Importance sampling for online variational learning

Mathis Chagneux[‡], Pierre Gloaguen[⋆], Sylvain Le Corff[†], and Jimmy Olsson[*]

[‡]Télécom Paris, Institut Polytechnique de Paris.

[*]LMBA,Université Bretagne Sud.

[†]LPSM, Sorbonne Université, UMR CNRS 8001.

[*]Department of Mathematics, KTH Royal Institute of Technology.

Abstract

This article addresses online variational estimation in state-space models. We focus on learning the smoothing distribution, i.e. the joint distribution of the latent states given the observations, using a variational approach together with Monte Carlo importance sampling. We propose an efficient algorithm for computing the gradient of the evidence lower bound (ELBO) in the context of streaming data, where observations arrive sequentially. Our contributions include a computationally efficient online ELBO estimator, demonstrated performance in offline and true online settings, and adaptability for computing general expectations under joint smoothing distributions.

1 Introduction

This article considers online variational estimation problems in state-space models (SSMs), where the observations $Y_{0:t}$[1] depend on a hidden Markov chain denoted by $X_{0:t}$. In this setting, a classical goal is to learn the *smoothing distribution*, which is the conditional distribution of $X_{0:t}$ given $Y_{0:t}$. A popular approach to approximate such a posterior distribution is to rely on particle smoothing, see Chopin et al. (2020) and the references therein, but the latter suffers from the curse of dimensionality in the state dimension Bengtsson et al. (2008), which has recently motivated the use of variational inference as an alternative. Here, the posterior distribution is approximated by a distribution $q^\lambda_{0:t}$, depending on some unknown parameter $\lambda \in \Lambda$. This parameter is then trained by maximizing the evidence lower bound (ELBO):

$$\mathcal{L}^\lambda_t = \mathbb{E}_{q^\lambda_{0:t}} \left[\log \frac{p_{0:t}(X_{0:t}, Y_{0:t})}{q^\lambda_{0:t}(X_{0:t})} \right] , \tag{1}$$

where $p_{0:t}$ is the joint distribution of the hidden states and the observations. Classical optimization procedures rely on the gradient of $\mathcal{L}^\lambda_t$ with respect to λ. Most associated works can be divided into two categories. On the one hand, a large body of methods are only amenable to offline estimation Krishnan et al. (2017); Lin et al. (2018a); Johnson et al. (2016), i.e. one needs to have access the entire sequence of observations $Y_{0:t}$ beforehand to compute the ELBO and its gradients. On the other hand, some methods provide online procedures via alternative variational objectives which depart from (1) using additional assumptions on the state dependencies under the variational law Marino et al. (2018); Dowling et al. (2023). In this paper, we

[1] In all the paper, $a_{u:v}$ is a short-hand notation for $(a_u, \dots, a_v)$.

propose an efficient algorithm to update the actual gradient of $\mathcal{L}_t^\lambda$ in the context of streaming data, when observations arrive on the fly and are processed only once.

- We propose a computationally efficient online estimator of the ELBO when the variational distribution has a Markov structure. Contrary to computationally intensive Sequential or Markov Chain Monte Carlo methods, our algorithm relies on simple i.i.d samples from the variational distribution.
- Experimentally, we demonstrate the performance of our estimators in the offline setting, and assess its performance in online settings with many observations to learn time series representations.
- The proposed algorithm is not limited to the online computation and optimization of the ELBO and can be directly adapted to compute more general expectations under distributions admitting a Markovian structure.

2 Model and learning task

Consider a SSM where the hidden Markov chain in $\mathbb{R}^{d_x}$ is denoted by $(X_t)_{t\geqslant 0}$. The distribution of X_0 has density χ with respect to the Lebesgue measure μ and for all $t \geqslant 0$, the conditional distribution of X_{t+1} given $X_{0:t}$ has density $m_t(X_t, \cdot)$. In SSMs, it is assumed that this state is partially observed through an observation process $(Y_t)_{0\leqslant t\leqslant T}$ taking on values in $\mathbb{R}^{d_y}$. The observations $Y_{0:t}$ are assumed to be independent conditionally on $X_{0:t}$ and, for all $0 \leqslant t \leqslant T$, the distribution of Y_t given $X_{0:t}$ depends on X_t only and has density $g_t(X_t, \cdot)$ with respect to the Lebesgue measure. The whole model is then defined by the joint distribution of hidden states and observations:

$$p_{0:t}(x_{0:t}, y_{0:t}) = \prod_{s=0}^{t} \ell_s(x_{s-1}, x_s, y_s),$$

where $\ell_s(x_{s-1}, x_s, y_s) := m_s(x_{s-1}, x_s) g_s(x_{s-1}, y_s)$ for $s \geq 1$ and $\ell_0(x_{-1}, x_0, y_0) := \chi(x_0) g_0(x_0, y_0)$.

2.1 Smoothing in latent data models

A classical learning task in SSMs is *state inference*, which is to estimate the smoothing distribution, *i.e.*, the conditional p.d.f. of $X_{0:t}$ given $Y_{0:t}$. This distribution is given by

$$\phi_{0:t}(x_{0:t}) \propto p_{0:t}(x_{0:t}, y_{0:t}) \ .$$

The marginal at time t of this joint distribution is known as the *filtering distribution* at time t, and its density w.r.t. the Lebesgue measure is written ϕ_t. It is straightforward to express the joint smoothing density through the following *backward factorization*:

$$\phi_{0:t}(x_{0:t}) = \phi_t(x_t) \prod_{s=1}^{t} b_{s-1|s}(x_s, x_{s-1}) \ . \tag{2}$$

where, for $1 \leqslant s \leqslant t$,

$$b_{s-1|s}(x_s, x_{s-1}) = \frac{m_s(x_{s-1}, x_s)\phi_{s-1}(x_{s-1})}{\int m_s(x_{s-1}, x_s)\phi_{s-1}(x_{s-1}) \, \mathrm{d}x_{s-1}}, \tag{3}$$

referred to as the *backward kernels*, provide the conditional p.d.f. of X_{s-1} given $(X_s, Y_{0:s-1})$. It is worth noting that (3) emphasizes the Markov structure of the smoothing distribution. Unfortunately, this distribution lacks generally a closed-form expression due to the integral in the denominator and the intractability of the filtering distributions.

2.2 Backward sequential variational inference

In variational approaches the smoothing distribution $\phi_{0:t}$ is approximated by choosing a candidate in a parametric family $\{q^\lambda_{0:t}\}_{\lambda\in\Lambda}$, referred to as the *variational* family, where Λ is a parameter set.

A critical point therefore lies in the form of the variational family. Motivated by the Markov structure of the smoothing distribution, most works impose structure on the variational family via a factorized decomposition of $q^\lambda_{0:t}$ over $x_{0:t}$. A variational counterpart of (2), introduced in the work of Campbell et al. (2021), is to define

$$q^\lambda_{0:t}(x_{0:t}) = q^\lambda_t(x_t) \prod_{s=1}^{t} q^\lambda_{s-1|s}(x_s, x_{s-1}) \,, \tag{4}$$

where q^λ_t (resp. $q^\lambda_{s-1|s}(x_s, \cdot)$) are user-chosen p.d.f. whose parameters depend on $Y_{0:t}$ (resp. $Y_{0:s-1}$). A key advantage of this factorization is that it respects the true dependencies involved in (3). Additionally, Chagneux et al. (2024) established an upper bound on the error when expectations w.r.t. the smoothing distribution are approximated by expectations w.r.t. variational distributions satisfying this backward factorization. In the following, we consider variational distributions satisfying (4). Variational inference Blei et al. (2017) then consists in learning the best λ by maximizing the ELBO given in (1). This is typically done via gradient ascent algorithms that requires to compute the gradient of the ELBO. The next sections depict a new algorithm to approximate this gradient recursively.

3 Recursion for the gradient of the ELBO

For concise notations, write[2]

$$\tilde{f}^\lambda_t(x_{t-1}, x_t) = \begin{cases} \log \ell_0(x_{-1}, x_0, y_0) & \text{if } t = 0, \\ \log \frac{\ell_t(x_{t-1}, x_t, y_t)}{q^\lambda_{t-1|t}(x_t, x_{t-1})} & \text{if } t > 0 \end{cases} \tag{5}$$

and $f^\lambda_{0:t}(x_{0:t}) = \sum_{s=0}^{t} \tilde{f}^\lambda_s(x_{s-1}, x_s)$. Then,

$$\mathcal{L}^\lambda_t = \mathbb{E}_{q^\lambda_{0:t}} \left[f^\lambda_{0:t}(X_{0:t}) - \log q^\lambda_t(X_t) \right].$$

In the following results, all gradients are computed w.r.t. λ.

