

$$\lhd_k(x,\, \mathrm{d}y) = \delta_x(\,\mathrm{d}y_1\,) \ldots, \delta_x(\,\mathrm{d}y_k\,).$$

There is essentially just one operation...

- Along an edge compute

$$h(x) = \int \kappa(x,\, \mathrm{d}y) h(y).$$

- At splits, make k-fold copy using kernel

$$\lhd_k(x,\, \mathrm{d}y) = \delta_x(\,\mathrm{d}y_1\,) \ldots, \delta_x(\,\mathrm{d}y_k\,).$$

Then in BIF, with $h_{1,3}(y) = h_{0\to 1}(y_1) h_{0\to 3}(y_2)$, we get

$$h_0(x) = \int \lhd_2(x,\, \mathrm{d}y) h_{1,3}(y) =$$

There is essentially just one operation...

- Along an edge compute

$$h(x) = \int \kappa(x,\, \mathrm{d}y) h(y).$$

- At splits, make k-fold copy using kernel

$$\lhd_k(x,\, \mathrm{d}y) = \delta_x(\,\mathrm{d}y_1) \ldots, \delta_x(\,\mathrm{d}y_k).$$

Then in BIF, with $h_{1,3}(y) = h_{0 \to 1}(y_1) h_{0 \to 3}(y_2)$, we get

$$h_0(x) = \int \lhd_2(x,\, \mathrm{d}y) h_{1,3}(y) = h_{0 \to 1}(x) h_{0 \to 3}(x).$$

Natural to convert DAG to string diagram...

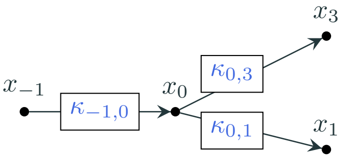
Guided process

Backward Information Filter (BIF)

Key idea: replace $h_{s \to t}$ by $g_{s \to t}$ that makes BIF tractable.

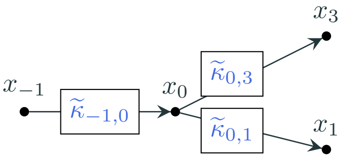
Backward Information Filter (BIF)

Key idea: replace $h_{s\to t}$ by $g_{s\to t}$ that makes BIF tractable.



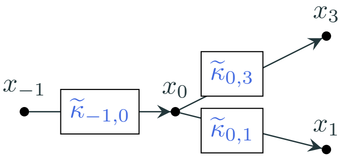
Backward Information Filter (BIF)

Key idea: replace $h_{s \to t}$ by $g_{s \to t}$ that makes BIF tractable.



Backward Information Filter (BIF)

Key idea: replace $h_{s \to t}$ by $g_{s \to t}$ that makes BIF tractable.

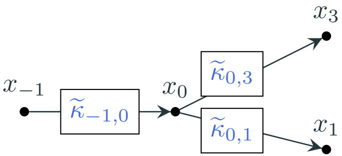


Get

$$h_{0 \to 3} = \kappa_{0,3} h_3 \quad \text{and} \quad h_{0 \to 1} = \kappa_{0,1} h_1$$

Backward Information Filter (BIF)

Key idea: replace $h_{s \to t}$ by $g_{s \to t}$ that makes BIF tractable.

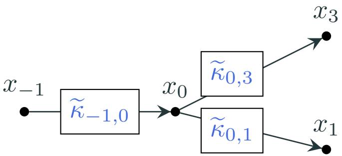


Get

$$h_{0 \to 3} = \widetilde{\kappa}_{0,3} h_3 \quad \text{and} \quad h_{0 \to 1} = \widetilde{\kappa}_{0,1} h_1$$

Backward Information Filter (BIF)

Key idea: replace $h_{s \to t}$ by $g_{s \to t}$ that makes BIF tractable.