Proposition 3.1. *The ELBO and its gradient satisfy:*

$$\mathcal{L}^\lambda_t = \mathbb{E}_{q^\lambda_t} \left[H^\lambda_t(X_t) \right] - \mathbb{E}_{q^\lambda_t} \left[\log q^\lambda_t(X_t) \right] \tag{6}$$

$$\nabla \mathcal{L}^\lambda_t = \mathbb{E}_{q^\lambda_t} \left[\{ \nabla \log q^\lambda_t \cdot H^\lambda_t \}(X_t) + G^\lambda_t(X_t) \right], \tag{7}$$

where $H^\lambda_t(x_t)$ is a function from $\mathbb{R}^{d_x}$ to $\mathbb{R}$ satisfying the recursion

$$H^\lambda_t(x_t) = \mathbb{E}_{q^\lambda_{t-1|t}(x_t, \cdot)} \left[H^\lambda_{t-1}(X_{t-1}) + \tilde{f}^\lambda_t(X_{t-1}, x_t) \right] \,, \tag{8}$$

[2]The dependency of each term $\tilde{f}^\lambda_t$ on y_t is omitted to lighten the notations.

with $H_0^\lambda(x_0) = \tilde{f}_0^\lambda(x_{-1}, x_0)$*, and* $G_t^\lambda(x_t)$ *is a function from* $\mathbb{R}^{d_x}$ *to* Λ *satisfying* $G_0^\lambda(x_0) = 0$ *and*

$$G_t^\lambda(x_t) = \mathbb{E}_{q_{t-1|t}^\lambda(x_t,\cdot)}\left[G_{t-1}^\lambda(X_{t-1}) + \nabla \log q_{t-1|t}^\lambda(x_t, X_{t-1}) \mathsf{H}_{t-1}^\lambda(X_{t-1}, x_t)\right] , \tag{9}$$

with $\mathsf{H}_{t-1}^\lambda(x_{t-1}, x_t) = H_{t-1}^\lambda(x_{t-1}) + \tilde{f}_t^\lambda(x_{t-1}, x_t)$.

Proof. The proof is postponed to Appendix A. □

For a fixed sequence of length t, recursive computation of $\nabla \mathcal{L}_t^\lambda$ therefore consists in (i) computing recursively $\nabla H_t^\lambda(x_t)$ from 0 to t using the recursions of Proposition 3.1 and (ii) computing the final expectation (7). As this final expectation is w.r.t. the chosen variational distribution q_t^λ, standard Monte Carlo sampling can be used to approximate it. It remains to approximate online the intermediate functions $H_t^\lambda(x_t)$ and $G_t^\lambda(x_t)$, involving conditional expectations with respect to the kernels $q_{t-1|t}^\lambda$. Note that these kernels aim to approximate the true backward kernels $b_{t-1|t}$, which depend only on observations $Y_{0:t-1}$, *i.e.*, only on the past, and hence, are prone to online learning.

4 Approximation of the gradient

4.1 Offline computation

For the sake of clarity, we first depict the algorithm for a fixed size sequence of observations $Y_{0:T}$. For a fixed λ, we use Proposition 3.1 to compute $\nabla \mathcal{L}_T^\lambda$. We suppose we have access to a sequence of variational distributions $(q_t^\lambda)_{t\geq 0}$ and $(q_{t-1|t}^\lambda)_{t\geq 1}$ which can be evaluated and sampled from easily. The choice of such sequences is discussed in Section 5.

Initialization. Simulate a N-sample $\{\xi_0^j\}_{1\leq j\leq N} \overset{i.i.d}{\sim} q_0^\lambda$, set $\hat{H}_0^{\lambda,j} = H_0^\lambda(\xi_0^j), \hat{G}_0^{\lambda,j} = G_0^\lambda(\xi_0^j) = 0$. The key point for the propagation step is that these two functionals are known (at $t = 0$) only on a *finite support*.

Recursive approximation of H_t^λ and G_t^λ. At time t, simulate a N-sample $\{\xi_t^i\}_{1\leq i\leq N} \overset{i.i.d}{\sim} q_t^\lambda$. $H_t^\lambda(\xi_t^i)$ and $G_t^\lambda(\xi_t^i)$ are approximated respectively by:

$$\hat{H}_t^{\lambda,i} = \sum_{j=1}^N \bar{w}_{t-1|t}^{\lambda,i,j}\left(\hat{H}_{t-1}^{\lambda,j} + \tilde{f}_t^\lambda(\xi_{t-1}^j, \xi_t^i)\right) , \tag{10}$$

$$\hat{G}_t^{\lambda,i} = \sum_{i=1}^N \bar{w}_{t-1|t}^{\lambda,i,j}\left\{\hat{G}_{t-1}^{\lambda,j} + \nabla \log q_{t-1|t}^\lambda(\xi_t^i, \xi_{t-1}^j)\left(\hat{H}_{t-1}^{\lambda,j} + \tilde{f}_t^\lambda(\xi_{t-1}^j, \xi_t^i)\right)\right\} , \tag{11}$$

where

$$\bar{w}_{t-1|t}^{\lambda,i,j} = \frac{q_{t-1|t}^\lambda(\xi_t^i, \xi_{t-1}^j)/q_{t-1}^\lambda(\xi_{t-1}^j)}{\sum_{k=1}^N q_{t-1|t}^\lambda(\xi_t^i, \xi_{t-1}^k)/q_{t-1}^\lambda(\xi_{t-1}^k)} . \tag{12}$$

Estimators (10) and (11) are *self-normalized importance sampling* (SNIS) estimators of equations (8) and (9). Equation (12) gives the shared importance weights of these estimator. It is worth noting that we cannot do direct Monte Carlo approximations of (8)-(9) by simulating samples from $q_{t-1|t}^\lambda(\xi_t^i, \cdot)$, as the functionals H_{t-1}^λ and G_{t-1}^λ would have no approximation on such samples. The use of importance sampling is thus

mandatory to update the approximations. It is known, though, that performance of such estimator strongly rely on the importance distribution. Section 5 proposes an efficient implementation to link the proposal distribution q_t^λ to the target $q_{t-1|t}^\lambda$. The self-normalization in (12) is motivated by computational considerations as it reduces the variance of the estimator in our simulation settings.

Estimators at final time. At final time T, simulate a N-sample $\{\xi_T^i\}_{1\leq i\leq N} \overset{i.i.d}{\sim} q_T^\lambda$, compute $\hat{H}_T^{\lambda,i}$ and $\hat{G}_T^{\lambda,i}$ using (10)-(11), and approximate the ELBO and its gradient with:

$$\widehat{\mathcal{L}}_T^\lambda = \frac{1}{N}\sum_{i=1}^N \left\{ \hat{H}_T^{\lambda,i} - \log q_T^\lambda(\xi_T^i) \right\}, \tag{13}$$

$$\widehat{\nabla}\mathcal{L}_T^\lambda = \frac{1}{N}\sum_{i=1}^N \left\{ \nabla \log q_T^\lambda(\xi_T^i) \cdot \hat{H}_T^{\lambda,i} + \hat{G}_T^{\lambda,i} \right\} . \tag{14}$$

Note that an appealing alternative to estimator (14) would be to perform auto-differentiation of estimator (13), as it would avoid to perform the recursion (11). However, in our setting, this approach is flawed because the approximation (10) is biased due the SNIS. Building gradients via auto-differentiation of a biased estimator can lead to catastrophic divergence, especially in the case of ELBO maximization which is based on an upper bound. Typically, the auto-differentiation will lead to the parameters that maximize the bias of the approximation.

4.2 Online computation

In the online computation framework, we aim at approximating the gradient at each time t, using the new observation, and updating the current parameter λ_t using this gradient.

Initialization. Starting from an initial guess λ_0, the initialization is the same as in Section 4.1. Then the first gradient is approximated with:

$$\widehat{\nabla}\mathcal{L}_0^{\lambda_0} = \frac{1}{N}\sum_{i=1}^N \nabla \log q_0^{\lambda_0}(\xi_0^i) \cdot \hat{H}_0^{\lambda_0,i} + \hat{G}_0^{\lambda_0,i} .$$

Then, λ_1 is set by updating λ_0 using this gradient, typically by setting $\lambda_1 = \lambda_0 + \gamma_0 \widehat{\nabla}\mathcal{L}_0^{\lambda_0}$ for some step size γ_0.

Recursive approximation of $H_t^{\lambda_t}$ and $G_t^{\lambda_t}$. At time t, simulate a N-sample $\{\xi_t^i\}_{1\leq i\leq N} \overset{i.i.d}{\sim} q_t^{\lambda_t}$. $H_t^{\lambda_t}(\xi_t^i)$ and $G_t^{\lambda_t}(\xi_t^i)$ are approximated respectively by:

$$\hat{H}_t^{\lambda_t,i} = \sum_{j=1}^N \bar{w}_{t-1|t}^{\lambda_t,i,j} \left(\hat{H}_{t-1}^{\lambda_{t-1},j} + \tilde{f}_t^{\lambda_t}(\xi_{t-1}^j, \xi_t^i) \right) , \tag{15}$$

$$\hat{G}_t^{\lambda_t,i} = \sum_{i=1}^N \bar{w}_{t-1|t}^{\lambda_t,i,j} \left\{ \hat{G}_{t-1}^{\lambda_{t-1},j} + \nabla \log q_{t-1|t}^{\lambda_t}(\xi_t^i, \xi_{t-1}^j) \left(\hat{H}_{t-1}^{\lambda_{t-1},j} + \tilde{f}_t^{\lambda_t}(\xi_{t-1}^j, \xi_t^i) \right) \right\} , \tag{16}$$

where

$$\bar{w}_{t-1|t}^{\lambda_t,i,j} = \frac{q_{t-1|t}^{\lambda_t}(\xi_t^i, \xi_{t-1}^j)/q_{t-1}^{\lambda_{t-1}}(\xi_{t-1}^j)}{\sum_{k=1}^N q_{t-1|t}^{\lambda_t}(\xi_t^i, \xi_{t-1}^k)/q_{t-1}^{\lambda_{t-1}}(\xi_{t-1}^k)} \ . \tag{17}$$

Equations (15)-(17) are almost identical to (10)-(12). The key difference is presence of quantities, on the right hand sides of (15)-(16), that were computed with λ_{t-1}, thus introducing new approximations. These approximations, which are commonly made in recursive maximum likelihood settings, are mandatory for practical implementation of online learning. Section 7 shows that these approximation, which would make theoretical study more complex, still lead to good results in practice.