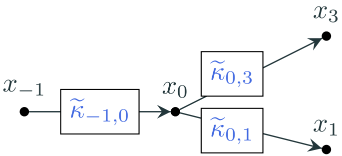


Get

$$g_{0 \to 3} = \widetilde{\kappa}_{0,3} g_3 \quad \text{and} \quad g_{0 \to 1} = \widetilde{\kappa}_{0,1} g_1$$

Backward Information Filter (BIF)

Key idea: replace $h_{s \to t}$ by $g_{s \to t}$ that makes BIF tractable.



Get

$$g_{0 \to 3} = \widetilde{\kappa}_{0,3} g_3 \quad \text{and} \quad g_{0 \to 1} = \widetilde{\kappa}_{0,1} g_1$$

Fusion: by conditional independence of children we have

$$g_0(x) = g_{0 \to 1}(x) g_{0 \to 3}(x).$$

By identifying with vectors

$$g_0 = g_{0 \to 1} \odot g_{0 \to 3}.$$

Guided process

Let the maps $x \mapsto g_{s \to t}(x)$ be specified for each edge (s, t) and define

$$g_s(x) = \prod_{t \in \mathrm{ch}(s)} g_{s \to t}(x), \qquad s \in \mathcal{S}_0. \tag{1}$$

Guided process

Let the maps $x \mapsto g_{s \to t}(x)$ be specified for each edge (s, t) and define

$$g_s(x) = \prod_{t \in \mathrm{ch}(s)} g_{s \to t}(x), \qquad s \in \mathcal{S}_0. \tag{1}$$

Practical way to choose $g_{s \to t}$: replace kernel $\kappa_{s \to t}$ by approximation $\widetilde{\kappa}_{s \to t}$.

Guided process

Let the maps $x \mapsto g_{s \to t}(x)$ be specified for each edge (s, t) and define

$$g_s(x) = \prod_{t \in \mathrm{ch}(s)} g_{s \to t}(x), \qquad s \in \mathcal{S}_0. \tag{1}$$

Practical way to choose $g_{s \to t}$: replace kernel $\kappa_{s \to t}$ by approximation $\widetilde{\kappa}_{s \to t}$.

Define the guided process X° as the process starting in $X_0^\circ = x_0$ and from the roots onwards evolving *on* the DAG $\mathcal{G}$ according to transition kernel

$$\kappa^\circ_{\mathrm{pa}(s) \to s}(x_{\mathrm{pa}(s)}; \mathrm{d}y) = \frac{g_s(y) \kappa_{\mathrm{pa}(s) \to s}(x_{\mathrm{pa}(s)}; \mathrm{d}y)}{\int g_s(y) \kappa_{\mathrm{pa}(s) \to s}(x_{\mathrm{pa}(s)}; \mathrm{d}y)}, \qquad s \in \mathcal{S}.$$

Use of guided process

Let $\mathcal{S}$ denote the set of non-leaf vertices.

Theorem. Assume kernels towards leaf-nodes admit densities $h_{\mathrm{pa}(v)\to v}$. Then

$$h_0(x_0) = g_0(x_0)\mathbb{E}\left[\prod_{s\in\mathcal{S}} w_{\mathrm{pa}(s)\to s}(X^\circ_{\mathrm{pa}(s)}) \prod_{v\in\mathcal{V}} \frac{h_{\mathrm{pa}(v)\to v}(X^\circ_{\mathrm{pa}(v)})}{g_{\mathrm{pa}(v)\to v}(X^\circ_{\mathrm{pa}(v)})}\right]$$

with weights defined by

Use of guided process

Let $\mathcal{S}$ denote the set of non-leaf vertices.

Theorem. Assume kernels towards leaf-nodes admit densities $h_{\mathrm{pa}(v)\to v}$. Then

$$h_0(x_0) = g_0(x_0)\mathbb{E}\left[\prod_{s\in\mathcal{S}} w_{\mathrm{pa}(s)\to s}(X^\circ_{\mathrm{pa}(s)}) \prod_{v\in\mathcal{V}} \frac{h_{\mathrm{pa}(v)\to v}(X^\circ_{\mathrm{pa}(v)})}{g_{\mathrm{pa}(v)\to v}(X^\circ_{\mathrm{pa}(v)})}\right]$$

with weights defined by

$$w_{\mathrm{pa}(s)\to s}(x_{\mathrm{pa}(s)}) = \frac{\int g_s(y)\kappa_{\mathrm{pa}(s)\to s}(x_{\mathrm{pa}(s)}; \mathrm{d}y)}{\prod_{u\in\mathrm{pa}(s)} g_{u\to s}(x_u)} \qquad s\in\mathcal{S}.$$

Use of guided process

Let $\mathcal{S}$ denote the set of non-leaf vertices.

Theorem. Assume kernels towards leaf-nodes admit densities $h_{\mathrm{pa}(v)\to v}$. Then

$$h_0(x_0) = g_0(x_0)\mathbb{E}\left[\prod_{s\in\mathcal{S}} w_{\mathrm{pa}(s)\to s}(X^\circ_{\mathrm{pa}(s)}) \prod_{v\in\mathcal{V}} \frac{h_{\mathrm{pa}(v)\to v}(X^\circ_{\mathrm{pa}(v)})}{g_{\mathrm{pa}(v)\to v}(X^\circ_{\mathrm{pa}(v)})}\right]$$

with weights defined by

$$w_{\mathrm{pa}(s)\to s}(x_{\mathrm{pa}(s)}) = \frac{\int g_s(y)\kappa_{\mathrm{pa}(s)\to s}(x_{\mathrm{pa}(s)}; \mathrm{d}y)}{\prod_{u\in\mathrm{pa}(s)} g_{u\to s}(x_u)} \qquad s \in \mathcal{S}.$$

Computationally, this implies a bidirectional scheme:

1. Backward pass for Filtering;
2. Forward pass for Guiding.

Wrap-up

- If the state space is finite, BIF provides the likelihood.
- Key to tractability is that h can always be represented as a vector.
- ⚠ In general BIF is intractable.

Wrap-up

- If the state space is finite, BIF provides the likelihood.
- Key to tractability is that h can always be represented as a vector.
- ⚠ In general BIF is intractable.
- Resolve by backward filtering with simpler kernels and forward simulating the corresponding guided process.
- This results in weighted samples from the conditioned process.

Application: interacting particle process

Forward transitions:

$$\kappa_i(t,x) = \begin{bmatrix} \psi\left(\lambda N_i(t,x)\right) & 1-\psi\left(\lambda N_i(t,x)\right) & 0 \\ 0 & \psi(\mu) & 1-\psi(\mu) \\ 1-\psi(\nu) & 0 & \psi(\nu) \end{bmatrix},$$

where

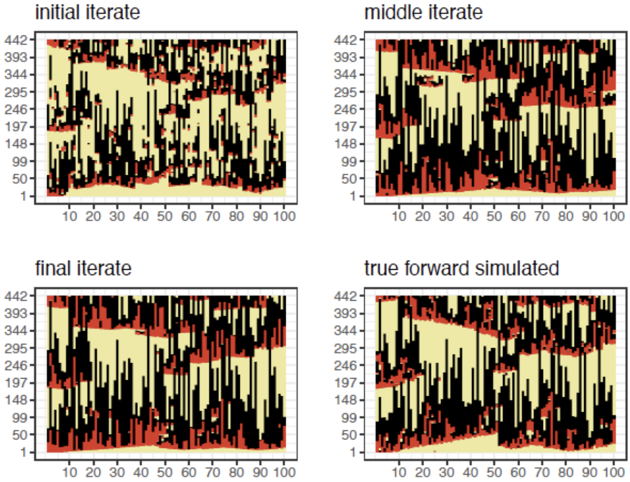
$$N_i(x) = \{\text{number of infected neighbours of individual } i \text{ in state } x\}$$

and $\psi(u) = \exp(-\tau u)$.