5 Computational considerations

Defining backward kernels using forward potentials. Equations (12) and (17) suggest that the performance of the proposed algorithm would strongly rely on the definition of variational distributions, and on the link between $q_{t-1}^{\lambda_{t-1}}$ and $q_{t-1|t}^{\lambda_t}$. We therefore introduce additional structure in the variational family given by (4), using potential functions $\psi_t^\lambda : \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \to \mathbb{R}$ to link these distributions. Specifically, we prescribe that, for all $t \geq 1$,

$$q_{t-1|t}^{\lambda_t}(x_t, x_{t-1}) \propto q_{t-1}^{\lambda_{t-1}}(x_{t-1})\psi_t^{\lambda_t}(x_{t-1}, x_t) \ . \tag{18}$$

The functions $\psi_t^{\lambda_t}$ can be made arbitrarily complex, such that, for all t, the backward variational kernel $q_{t-1|t}^{\lambda_t}$ has arbitrarily complex dependencies w.r.t x_t.

Backward sampling. Computing the backward weights of (17) has the drawback of a $O(N^2)$ complexity due to the computation normalizing constant, which can be prohibitive when N is large (typically for high dimensional state spaces). One solution, introduced by Olsson et al. (2017) in the context of particle smoothing, is the *backward sampling*. At t, given ξ_t^i, sample independently M indexes $j_1, \ldots, j_M \in \{1, \ldots, N\}$ from the multinomial distribution with weights $\{\bar{w}_{t-1|t}^{\lambda,i,j}\}_{j \leq N}$, and replace (15) by $\sum_{k=1}^M (\hat{H}_{t-1}^{\lambda_{t-1},j_k} + \tilde{f}_t^{\lambda_t}(\xi_{t-1}^{j_k}, \xi_t^i))/M$. Olsson et al. (2017) showed that M can be much smaller than N (typically, $M = 2$), and then, this can lead to great improvement in complexity. Here, noting that $\bar{w}_{t-1|t}^{\lambda,i,j} \propto_j \psi_t^\lambda(\xi_{t-1}^j, \xi_t^i)$, the backward sampling step is done with an accept-reject mechanism without having to compute the normalizing constant of the weights. We refer the reader to Olsson et al. (2017), Gloaguen et al. (2022) and Dau and Chopin (2022) for alternative backward sampling approach.

Parameterization of variational distributions. In practice, employing the backward factorization under decomposition (18) in the online setting requires explicit parameterization, for all $t \geq 0$, of a distribution q_t^λ and a potential ψ_t^λ (not necessarily normalized) both of which depend on observations up to time t at most. For computational efficiency, each q_t^λ is chosen as a p.d.f. from a parametric distribution within the exponential family. This family is denoted as $\mathsf{P} = \{P_\eta\}_{\eta \in \mathcal{E}}$ where η is the corresponding natural parameter and $\mathcal{E}$ the parameter space for this family (typically be the family of Gaussian distributions defined on $\mathbb{R}^{d_x}$, in our experiments). Let η_t^λ be the parameter of q_t^λ. To ensure that distributions $q_{t-1|t}^\lambda$ belong to the same family, we impose that

$$\psi_t^\lambda(x_{t-1}, x_t) = \exp\left(\tilde{\eta}_t^\lambda(x_t) \cdot T(x_{t-1})\right) ,$$

where $\tilde{\eta}_t^\lambda(x_t) = \mathsf{MLP}^\lambda(x_t)$[3] and $T(x_{t-1})$ are a natural parameter and a sufficient statistic for the family P. Then, thanks to (18), $q_{t-1|t}^\lambda(x_t, \cdot)$ will be a p.d.f. from P with natural parameter $\eta_{t-1|t}^\lambda = \eta_{t-1}^\lambda + \tilde{\eta}_t^\lambda$. In this convenient setting, the backward kernels $q_{t-1|t}^\lambda$ can have arbitrarily complex dependencies on x_t while their p.d.f is analytically derived from the potentials. This eliminates the necessity to calculate normalizing constants (required, for instance, when computing (16)), all the while avoiding the reduction of these kernels to mere transformations or linearizations (e.g., linear-Gaussian kernels). For the parameters of q_t^λ two main approaches exist.

- *Amortized schemes* where the parameters of q_t^λ are updated using a parameterized mapping at every time t. This can be done using intermediate quantities $a_t \in \mathsf{A}$ (where A is a user-defined space), such that $a_t = \mathsf{MLP}^\lambda(a_{t-1}, y_t)$, and $\eta_t^\lambda = \mathsf{MLP}^\lambda(a_t)$. Initialization is performed using a random parameter a_{-1}, which may be fixed or learnt. Amortized schemes are computationally efficient (as knowledge from previous predictions is used to produce the current parameters), but require manually defining complex mappings. The recursions may be analytical (and not rely on a MLP), for example when $q_{0:t}^\lambda$ is the smoothing distribution of a linear-Gaussian, or when conjugacy is further leveraged to update the parameters $(\eta_t^\lambda)_{t\geq 0}$ (see Appendix C). Whatever the case, while the number of parameters becomes independent of t, the computational burden of the backpropagation through the states $(a_s)_{s\leq t}$ grows linearly with t. To prevent this, a solution is to truncate backpropagation, i.e. to assume that $(a_s^\lambda)_{s\leq t-\Delta}$ is independent of λ for some Δ.
- *Non-amortized schemes* where each q_t^λ and ψ_t^λ have their own parameter η_t and $\tilde{\eta}_t$, not related to those at time $t-1$. In this case, the optimized vector λ contains the parameters $(\eta_t)_{t\geq 0}$, and the number of parameters then grows linearly with t. This scheme modifies equation (9) (see Appendix B for details).

Variance reduction of the gradient estimator. Equations (7) and (9) involve computing expectations of *score functions*, i.e. expectations of the form $\mathbb{E}_{q^\lambda}\left[\nabla \log q^\lambda(X) \cdot f(X)\right]$, for some p.d.f. $q^\lambda(X)$. As studied in Mohamed et al. (2020), direct Monte Carlo estimation of the score-function yields high variance and should typically not be used without a proper variance reduction technique. The most straightforward approach is to design a *control variate*. Exploiting the fact that $\mathbb{E}_{q^\lambda}\left[\nabla \log q^\lambda(X)\right] = 0$, the target expectation is then equal to $\mathbb{E}_{q^\lambda}[\nabla \log q^\lambda(X)(f(X) - \mathbb{E}_{q^\lambda}[f(X)])]$. Fortunately, Monte Carlo estimates of $\mathbb{E}_{q_t^\lambda}[H_t^\lambda]$ and $\mathbb{E}_{q_{t-1|t}^\lambda}[H_{t-1}^\lambda(X_{t-1}) + \tilde{f}_t^\lambda(X_{t-1}, x_t)]$ are directly given as byproducts of algorithm of Section 4.2, namely by $N^{-1}\sum_{i=1}^N \hat{H}_t^{\lambda,i}$ and $\{\hat{H}_t^{\lambda,i}\}_{1\leq i\leq N}$. Our methodology comes built-in with variance reduction without the need to recompute additional quantities. As an alternative to this variance reduction technique, a natural consideration arises regarding the potential use of the reparametrization trick, as it often results in Monte Carlo estimators with lower variance compared to those obtained using the score function. However, implementing the reparametrization trick in this context necessitates expressing $\nabla \mathcal{L}_t^\lambda$ as an expectation with respect to a random variable $Z_{0:t}$ that does not depend on λ. Moreover, the recursive expression of this expectation at time $t+1$ must be derivable from its predecessor, which poses a non-trivial challenge. For example, in the classical case where $q_{0:t}$ is the p.d.f. of a multivariate Gaussian random variable with mean μ and variance Σ, and the expectation is taken w.r.t. $Z_{0:t} \sim \mathcal{N}(0, I_{d_x \times (t+1)})$, such a recursion is not feasible as the ELBO is no longer an additive functional when $X_{0:t}$ is replaced by $\mu + \Sigma^{\frac{1}{2}} Z_{0:t}$.

Appendix F provides the full algorithm using the backward sampling approach and the control variate estimate.

[3] MLP is used to denote a multi-layer perceptron.