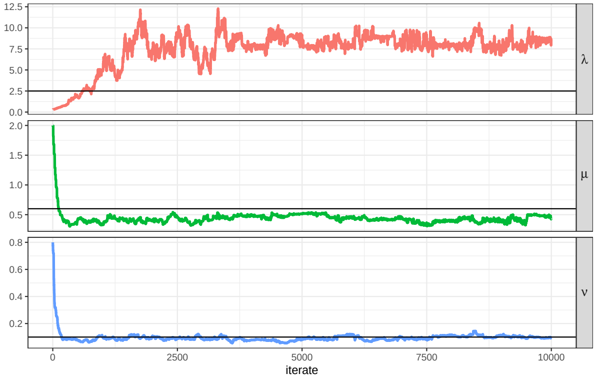
Auxiliary kernel for backward filtering:

$$\widetilde{\kappa}_i = \begin{bmatrix} \psi(\widetilde{\lambda}_i(t)) & 1-\psi(\widetilde{\lambda}_i(t)) & 0 \\ 0 & \psi(\mu) & 1-\psi(\mu) \\ 1-\psi(\nu) & 0 & \psi(\nu) \end{bmatrix}.$$

Application: interacting particle process



Application: interacting particle process



Continuous time transitions

Rethinking the discrete-time case:

- Edge

$$x_S \longrightarrow x_T$$

Suppose $x \mapsto h(T, x)$ is given; wish to find $x \mapsto h(S, x)$.

Continuous time transitions

Rethinking the discrete-time case:

- Edge

  $$x_S \longrightarrow x_T$$

  Suppose $x \mapsto h(T, x)$ is given; wish to find $x \mapsto h(S, x)$.

- "Discrete-time" generator

$$(\mathcal{A}h)(S, x) : \quad = \quad \mathbb{E}[h(T, X_T) - h(S, X_S) \mid X_S = x]$$

Continuous time transitions

Rethinking the discrete-time case:

- Edge

  $$x_S \longrightarrow x_T$$

  Suppose $x \mapsto h(T, x)$ is given; wish to find $x \mapsto h(S, x)$.

- "Discrete-time" generator

$$\begin{aligned}(\mathcal{A}h)(S,x) : \quad &= \quad \mathbb{E}[h(T, X_T) - h(S, X_S) \mid X_S = x] \\ &= \quad \int h(T,y)\kappa_{S \to T}(x, \mathrm{d}y) - h(S,x).\end{aligned}$$

- ⚠ Obtain $x \mapsto h(S, x)$ by solving $(\mathcal{A}h)(S, x) = 0$.

Continuous time transitions

Define the infinitesimal generator of the space-time process (t, X_t): for $S \leq s < s + h \leq T$

$$
\begin{aligned}
(\mathcal{A}h)(s,x) &= \lim_{h \downarrow 0} h^{-1} \mathbb{E}[h(s+h, X_{s+h}) - h(s, X_s) \mid X_s = x] \\
&=
\end{aligned}
$$

Continuous time transitions

Define the infinitesimal generator of the space-time process (t, X_t): for $S \leq s < s + h \leq T$

$$
\begin{aligned}
(\mathcal{A}h)(s,x) &= \lim_{h \downarrow 0} h^{-1} \mathbb{E}[h(s+h, X_{s+h}) - h(s, X_s) \mid X_s = x] \\
&= (\mathcal{L}h)(s,x) + \frac{\partial}{\partial s} h(s,x).
\end{aligned}
$$

Continuous time transitions

Define the infinitesimal generator of the space-time process (t, X_t): for $S \leq s < s + h \leq T$

$$\begin{aligned}(\mathcal{A}h)(s,x) &= \lim_{h \downarrow 0} h^{-1} \mathbb{E}[h(s+h, X_{s+h}) - h(s, X_s) \mid X_s = x] \\ &= (\mathcal{L}h)(s,x) + \frac{\partial}{\partial s} h(s,x).\end{aligned}$$

- Obtain $x \mapsto h(S,x)$ from solving

$$(\mathcal{A}h)(s,x) = 0 \quad \text{subject to} \quad h(T,\cdot).$$

Continuous time transitions

Define the infinitesimal generator of the space-time process (t, X_t): for $S \leq s < s + h \leq T$

$$\begin{aligned}(\mathcal{A}h)(s,x) &= \lim_{h \downarrow 0} h^{-1} \mathbb{E}[h(s+h, X_{s+h}) - h(s, X_s) \mid X_s = x] \\ &= (\mathcal{L}h)(s,x) + \frac{\partial}{\partial s} h(s,x).\end{aligned}$$

- Obtain $x \mapsto h(S,x)$ from solving

$$(\mathcal{A}h)(s,x) = 0 \quad \text{subject to} \quad h(T, \cdot).$$

- h induces a change of measure from X to the process $X^\star$ with inf. generator

$$h\mathcal{L}^\star f = \mathcal{L}(fh) - f\mathcal{L}h.$$

Continuous time transitions

Define the infinitesimal generator of the space-time process (t, X_t): for $S \leq s < s+h \leq T$

$$\begin{aligned}(\mathcal{A}h)(s,x) &= \lim_{h\downarrow 0} h^{-1}\mathbb{E}[h(s+h, X_{s+h}) - h(s, X_s) \mid X_s = x] \\ &= (\mathcal{L}h)(s,x) + \frac{\partial}{\partial s} h(s,x).\end{aligned}$$

- Obtain $x \mapsto h(S,x)$ from solving

$$(\mathcal{A}h)(s,x) = 0 \quad \text{subject to} \quad h(T,\cdot).$$

- h induces a change of measure from X to the process $X^\star$ with inf. generator

$$h\mathcal{L}^\star f = \mathcal{L}(fh) - f\mathcal{L}h.$$

⚠ Solving Kolmogorov backward equation is usually intractable.

Defining the guided process via its inf.generator

- Backward filter with $\widetilde{\mathcal{L}}$ instead of $\mathcal{L}$, such that solving $(\widetilde{\mathcal{L}}g)(s,x) + \frac{\partial}{\partial s} g(s,x) = 0$ becomes tractable.

Defining the guided process via its inf.generator

- Backward filter with $\widetilde{\mathcal{L}}$ instead of $\mathcal{L}$, such that solving $(\widetilde{\mathcal{L}}g)(s,x) + \frac{\partial}{\partial s}g(s,x) = 0$ becomes tractable.
- g induces a change of measure from X to X° with inf. generator

$$g\mathcal{L}^\circ f = \mathcal{L}(fg) - f\mathcal{L}g$$

  Identify guided process from $\mathcal{L}^\circ$.

Defining the guided process via its inf.generator

- Backward filter with $\widetilde{\mathcal{L}}$ instead of $\mathcal{L}$, such that solving $(\widetilde{\mathcal{L}}g)(s,x) + \frac{\partial}{\partial s} g(s,x) = 0$ becomes tractable.
- g induces a change of measure from X to X° with inf. generator

$$g\mathcal{L}^\circ f = \mathcal{L}(fg) - f\mathcal{L}g$$

  Identify guided process from $\mathcal{L}^\circ$.

- Correct for "wrong" h by weight

$$\exp\left(\int_{t_i}^{t_{i+1}} \frac{(\mathcal{L} - \widetilde{\mathcal{L}})g}{g}(u, X_u^\circ)\, \mathrm{d}u\right).$$

- Care is needed in choice of $\widetilde{\mathcal{L}}$: matching conditions.

How to solve the Kolmogorov Backward Equation?