6 Related work

Sequential Monte Carlo smoothing. The presented methodology draws from recent advances in sequential Monte Carlo (SMC methods) for SSMs, especially by i) proposing Monte Carlo approach for the approximation of conditional expectations under the backward kernels, and ii) introducing the structure (18) to link $q_{t-1}^{\lambda_{t-1}}$ and $q_{t-1|t}^{\lambda_t}$. As a major difference, though, we emphasize that here, all Monte Carlo samples are obtained by i.i.d. sampling, avoiding the *selection/mutation* steps of SMC, which lead to dependant samples over time. We refer the reader to Douc et al. (2014) (Section 11) for a presentation of the general concepts underpinning this class of algorithms, and Olsson et al. (2017), Gloaguen et al. (2022) and Dau and Chopin (2022) for recent works on this class of algorithms. It is worth noting here that the algorithm proposed here can actually be implemented for every expectations of *any additive functional* as defined in these references, and not only the ELBO or it's gradient.

Sequential variational inference. In sequential variational inference, early works tackling smoothing focus on offline scenarios with a forward factorization different from the one used here Johnson et al. (2016); Krishnan et al. (2017); Lin et al. (2018b); Marino et al. (2018). A drawback of forward factorizations is the incompatibility with online setting. Campbell et al. (2021) provides the first work which explores online variational additive smoothing for recursive computation of the ELBO and its gradients. The main difference with our approach is the way to approximate conditional expectations with respect to backward kernels. The authors rely on functional approximations of the conditional expectations at each timestep. The major drawback of this solution is that it requires running an inner optimization (to learn the best regression function as a proxy of the target conditional expectation) *at every iteration* t, which may be very costly, and requires an additional choice of the regression functions.

Recently, Chagneux et al. (2024) established the first theoretical result on error control in sequential variational inference, building upon the backward factorization proposed in Campbell et al. (2021). This result, coupled with the potential for online learning, serves as motivation for our algorithm.

A distinct line of research Marino et al. (2018); Zhao and Park (2020); Dowling et al. (2023) chooses to trade smoothing for filtering by targeting the marginal distributions $(\phi_t)_{t\leq 0}$ at each timestep with variational distributions q_t^λ that depend only on the observations up to t. A distinctive trait of these works is the additional assumption that, at t, q_t^λ is a good approximation of ϕ_t which can be used in further timesteps to learn the next approximations. In practice, this can hardly be verified especially under Gaussian variational families. That said, some of the ideas proposed in these works may be relevant for the smoothing problem, which we discuss more extensively as a perspective in Section 8.

7 Experiments

7.1 Linear-Gaussian HMM

We first evaluate our algorithm on a Linear-Gaussian HMM, which admits analytical smoothing distribution. such that optimal smoothing is available. We set:

$$X_0 \sim \mathcal{N}(\mu_0, Q_0),\ X_t = AX_{t-1} + \nu_t\ , t \geq 1\ ,$$
$$Y_t = BX_t + \epsilon_t\ , t \geq 0\ ,$$

where ν_t and ϵ_t are Gaussian centered noises with unknown variances Q and R, and μ_0, Q_0, A and B are unknown parameters with appropriate dimensions. In this case the Kalman smoothing recursions yield the

smoothing distribution, which is a Gaussian distribution. It is then possible to choose a variational model parameterized by λ which gets arbitrarily close to the true posterior by prescribing that each q_t^λ is the p.d.f. of a Gaussian distribution and $q_{t-1|t}^\lambda$ is a Gaussian kernel with linear dependance on x_t. In this case, the ELBO can also be computed recursively in closed-form because the conditional expectations $(H_t^\lambda)_{t\geq 0}$ are quadratic forms.

Learning in an offline setting. We first evaluate our algorithm on a sequence of fixed-length T to showcase that the proposed framework indeed enables to perform a gradient ascent algorithm. As an oracle baseline, we can compute the closed-form ELBO and its associated gradient via the reparameterization trick. As an alternative, an unbiased offline Monte Carlo estimate of the ELBO is obtained by drawing trajectoires using the backward dynamics given by the kernels $(q_{t-1|t}^\lambda)_{t\leq T}$ and using the reparameterization trick to obtain is gradient. We refer to this method as *backward MC*. To compare the methods at hand, we evaluate our ability to perform gradient-ascent to optimize the ELBO with respect to λ. For our recursive method, we choose $\Delta = 2$ to truncate the backpropagation, as we observe that $\Delta < 2$ prevents our method from converging altogether, while $\Delta > 2$ only improves convergence speed by a small margin. The experiment is run using 10 different parameters for the generative model, $d_x = d_y = 10$, $T = 500$ and $N = 2$ for the two methods involving Monte Carlo sampling. Figure 1 displays the evolution of the ELBO using both approaches. It shows the convergence of our score-based solution to the correct optimum given by the analytical computations. This is particularly appealing and notably demonstrates that our online gradient-estimation method may perform well using few samples. In practice, we observe that the variance reduction introduced in Section 5 is crucial in reaching such performance. Finally, despite the added cost of updating the intermediate quantities for the gradients at each timestep, we observe that the computational times of our solution is about 2.5 time slower than the optimization based on the oracle gradient computed analytically (in average, 89.5 ms per gradient step for our method, 37.1 for the oracle method). We want to emphasize here that we do not advocate for our method in the context of offline learning (hence for time series of small length).

Online learning from streaming data. In a second setting, we keep the same generative and variational models and dimensionality but generate a large sequence of $T = 500000$ observations and simulate the optimization of the joint ELBO $\mathcal{L}_T^\lambda$. The purpose here is to update the variational parameters online, i.e. by discarding already seen data at each step. In the context of stochastic optimization, since $\mathcal{L}_T^\lambda = \sum_{t=0}^T \mathcal{L}_t^\lambda - \mathcal{L}_{t-1}^\lambda$ (with the convention $\mathcal{L}_{-1}^\lambda = 0$), the right quantity to optimize becomes $\nabla\{\mathcal{L}_t^\lambda - \mathcal{L}_{t-1}^\lambda\}$. In practice we update λ_{t+1} by setting:

$$\lambda_{t+1} = \lambda_t + \gamma_{t+1}\left(\nabla\mathcal{L}_t^{\lambda_t} - \nabla\mathcal{L}_{t-1}^{\lambda_{t-1}}\right) , \tag{19}$$

in order to avoid recomputing the previous gradient[4]. In Figure 4, we plot the evolution of the ELBO through this optimization process.

7.2 Chaotic recurrent neural network.

We now consider the model used in Campbell et al. (2021), where $X_0 \sim \mathcal{N}(0, Q)$, and, for $t \geq 1$:

$$\begin{aligned} X_t =& X_{t-1} + \frac{\Delta}{\tau}\left(\gamma W \tanh\left(X_{t-1}\right) - X_{t-1}\right) + \eta_t , \\ Y_t =& X_t + \epsilon_t,\ t \geq 0 , \end{aligned}$$

[4]This approximation is typically made in traditional recursive maximum likelihood methods

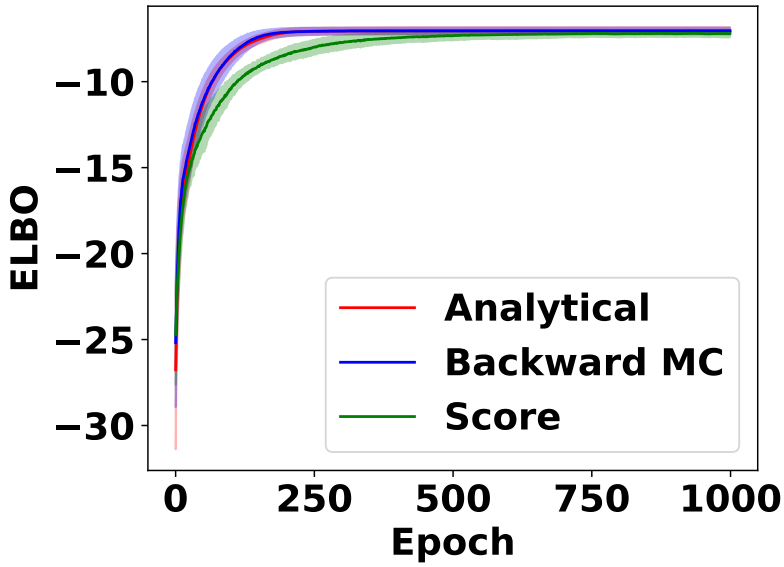

Figure 1: Evolution of $\mathcal{L}_T^\lambda/T$ computed with three different methods and with three different types of gradients estimates. Full lines: means of the 10 replicates. Shaded lines: standard deviations of the 10 replicates.

where $\eta \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, Q)$ is an isotropic Gaussian distribution and ϵ is a Student-t distribution, these two distributions being mutually independent. The hyperparameters chosen are the ones of Campbell et al. (2021) (see Appendix E.1).