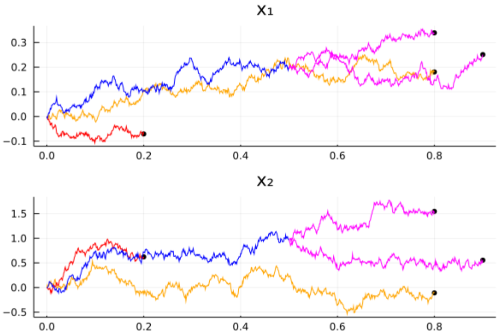
For $s \in (S, T]$,

$$(\widetilde{\mathcal{L}}g)(s,x) + \frac{\partial}{\partial s} g(s,x) = 0, \qquad g(T,\cdot) = g_T(\cdot).$$

Examples/strategies:

1. If $\widetilde{\mathcal{L}}$ is the infinitesimal generator of a linear diffusion process, then $\log g(t,x) = c(t) + F(t)'x + x'H(t)x$ with ODE-system for $(H(t), F(t), c(t))$.
2. Ansatz $g(t,x) = \sum_j c(t)\psi_j(t)$. Derive ODE for $c(t)$.

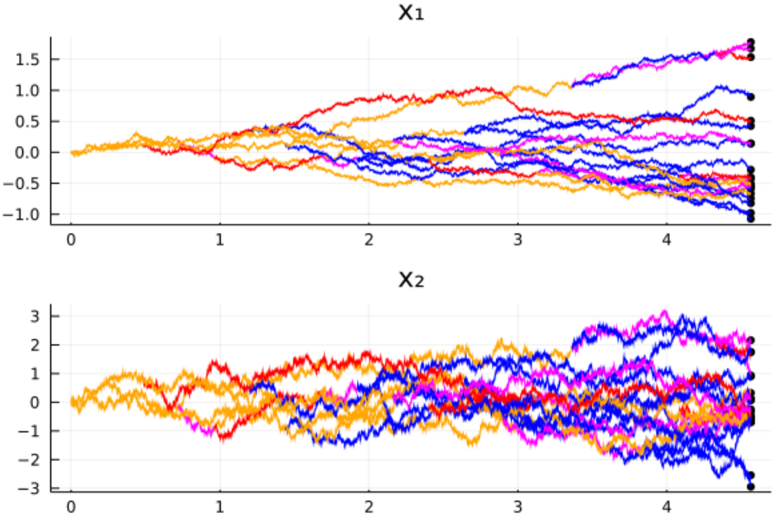
Example: branching diffusion



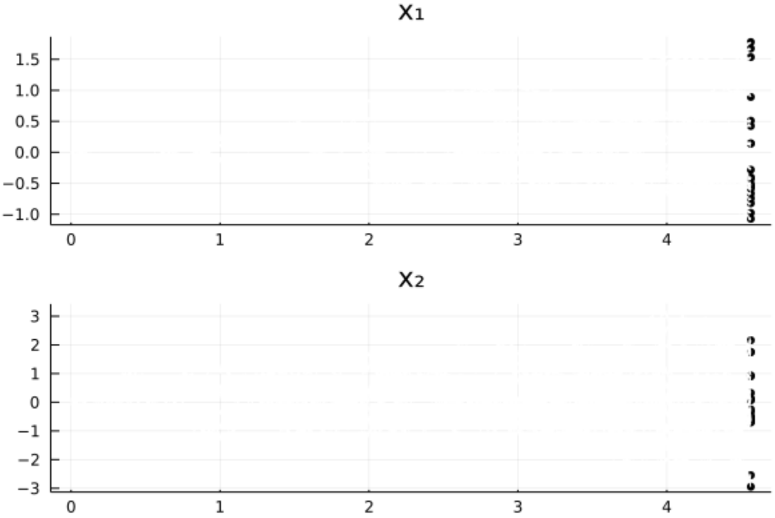
SDE on a tree where on each branch

$$\mathrm{d}X_t = \tanh . \left(\begin{bmatrix} -\theta_1 & \theta_1 \\ \theta_2 & -\theta_2 \end{bmatrix} X_t \right) \mathrm{d}t + \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \mathrm{d}W_t .$$

Numerical illustration: SDE on a tree



Numerical illustration: SDE on a tree



Numerical illustration: SDE on a tree

On each branch

$$\mathrm{d}X_t = \tanh.\left(\begin{bmatrix} -\theta_1 & \theta_1 \\ \theta_2 & -\theta_2 \end{bmatrix} X_t\right) \mathrm{d}t + \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \mathrm{d}W_t.$$

Numerical illustration: SDE on a tree

On each branch

$$\mathrm{d}X_t = \tanh . \left(\begin{bmatrix} -\theta_1 & \theta_1 \\ \theta_2 & -\theta_2 \end{bmatrix} X_t \right) \mathrm{d}t + \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \mathrm{d}W_t.$$

- Backward filter a linear process (essentially $\widetilde{\kappa}$)

Numerical illustration: SDE on a tree

On each branch

$$\mathrm{d}X_t = \tanh . \left(\begin{bmatrix} -\theta_1 & \theta_1 \\ \theta_2 & -\theta_2 \end{bmatrix} X_t \right) \mathrm{d}t + \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \mathrm{d}W_t.$$

- Backward filter a linear process (essentially $\widetilde{\kappa}$)
- Write X° as pushforward of (x_0, ξ, Z), with $\xi = (\theta_1, \theta_2, \sigma_1, \sigma_2)$
- MCMC on (ξ, Z)

Numerical illustration: SDE on a tree

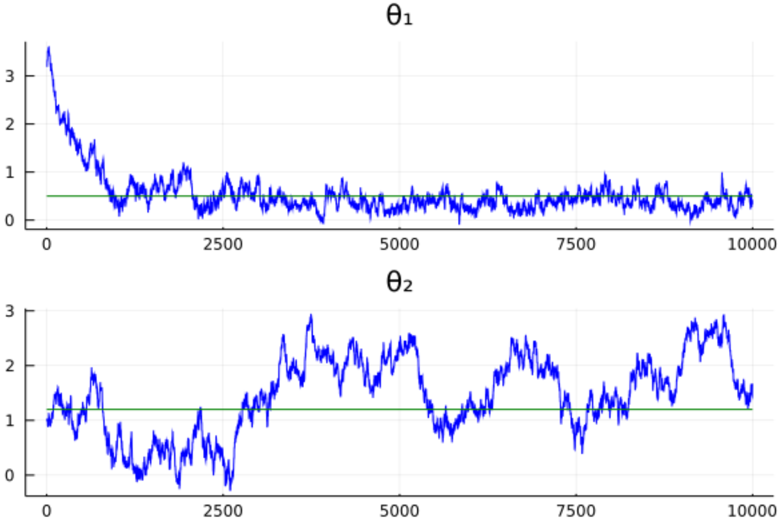
On each branch

$$\mathrm{d}X_t = \tanh.\left(\begin{bmatrix} -\theta_1 & \theta_1 \\ \theta_2 & -\theta_2 \end{bmatrix} X_t\right)\mathrm{d}t + \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}\mathrm{d}W_t.$$

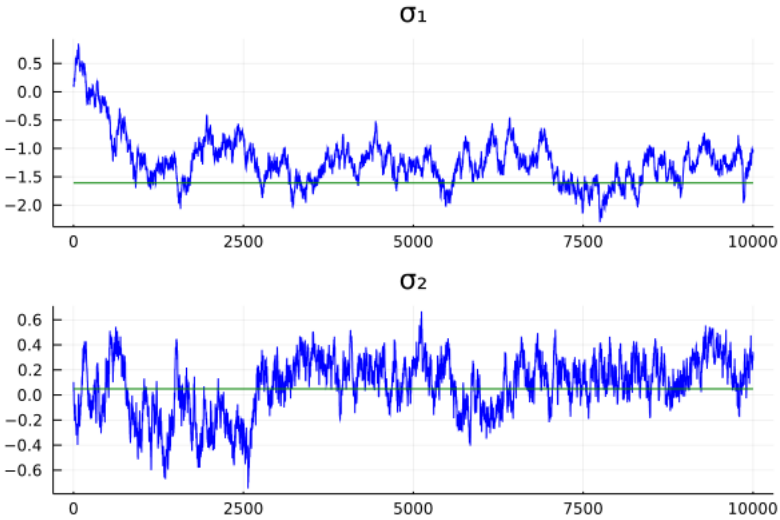
- Backward filter a linear process (essentially $\widetilde{\kappa}$)
- Write X° as pushforward of (x_0, ξ, Z), with $\xi = (\theta_1, \theta_2, \sigma_1, \sigma_2)$
- MCMC on (ξ, Z)

Implementation in `MitosisStochasticDiffEq.jl` by Frank Schäfer (MIT).