Learning in an offline setting. Again, we start by evaluating the performance of our gradients against the backward trajectory sampling approach run on the same model, for a sequence of fixed length $T = 500$ with $d_x = d_y = 5$. For the variational family, we choose the setting presented in the section 5 where a linear-Gaussian kernels with parameters (A^λ, Q^λ) is used both to update the distributions $(q_t^\lambda)_{t\geq 0}$ and the backward kernels. We run gradient-ascent on λ by performing gradient steps using the quantity $\nabla\mathcal{L}_T^\lambda/T$ approximated via backward trajectory sampling and via our score-based method. As before, we use the same hyperparameters and optimization schemes for both methods. Table 1 reports the performance against the true states, averaged over dimensions, i.e. the quantity

$$\Delta_T^\lambda = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{d_x}\sum_{k=1}^{d_x}\left(x_t^{*(k)} - \mathbb{E}_{q_{0:T}^\lambda}\left[X_t^{(k)}\right]\right)^2}\,.$$

Recursive gradients in the offline setting. Even when we have access to an entire sequence of observations $y_{0:T}$, it can still be beneficial to use the recursive gradients approach for faster convergence. Indeed, when gradients are only available after processing the whole *batch*[5], the best we can do at optimization given

[5] i.e. the whole set of observation

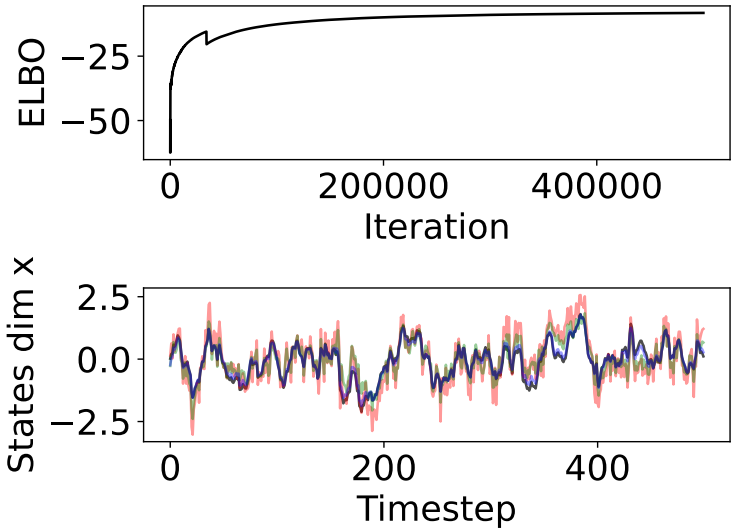

Figure 2: Top: evolution of $\frac{1}{t}\widehat{\mathcal{L}}_t^{\lambda}$ during the online learning. Bottom: evaluation on a test sequence of 500 observations from the same generative model (over 1 particular dimension, in black) for parameters obtained at iterations 1 (red), $10,000$ (green) and $500,000$ (blue).

Gradients	$\Delta_{T,p}^{\lambda}$ ($\times 10^{-2}$)	Avg. time
Score-based	13.5 ± 0.7 (**12.2**)	173 ms
Backward sampling	11.9 ± 0.4 (**11.4**)	17 ms

Table 1: RMSE between the true states x_t^* and the predicted marginal means $\mathbb{E}_{q_{0:T}^{\lambda}}[X_t]$ and average time per gradient step.

a fixed number of observations T is to update the parameter with

$$\lambda^{(k+1)} = \lambda^{(k)} + \gamma_{k+1} \nabla \mathcal{L}_T^{\lambda^{(k)}} , \tag{20}$$

where one such update is usually referred to as an "epoch", and $\lambda^{(k)}$ is the value of estimated parameter after k epochs. Using the recursive gradients, one may perform T intermediate updates within an epoch using

$$\lambda_{t+1}^{(k)} = \lambda_t^{(k)} + \gamma_{t+1}^{(k)} \left\{ \nabla \mathcal{L}_{t+1}^{\lambda_t^{(k)}} - \nabla \mathcal{L}_t^{\lambda_{t-1}^{(k)}} \right\} , \tag{21}$$

and

$$\lambda_0^{(k+1)} = \lambda_T^{(k)} ,$$

i.e. inside one epoch we optimize λ recursively on the observations. We compare the two options by optimizing on 10 different sequences of $T = 500$ observations, performing 10 epochs on each, using updates of the form (20) for the backward trajectory sampling approach and using updates of the form (21) with our score-based approach. Figure 3, displays the epoch-wise training curves for each method with $d_x = d_y = 5$, where we observe that optimizing with intermediate updates of (21) converges faster overall.

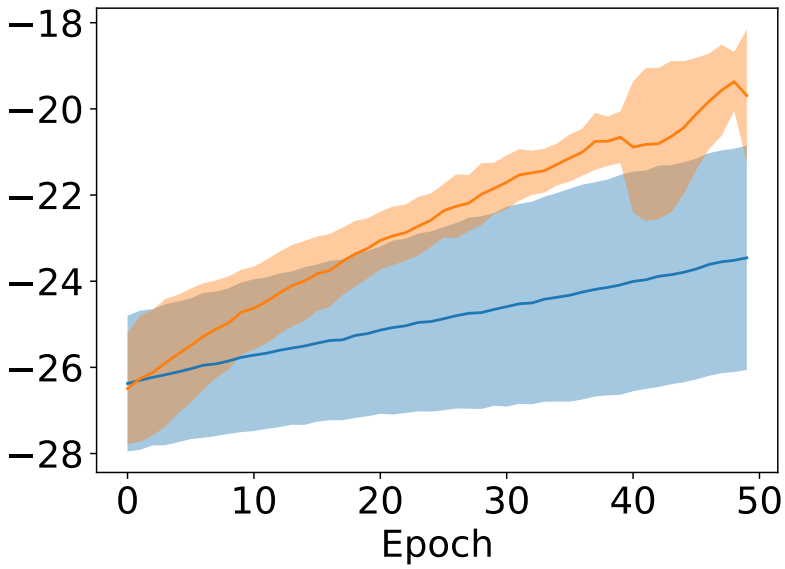

Figure 3: Evolution of $\mathcal{L}_T^\lambda/T$ for $\lambda = \lambda^{(k)}$ when performing with temporal updates inside an epoch (via recursive gradients) or without (as in Figure 1), $k \in \{0, \ldots, 50\}$. The full lines are the average over the 10 different runs, the shaded lines are the standard deviations across these runs.

Comparison with Campbell et al. (2021). The proposed method of this paper mainly differs from Campbell et al. (2021) in the way we approximate the backward statistics H_t^λ by using a recursive sampling approach rather than a regression approach. In order to compare the two approaches, we reproduce the experiment of appendix B.2 of Campbell et al. (2021), where the authors evaluate their ability to predict the hidden state one step backward (therefore performing 1-step smoothing).

Specifically, we aim at evaluating the quality of our approach to estimate the conditional law of X_{t-1} given $Y_{0:t}$ and of X_t given $Y_{0:t}$ by evaluating $\mathbb{E}_{q_{t-1:t}^\lambda}[X_{t-1}]$ and $\mathbb{E}_{q_t^\lambda}[X_t]$, where λ is learnt using the same setting as Campbell et al. (2021), i.e. a non-amortized scheme (see *Non-amortized schemes*, Section 5 and Appendix B). The details of implementation are given in Appendix E.1

Table 2, reports the average 1-step smoothing errors and filtering errors, i.e. the quantities

$$\kappa_T^{(1)} = \frac{1}{T-1}\sum_{t=1}^{T-1}\sqrt{\frac{1}{d_x}\sum_{k=1}^{d_x}\left(\mathbb{E}_{q_{t-1:t}^\lambda}\left[X_{t-1}^{(k)}\right] - x_{t-1}^{*(k)}\right)^2}$$

and

$$\kappa_T^{(2)} = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{d_x}\sum_{k=1}^{d_x}\left(\mathbb{E}_{q_t^\lambda}\left[X_t^{(k)}\right] - x_t^{*(k)}\right)^2}$$

when training our method in these two settings with $d_x = d_y = 5$. We also report the errors the computational times for the two methods averaged over 8 runs using 8 different generative models (hence 8 different sequences).

Method	Smooth.	Filt.	Time
Ours	8.9 (0.2)	10.3 (0.2)	1 ms
Campbell et al. (2021)	9.2 (0.2)	10.3 (0.2)	4.8 ms

Table 2: RMSE ($\times 10^{-2}$) between the true states x_t^* and $\widehat{\mathbb{E}}_{q_{t-1:t}^\lambda}[X_{t-1}]$ (column Smooth.) and $\widehat{\mathbb{E}}_{q_t^\lambda}[X_t]$ (Filt.) (with the standard error), and average time per gradient step.

Sequence	Smoothing RMSE	Filtering RMSE
Training	0.281	0.311
Eval	0.278 ($\pm$ 0.01)	0.305 ($\pm$ 0.014)

Table 3: Smoothing and filtering RMSE values for the training sequence and other sequences drawn from the same generative model, when λ is learnt online.

One can see that for comparable results, our approach based on Monte Carlo for estimating the backward expectation is about 5 times faster than the regression approach.

Online learning from streaming data. Finally, we evaluate the performance in the true online setting when training on a sequence of $T = 100,000$ observations using parameter updates of the form of (19). We choose $d_x = d_y = 10$ and $N = 100$ particles. To parameterize $q_{0:t}^\lambda$ we use the amortized model presented at the beginning of this section. Table 3 provides the smoothing and filtering RMSE against the true states at the end of optimization. We also show the inference performance on new sequences generated under the same generative model. The results clearly highlight that the fitted λ is relevant for new sequences, and illustrates the performance of our method in the amortized setting. This scheme is then appealing when one wants to train a single model on a long stream of incoming data, then re-use it for offline state inference on new sequences of arbitrary length.

8 Discussion

For future research, we identify two directions that could benefit from further investigation. First, we have only implemented the versions of our algorithm that rely on exponentially conjugated potentials, and as such more general parameterizations need to be evaluated. In practice, when the forward potentials $(\psi_t^\lambda)_{t\geq 0}$ are arbitrarily parameterized functions, it is expected that more flexible joint variational approximations can be obtained, and hence better results under complex nonlinear models.