Numerical illustration: SDE on a tree



Numerical illustration: SDE on a tree



Wrap-up / conclusions

Wrap-up

Backward Filtering Forward Guiding: framework for doing likelihood based inference in directed acyclic graphs, where transitions over edges may correspond to the evolution of a stochastic process for a certain time span.

Wrap-up

Backward Filtering Forward Guiding: framework for doing likelihood based inference in directed acyclic graphs, where transitions over edges may correspond to the evolution of a stochastic process for a certain time span.

- Defining guided processes on graphical models (for "non-tree"-case: see preprint).
- Both discrete-time and continuous-time transitions incorporated.

Wrap-up

Backward Filtering Forward Guiding: framework for doing likelihood based inference in directed acyclic graphs, where transitions over edges may correspond to the evolution of a stochastic process for a certain time span.

- Defining guided processes on graphical models (for “non-tree”-case: see preprint).
- Both discrete-time and continuous-time transitions incorporated.
- Illustrations for interacting particle process and branching diffusion.
- Not covered: compositionality results (some category theory, see earlier versions on arXiv).

Wrap-up

Backward Filtering Forward Guiding: framework for doing likelihood based inference in directed acyclic graphs, where transitions over edges may correspond to the evolution of a stochastic process for a certain time span.

- Defining guided processes on graphical models (for "non-tree"-case: see preprint).
- Both discrete-time and continuous-time transitions incorporated.
- Illustrations for interacting particle process and branching diffusion.
- Not covered: compositionality results (some category theory, see earlier versions on arXiv).

Ongoing: SPDEs, SDEs on manifolds, chemical reaction networks.

Open postdoc position at VU Amsterdam.

References

- Continuous-discrete smoothing of diffusions
  MIDER, SCHAUER, VDM, Electronic Journal of Statistics

  Bayesian inference for partially observed diffusions.

References

- Continuous-discrete smoothing of diffusions
  Mider, Schauer, VdM, Electronic Journal of Statistics

  Bayesian inference for partially observed diffusions.
- Automatic Backward Filtering Forward Guiding for Markov processes and graphical models, VdM and Schauer, preprint on arXiv.

  A generalisation to Markov processes on graphical models including ideas on compositionality from category theory.

References

- Continuous-discrete smoothing of diffusions
  Mider, Schauer, VdM, Electronic Journal of Statistics

  Bayesian inference for partially observed diffusions.
- Automatic Backward Filtering Forward Guiding for Markov processes and graphical models, VdM and Schauer, preprint on arXiv.

  A generalisation to Markov processes on graphical models including ideas on compositionality from category theory.
- Introduction to Automatic Backward Filtering Forward Guiding, VdM, arXiv, submitted.

  Gentle introduction to the more advanced paper.

References

- Continuous-discrete smoothing of diffusions
  Mider, Schauer, VdM, Electronic Journal of Statistics

  Bayesian inference for partially observed diffusions.

- Automatic Backward Filtering Forward Guiding for Markov processes and graphical models, VdM and Schauer, preprint on arXiv.

  A generalisation to Markov processes on graphical models including ideas on compositionality from category theory.

- Introduction to Automatic Backward Filtering Forward Guiding, VdM, arXiv, submitted.

  Gentle introduction to the more advanced paper.

- Inference in Hidden Markov Models, Cappé, Moulines and Rydén

  Good source on filtering, smoothing, parameter estimation in HMM.