Then, a more thorough analysis could be conducted to study the proper *stepwise* objective to optimize in situations where parameter updates are performed at every timestep. In this work, we have relied on the decomposition $\mathcal{L}_t^\lambda = \sum_{s=1}^t \mathcal{L}_s^\lambda - \mathcal{L}_{s-1}^\lambda$ as a justification to solve the optimization problem in λ via online stochastic gradient updates which maximize the ELBO over time. In contrast, works like Zhao and Park (2020); Dowling et al. (2023) develop an online variational optimization procedure by deriving lower bounds on the incremental likelihood. In practice, these solutions depart from the original ELBO and formulate intermediate optimization problems at each timestep by deriving a "single step" ELBO from the Kullback-Leibler divergence between ϕ_t and q_t^λ at each $t \geq 0$. As such, they do not target the smoothing distributions, and a joint variational distribution on the state sequence $X_{0:t}$ is only available in mean-field form $q_{0:t}^\lambda = \prod_{s=0}^t q_s^\lambda$, which does not capture dependencies between the states. In Dowling et al. (2023), however, the focus is put on learning the true model transitions, and they introduce a "hybrid" version of

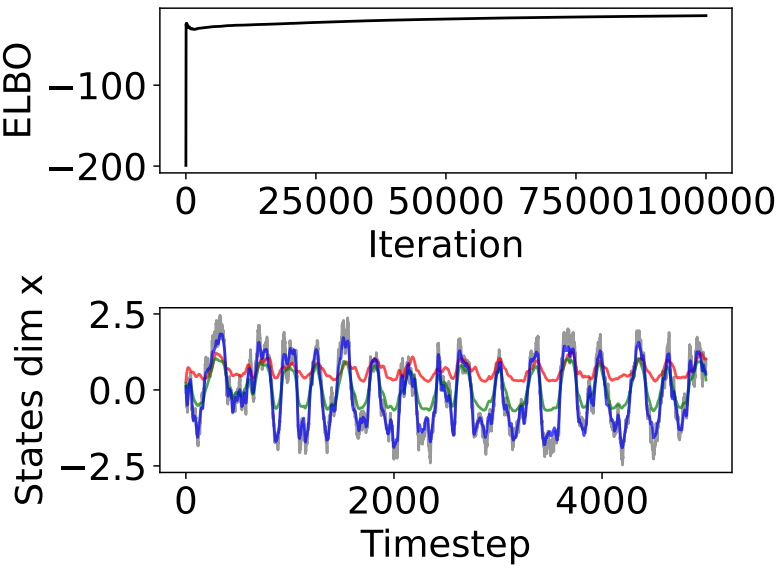

Figure 4: Top: evolution of $\frac{1}{t}\widehat{\mathcal{L}}_t^{\lambda}$ during the online learning. Bottom: evaluation on a test sequence of 5,000 observations from the same generative model (over 1 particular dimension, in grey) for parameters obtained at iterations 1 (red), $10,000$ (green) and $100,000$ (blue).

the *predictive* distribution of X_t given $Y_{0:t-1}$ defined as $\bar{q}_t^{\lambda}(x_t) = \mathbb{E}_{q_{t-1}^{\lambda}}[m_t(X_{t-1}, x_t)]$ for all $t \geq 0$, which is used to propagate the variational distributions. In the context of smoothing, we may similarly design variational backward kernels which rely on the true dynamics, i.e. for all $x_t \in \mathbb{R}^{d_x}$, $q_{t-1|t}^{\lambda}(x_t, x_{t-1}) \propto q_{t-1}^{\lambda}(x_{t-1}) m_t(x_{t-1}, x_t)$, in which case the normalizing constant is precisely $\bar{q}_t^{\lambda}(x_t)$ as defined above.

References

Bengtsson, T., Bickel, P., and Li, B. (2008). Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems. In *Probability and statistics: Essays in honor of David A. Freedman*, volume 2, pages 316–335. Institute of Mathematical Statistics.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Campbell, A., Shi, Y., Rainforth, T., and Doucet, A. (2021). Online variational filtering and parameter learning. *Advances in Neural Information Processing Systems*, 34.

Chagneux, M., Gassiat, É., Gloaguen, P., and Le Corff, S. (2024). Additive smoothing error in backward variational inference for general state-space models. *Journal of Machine Learning Research*.

Chopin, N., Papaspiliopoulos, O., et al. (2020). *An introduction to sequential Monte Carlo*, volume 4. Springer.

Dau, H.-D. and Chopin, N. (2022). On the complexity of backward smoothing algorithms. *arXiv preprint arXiv:2207.00976*.

Douc, R., Moulines, E., and Stoffer, D. (2014). *Nonlinear time series: theory, methods and applications with R examples*. CRC Press.

Dowling, M., Zhao, Y., and Park, I. M. (2023). Real-time variational method for learning neural trajectory and its dynamics. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Gloaguen, P., Corff, S. L., and Olsson, J. (2022). A pseudo-marginal sequential Monte Carlo online smoothing algorithm. *Bernoulli*, 28(4):2606 – 2633.

Johnson, M. J., Duvenaud, D. K., Wiltschko, A., Adams, R. P., and Datta, S. R. (2016). Composing graphical models with neural networks for structured representations and fast inference. *Advances in neural information processing systems (NeurIPS)*, 29.

Krishnan, R., Shalit, U., and Sontag, D. (2017). Structured inference networks for nonlinear state space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Lin, W., Khan, M. E., and Hubacher, N. (2018a). Variational message passing with structured inference networks. In *International Conference on Learning Representations*.

Lin, W., Khan, M. E., and Hubacher, N. (2018b). Variational message passing with structured inference networks. In *International Conference on Learning Representations*.

Marino, J., Cvitkovic, M., and Yue, Y. (2018). A general method for amortizing variational filtering. In *Advances in neural information processing systems (NeurIPS)*, volume 31.

Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. (2020). Monte carlo gradient estimation in machine learning. *The Journal of Machine Learning Research*, 21(1):5183–5244.

Olsson, J., Westerborn, J., et al. (2017). Efficient particle-based online smoothing in general hidden markov models: the PaRIS algorithm. *Bernoulli*, 23(3):1951–1996.

Zhao, Y. and Park, I. M. (2020). Variational online learning of neural dynamics. *Frontiers in computational neuroscience*, 14:71.

A Proof of proposition 3.1

We start by the definition of the ELBO:

$$
\begin{aligned}
\mathcal{L}_t^\lambda &= \mathbb{E}_{q_{0:t}^\lambda}\left[\log \frac{p_{0:t}(X_{0:t}, Y_{0:t})}{q_{0:t}^\lambda(X_{0:t})}\right] \\
&= \mathbb{E}_{q_{0:t}^\lambda}\left[\log \frac{\prod_{s=0}^t \ell_s(X_{s-1}, X_s, Y_s)}{q_t^\lambda(X_t)\prod_{s=1}^t q_{s-1|s}^\lambda(X_s, X_{s-1})}\right] && \text{by (2) and (4)} \\
&= \mathbb{E}_{q_{0:t}^\lambda}\left[\sum_{s=0}^t \log \frac{\ell_s(X_{s-1}, X_s, Y_s)}{q_{s-1|s}^\lambda(X_s, X_{s-1})} - \log q_t^\lambda(X_t)\right] && \text{Posing } q_{-1|0}^\lambda(x_0, x_{-1}) = 1 \\
&= \mathbb{E}_{q_{0:t}^\lambda}\left[\sum_{s=0}^t \tilde{f}_s^\lambda(X_{s-1}, X_s) - \log q_t^\lambda(X_t)\right] && \text{By the defintion (5)} \\
&= \mathbb{E}_{q_{0:t}^\lambda}\left[\sum_{s=0}^t \tilde{f}_s^\lambda(X_{s-1}, X_s)\right] - \mathbb{E}_{q_t^\lambda}\left[\log q_t^\lambda(X_t)\right] \\
&= \int \left\{\sum_{s=0}^t \tilde{f}_s^\lambda(x_{s-1}, x_s)\right\} q_t^\lambda(x_t) \prod_{s=1}^t q_{s-1|s}^\lambda(x_s, x_{s-1}) \mathrm{d}x_1 \dots \mathrm{d}x_t - \mathbb{E}_{q_t^\lambda}\left[\log q_t^\lambda(X_t)\right] \\
&= \mathbb{E}_{q_t^\lambda}\left[H_t^\lambda(X_t)\right] - \mathbb{E}_{q_t^\lambda}\left[\log q_t^\lambda(X_t)\right] \ ,
\end{aligned}
$$

where

$$
H_t^\lambda(x_t) := \int \left\{\sum_{s=0}^t \tilde{f}_s^\lambda(x_{s-1}, x_s)\right\} \prod_{s=1}^t q_{s-1|s}^\lambda(x_s, x_{s-1}) \mathrm{d}x_1 \dots \mathrm{d}x_{t-1} = \mathbb{E}_{q_{0:(t-1)|t}^\lambda}\left[f_{0:t}^\lambda(X_{0:t-1}, x_t)\right] \ ,
$$

with $f_{0:t}^\lambda$ the function defined in Section 3, and $q_{0:(t-1)|t}^\lambda(x_{0:t-1}, x_t) = \prod_{s=1}^t q_{s-1|s}^\lambda(x_s, x_{s-1})$ is a p.d.f., for every x_t, for $X_{0:t-1}$. Then, notice that:

$$
\begin{aligned}
H_t^\lambda(x_t) &= \int \left(\int \left\{\sum_{s=0}^{t-1} \tilde{f}_s^\lambda(x_{s-1}, x_s) + \tilde{f}_t^\lambda(x_{t-1}, x_t)\right\} \prod_{s=1}^{t-1} q_{s-1|s}^\lambda(x_s, x_{s-1}) \mathrm{d}x_1 \dots \mathrm{d}x_{t-2} \right) q_{t-1|t}^\lambda(x_t, x_{t-1}) \mathrm{d}x_{t-1} \\
&= \int \left(H_{t-1}^\lambda(x_{t-1}) + \tilde{f}_t^\lambda(x_{t-1}, x_t)\right) q_{t-1|t}^\lambda(x_t, x_{t-1}) \mathrm{d}x_{t-1} \\
&= \mathbb{E}_{q_{t-1|t}^\lambda}\left[H_{t-1}^\lambda(X_{t-1}) + \tilde{f}_t^\lambda(X_{t-1}, X_t)\right] \ .
\end{aligned}
$$

This development then shows the given expression of the ELBO and the recursion over H_t^λ. Consider the gradient of the ELBO, with respect to λ.

$$\begin{aligned}
\nabla \mathcal{L}_t^\lambda &= \nabla \mathbb{E}_{q_t^\lambda}\left[H_t^\lambda(X_t)\right] - \nabla \mathbb{E}_{q_t^\lambda}\left[\log q_t^\lambda(X_t)\right] \\
&= \nabla \int \left(H_t^\lambda(x_t) - \log q_t^\lambda(x_t)\right) q_t^\lambda(x_t) \mathrm{d}x_t \\
&= \int \left(\nabla H_t^\lambda(x_t) - \nabla \log q_t^\lambda(x_t)\right) q_t^\lambda(x_t) \mathrm{d}x_t + \int \left(H_t^\lambda(x_t) - \log q_t^\lambda(x_t)\right) \nabla q_t^\lambda(x_t) \mathrm{d}x_t \\
&= \mathbb{E}_{q_t^\lambda}\left[\nabla H_t^\lambda(X_t)\right] - \mathbb{E}_{q_t^\lambda}\left[\nabla \overset{=0}{\log} q_t^\lambda(X_t)\right] + \int \left(H_t^\lambda(x_t) - \log q_t^\lambda(x_t)\right) \nabla \left(\log q_t^\lambda(x_t)\right) q_t^\lambda(x_t) \mathrm{d}x_t \\
&= \mathbb{E}_{q_t^\lambda}\left[\nabla H_t^\lambda(X_t) + \left(H_t^\lambda(x_t) - \log q_t^\lambda(x_t)\right) \times \nabla \log q_t^\lambda(x_t)\right],
\end{aligned}$$

where $\mathbb{E}_{q_t^\lambda}\left[\nabla \log q_t^\lambda(X_t)\right] = 0$ is justified by the fact that $\mathbb{E}_{q_t^\lambda}\left[\nabla \log q_t^\lambda(X_t)\right] = \int \nabla q_t^\lambda(x_t) \mathrm{d}x_t = 0$.

Now, if we denote $G_t^\lambda(x_t) = \nabla H_t^\lambda(x_t)$.

$$\begin{aligned}
G_t^\lambda(x_t) &= \nabla \mathbb{E}_{q_{0:(t-1)|t}^\lambda}\left[f_{0:t}^\lambda(X_{0:t-1}, x_t)\right] \\
&= \mathbb{E}_{q_{0:(t-1)|t}^\lambda}\left[\left\{\nabla \log q_{0:(t-1)|t}^\lambda \times f_{0:t}^\lambda\right\}(X_{0:t-1}, x_t)\right] + \mathbb{E}_{q_{0:(t-1)|t}^\lambda}\left[\nabla f_{0:t}^\lambda(X_{0:t-1}, x_t)\right].
\end{aligned}$$

Here, it turns out that[6], by definition of $f_{0:t}^\lambda$:

$$\nabla f_{0:t}^\lambda(X_{0:t-1}, x_t) = -\nabla \log q_{0:(t-1)|t}^\lambda(X_{0:t-1}, x_t),$$

therefore, the expectation of this term is equal to 0, and:

$$G_t^\lambda(x_t) = \mathbb{E}_{q_{0:(t-1)|t}^\lambda}\left[\left\{\nabla \log q_{0:(t-1)|t}^\lambda \times f_{0:t}^\lambda\right\}(X_{0:t-1}, x_t)\right] .$$

Finally, it remains to show the wanted recursion for $G_t^\lambda(x_t)$:

$$G_t^\lambda(x_t) = \mathbb{E}_{q_{0:(t-1)|t}^\lambda}\left[\left(\nabla \log q_{0:(t-2)|t-1}^\lambda(X_{0:t-1}) + \nabla \log q_{t-1|t}^\lambda(X_{t-1}, x_t)\right)\left(f_{0:t-1}^\lambda(X_{0:t-1}) + \tilde{f}_t^\lambda(X_{t-1}, x_t)\right)\right] \tag{22}$$

$$= \mathbb{E}_{q_{t-1|t}^\lambda}\left[G_{t-1}^\lambda(X_{t-1})\right] \tag{23}$$

$$+ \mathbb{E}_{q_{t-1|t}^\lambda}\left[\nabla \log q_{t-1|t}^\lambda(X_{t-1}, x_t)\left(\mathbb{E}_{q_{0:(t-2)|t-1}^\lambda}\left[f_{0:t-1}^\lambda(X_{0:t-1})\right] + \tilde{f}_t^\lambda(X_{t-1}, x_t)\right)\right] \tag{24}$$

$$+ \mathbb{E}_{q_{t-1|t}^\lambda}\left[\tilde{f}_t^\lambda(X_{t-1}, x_t) \times \mathbb{E}_{q_{0:(t-2)|t-1}^\lambda}\left[\nabla \log q_{0:(t-2)|t-1}^\lambda(X_{0:t-1})\right]\right]. \tag{25}$$

On the inner expectation of (24) we recognize H_{t-1}^λ and (25) is again equal to 0. We therefore have the wanted result:

$$G_t^\lambda(x_t) = \mathbb{E}_{q_{t-1|t}^\lambda}\left[G_{t-1}^\lambda(X_{t-1}) + \nabla \log q_{t-1|t}^\lambda(X_{t-1}, x_t) \times \left(H_{t-1}^\lambda(X_{t-1}) + \tilde{f}_t^\lambda(X_{t-1}, x_t)\right)\right] .$$

[6]This would not be the case for another additive functional than the one of the ELBO.

B Details on the non amortized scheme

In the non amortized scheme, λ is a set of disjoint parameters, each of them corresponding to a specific time step. Namely $\lambda = \{\lambda^0, \dots, \lambda^t\}$. In the notations of the article, the estimate λ_{t-1} of λ after having processed observations $y_{0:t-1}$ is an estimate of the set $\{\lambda^0, \dots, \lambda^{t-1}\}$. Therefore, the gradient of the ELBO will only be with respect to λ^t. This affects the expression of the statistic $G_t^{\lambda t}$, and one can see in equations (22)-(25) that the term (23) will now be 0 when the gradient is taken w.r.t. λ^t. This means that this term no longer has to be propagated. Indeed, as we set $q_{t-1|t}^{\lambda^t}(x_t, x_{t-1}) \propto q_{t-1}^{\lambda_{t-1}}(x_{t-1})\psi_t^{\lambda^t}(x_{t-1}, x_t)$, the gradient of the ELBO w.r.t. λ^t will be

$$\begin{aligned}\nabla_{\lambda^t}\mathcal{L}_t^{\lambda^t} =& \mathbb{E}_{q_t^{\lambda^t}}\Big[\mathbb{E}_{q_{t-1|t}^{\lambda^t}}\Big[\nabla \log q_t^{\lambda}(X_t) \times \tilde{f}_t^{\lambda^t}(X_{t-1}, X_t) \\ &+ \nabla_{\lambda^t} \log q_{t-1|t}^{\lambda^t}(X_{t-1}, x_t) \times \Big(H_{t-1}^{\lambda_{t-1}}(X_{t-1}) + \tilde{f}_t^{\lambda^t}(X_{t-1}, X_t)\Big)\Big]\Big] \ .\end{aligned}$$

This gradient will be estimated using Monte Carlo in the same way as in the algorithm. In Campbell et al. (2021), the inner conditional expectation is estimated with a regression approach of Appendix D instead of importance sampling as in our approach.

C Using exponential conjugacy to process observations

To further reduce the computational cost, one may actually leverage exponential conjugacy both to update the parameters of the distributions $(q_t^{\lambda})_{t\geq 0}$ and to derive the parameters of the backward kernels $(q_{t-1|t}^{\lambda})_{t\geq 0}$. This is possible, for example, whenever P is the Gaussian family. Denoting $\vec{\eta}$ the function such that $\vec{\eta}^{\lambda}(x_{t-1})$ is the natural parameter vector of the linear-Gaussian kernel $\mathcal{N}(A^{\lambda}x_{t-1}, Q^{\lambda})$, and $\vec{\phi}^{\lambda}(x_{t-1}, x_t)$ the density of that latter kernel evaluated at x_t, then closed form updates can be derived for all parameters at any timestep when choosing

- $\eta_t^{\lambda} = \mathbb{E}_{q_{t-1}^{\lambda}}\left[\vec{\eta}^{\lambda}(X_{t-1})\right] + \bar{\eta}_{y_t}^{\lambda}$ where $\bar{\eta}_{y_t}^{\lambda} = \mathsf{MLP}^{\lambda}(y_t)$ is a natural parameter. Here, the expectation on the right hand side is analogous to the predict step in Kalman filtering, but assimilation of the observation can involve a complex nonlinear mapping, as originally proposed in Johnson et al. (2016).
- $\psi_t^{\lambda}(x_{t-1}, x_t) \propto \vec{\phi}^{\lambda}(x_{t-1}, x_t)$, in which case $\psi_t^{\lambda}(\cdot, x_t)$ is still conjugated to q_{t-1}^{λ} for any x_t, and the parameters of $q_{t-1|t}^{\lambda}$ can be derived as explained above simply by deriving the natural parameter which makes $\vec{\phi}^{\lambda}(\cdot, x_t)$ conjugated to q_{t-1}^{λ}.

In this setting the backward kernels are linear and Gaussian, and the only neural network involved in the variational approximation is used to assimilate the observations. Additionally, the parameters $(A^{\lambda}, Q^{\lambda})$ are shared between the updates for $(q_t^{\lambda})_{t\geq 0}$ and those for $(q_{t-1|t}^{\lambda})_{t\geq 1}$, which is analogous to the true model recursions where the forward transition kernels are involved both in the filtering recursions and in the definition of the backward kernels.

D Functional regression

Here, we recall the alternate option used in Campbell et al. (2021) to propagate approximation of the backward expectations. Denoting $\mathcal{F} = \left\{g : \mathbb{R}^p \to \mathbb{R}^{d_x}, \mathbb{E}_{q_t^{\lambda}}[\|g(X_t)\|_2] < \infty\right\}$, $H_t^{\lambda}(x)$ satisfies (by definition

of conditional expectation):

$$H_t^\lambda = \operatorname*{argmin}_{g \in \mathcal{F}} \mathbb{E}_{q_{t-1:t}^\lambda(X_{t-1}, X_t)} \|g(X_t) - [H_{t-1}^\lambda(X_{t-1}) + \tilde{f}_t^\lambda(X_{t-1}, X_t)]\|_2,$$

which provides a regressive objective for learning an approximation of H_t^λ. In practice authors restrict the minimization problem to a subset of $\mathcal{F}$, a parametric family of functions (typically, a neural network) parameterized by γ, belonging to $\Gamma \subset \mathbb{R}^{d_\gamma}$, and learn this by approximating the expectation with Monte Carlo method. Namely, the authors propose to estimate H_t^λ by $H_{\hat{\gamma}_t}^\lambda$ where

$$\hat{\gamma}_t = \operatorname*{argmin}_{\gamma \in \Gamma} \frac{1}{N} \sum_{k=1}^{N} \|H_\gamma^\lambda(\xi_t^k) - [H_{\hat{\gamma}_{t-1}}^\lambda(\xi_{t-1}^k) + \tilde{h}_t(\xi_{t-1}^k, \xi_t^k)]\|_2 \ , \tag{26}$$

where $\left\{(\xi_{t-1}^i, \xi_t^i)\right\}_{i=1,\dots,N}$ is an i.i.d. sample under the variational joint distribution of (X_{t-1}, X_t) which has density $q_{t-1:t}^\lambda = q_t^\lambda q_{t-1|t}^\lambda$. Upon convergence, $H_{\hat{\gamma}_t}^\lambda$ is then used in the successive recursions (in the type of (15)).

E Experiments settings

E.1 Appendix for section 7.2

Parameters for the chaotic RNN We choose the same hyperparameters than Campbell et al. (2021) with $\Delta = 0.001$, $\tau = 0.025$, $\gamma = 2.5$, 2 degrees of freedom and a scale of 0.1 for the Student-t distribution, and define $Q = \text{diag}(0.01)$.

Implementation settings for the comparison with Campbell et al. (2021) Each λ^t contains the parameter $\eta_t = (\mu_t, \Sigma_t)$ of the distribution $q_t^{\lambda^t} \sim \mathcal{N}(\mu_t, \Sigma_t)$ and the parameter $\tilde{\eta}_t$ of the function $\psi_t^{\lambda^t}$. For this latter function, we match the number of parameters of Campbell et al. (2021) by defining $\psi_t^{\lambda^t}(x, y) = \exp\left(\tilde{\eta}_t(y) \cdot T(x)\right)$ with $\tilde{\eta}_t(y) = (\tilde{\eta}_{t,1}(y), \tilde{\eta}_{t,2})$ where $y \mapsto \tilde{\eta}_{t,1}(y)$ is a multi-layer perceptron with 100 neurons from $\mathbb{R}^{d_x}$ to $\mathbb{R}^{d_x}$, and $\tilde{\eta}_{t,2}$ is a negative definite matrix. We follow the optimization schedules of Campbell et al. (2021) with $K = 500$ gradient steps at each timestep.

F Full algorithm with backward sampling and control variate

Algorithm 1 One iteration of the online gradient ascent algorithm (for $t \geq 1$) in the amortized scheme

Require:

- Previous statistics $\{\hat{G}_{t-1}^{\lambda_{t-1},i}, \hat{H}_{t-1}^{\lambda_{t-1},i}\}_{i=1}^N$;
- Previous samples $\{\xi_{t-1}^i\}_{i=1}^N$;
- Intermediate quantity a_{t-1} (see Section 5 *Parameterization of variational distributions*);
- Current parameter estimate λ_t;
- Step size γ_t for the gradient optimization procedure;
- New observation y_t.

Ensure: $\{\hat{G}_t^{\lambda_t,i}, \hat{H}_t^{\lambda_t,i}\}_{i=1}^N$, λ_{t+1}, a_t.

Compute $a_t = \mathsf{MLP}^{\lambda_t}(a_{t-1}, y_t)$.

Compute $\eta_t^\lambda = \mathsf{MLP}^{\lambda_t}(a_t)$, the parameters of $q_t^{\lambda_t}$ and sample $\{\xi_t^i\}_{i=1}^N$ i.i.d. with distribution $q_t^{\lambda_t}$.

for $i = 1$ to $i = N$ **do**

Compute $\tilde{\eta}_t^{\lambda_t,i} = \mathsf{MLP}^{\lambda_t}(\xi_t^i)$

for $j = 1$ to $j = M$ **do**

// Backward sampling step, M is the number of backward samples

Sample $(j_k)_{1\leq k\leq M} \overset{\text{i.i.d.}}{\sim} \mathsf{Cat}(\{\bar{w}_{t-1|t}^{\lambda_t,i,j}\}_{1\leq j\leq N})$ with the weights of (17).

end for

Compute *// Recall that each term* $\tilde{f}_t^{\lambda_t}$ *depends on* y_t.

$$\hat{H}_t^{\lambda_t,i} = \frac{1}{M}\sum_{k=1}^M \left\{\hat{H}_{t-1}^{\lambda_{t-1},j_k} + \tilde{f}_t^{\lambda_t}(\xi_{t-1}^{j_k}, \xi_t^i)\right\} ,$$

$$\hat{G}_t^{\lambda_t,i} = \frac{1}{M}\sum_{k=1}^M \left\{\hat{G}_{t-1}^{\lambda_{t-1},j_k} + \nabla \log q_{t-1|t}^{\lambda_t}(\xi_{t-1}^{j_k}, \xi_t^i)\left(\hat{H}_t^{\lambda_{t-1},j_k} + \tilde{f}_t^{\lambda_t}(\xi_{t-1}^{j_k}, \xi_t^i) - \hat{H}_t^{\lambda_t,i}\right)\right\} .$$

// Note the difference with (16) *and the inclusion of control variate* $\hat{H}_t^{\lambda,i}$ *for the computation of* $\hat{G}_t^{\lambda,i}$

// $\nabla q_{t-1|t}^{\lambda_t}(\xi_{t-1}^{j_k}, \xi_t^i)$ *is typically computed with automatic differentiation*

end for

$$\lambda_{t+1} = \lambda_t + \frac{\gamma_t}{N}\sum_{i=1}^N \left\{\hat{G}_t^{\lambda_t,i} + \nabla \log q_t^{\lambda_{t-1}}(\xi_t^i)\left(\hat{H}_t^{\lambda_t,i} - \frac{1}{N}\sum_{k=1}^N \hat{H}_t^{\lambda_t,k}\right)\right\} .$$

// Note the difference with (14) *and the inclusion of control variate* $\frac{1}{N}\sum_{i=1}^N \hat{H}_t^{\lambda,i}$ *for the computation of* $\widehat{\nabla}\mathcal{L}_t^\lambda$

// $\nabla \log q_t^{\lambda_{t-1}}(\xi_t^i)$ *is typically computed with automatic differentiation*