Auto-Encoding Sequential Monte Carlo

Anonymous authors
Paper under double-blind review

Abstract

We build on auto-encoding sequential Monte Carlo (AESMC):[1] a method for model and proposal learning based on maximizing the lower bound to the log marginal likelihood in a broad family of structured probabilistic models. Our approach relies on the efficiency of sequential Monte Carlo (SMC) for performing inference in structured probabilistic models and the flexibility of deep neural networks to model complex conditional probability distributions. We develop additional theoretical insights and introduce a new training procedure which improves both model and proposal learning. We demonstrate that our approach provides a fast, easy-to-implement and scalable means for simultaneous model learning and proposal adaptation in deep generative models.

1 Introduction

We build upon AESMC (Anon, 2017), a method for model learning that itself builds on variational auto-encoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014) and importance weighted auto-encoders (IWAEs) (Burda et al., 2016). AESMC is similarly based on maximizing a lower bound to the log marginal likelihood, but uses SMC (Doucet & Johansen, 2009) as the underlying marginal likelihood estimator instead of importance sampling (IS). For a very wide array of models, particularly those with sequential structure, SMC forms a substantially more powerful inference method than IS, typically returning lower variance estimates for the marginal likelihood. Consequently, by using SMC for its marginal likelihood estimation, AESMC often leads to improvements in model learning compared with VAEs and IWAEs. We provide experiments on structured time-series data that show AESMC based learning was able to learn useful representations of the latent space for both reconstruction and prediction more effectively than the IWAE counterpart.

AESMC was introduced in an earlier preprint (Anon, 2017) concurrently with the closely related methods of Maddison et al. (2017); Naesseth et al. (2017). In this work we take these ideas further by providing new theoretical insights for the resulting evidence lower bounds (ELBOs), extending these to explore the relative efficiency of different approaches to proposal learning, and using our results to develop a new and improved training procedure. In particular, we introduce a method for expressing the gap between an ELBO and the log marginal likelihood as a Kullback-Leibler (KL) divergence between two distributions on an extended sampling space. Doing so allows us to investigate the behavior of this family of algorithms when the objective is maximized perfectly, which occurs only if the KL divergence becomes zero. In the IWAE case, this implies that the proposal distributions are equal to the posterior distributions under the learned model. In the AESMC case, it has implications for both the proposal distributions and the intermediate set of targets that are learned. We demonstrate that, somewhat counter-intuitively, using lower variance estimates for the marginal likelihood can actually be harmful to proposal learning. In other words, we show that tighter is not necessarily better, and often substantially worse, in the context of variational bounds. Using these insights, we propose an adaptation to the AESMC algorithm, which we call *alternating* ELBOs, that uses different lower bounds for updating the model parameters and proposal parameters. We further show that this adaptation empirically leads to both improved model learning and proposal adaptation performance.

[1] This work builds upon an earlier preprint of the same name (Anon, 2017) along with the independent, simultaneously developed, closely related, work of Maddison et al. (2017) and Naesseth et al. (2017).

2 Background

2.1 State-Space Models

State-space models (SSMs) are probabilistic models over a set of latent variables $x_{1:T}$ and observed variables $y_{1:T}$. Given parameters θ, a SSM is characterized by an initial density $\mu_\theta(x_1)$, a series of transition densities $f_{t,\theta}(x_t|x_{1:t-1})$, and a series of emission densities $g_{t,\theta}(y_t|x_{1:t})$ with the joint density being $p_\theta(x_{1:T}, y_{1:T}) = \mu_\theta(x_1)\prod_{t=2}^T f_{t,\theta}(x_t|x_{1:t-1})\prod_{t=1}^T g_{t,\theta}(y_t|x_{1:t})$.

We are usually interested in approximating the posterior $p_\theta(x_{1:T}|y_{1:T})$ or the expectation of some test function φ under this posterior $I(\varphi) := \int \varphi(x_{1:T})p_\theta(x_{1:T}|y_{1:T})\,\mathrm{d}x_{1:T}$. We refer to these two tasks as inference. Inference in models which are non-linear, non-discrete, and non-Gaussian is difficult and one must resort to approximate methods, for which SMC has been shown to be one of the most powerful approaches (Doucet & Johansen, 2009).

We will consider model learning as a problem of maximizing the marginal likelihood $p_\theta(y_{1:T}) = \int p_\theta(x_{1:T}, y_{1:T})\,\mathrm{d}x_{1:T}$ in the family of models parameterized by θ.

2.2 Sequential Monte Carlo

SMC performs approximate inference on a sequence of target distributions $(\pi_t(x_{1:t}))_{t=1}^T$. In the context of SSMs, the target distributions are often taken to be $(p_\theta(x_{1:t}|y_{1:t}))_{t=1}^T$. Given a parameter ϕ and proposal distributions $q_{1,\phi}(x_1|y_1)$ and $(q_{t,\phi}(x_t|y_{1:t}, x_{1:t-1}))_{t=2}^T$ from which we can sample and whose densities we can evaluate, SMC is described in Algorithm 1.

Using the set of weighted particles $(\tilde{x}_{1:T}^k, w_T^k)_{k=1}^K$ at the last time step, we can approximate the posterior as $\sum_{k=1}^K \bar{w}_T^k \delta_{\tilde{x}_{1:T}^k}(x_{1:T})$ and the integral I_φ as $\sum_{k=1}^K \bar{w}_T^k \varphi(\tilde{x}_{1:T}^k)$, where $\bar{w}_T^k := w_T^k / \sum_j w_T^j$ is the normalized weight and δ_z is a Dirac measure centred on z. Furthermore, one can obtain an unbiased estimator of the marginal likelihood $p_\theta(y_{1:T})$ using the intermediate particle weights:

$$\hat{Z}_{\mathrm{SMC}} := \prod_{t=1}^T \left[\frac{1}{K}\sum_{k=1}^K w_t^k\right]. \tag{1}$$

Algorithm 1: Sequential Monte Carlo

Data: observed values $y_{1:T}$, model parameters θ, proposal parameters ϕ
Result: particles $(\tilde{x}_{1:T}^k)_{k=1}^K$, weights $(w_T^k)_{k=1}^K$, marginal likelihood estimate $\hat{Z}_{\mathrm{SMC}}$
begin
- Sample initial particle values $x_1^k \sim q_{1,\phi}(\cdot|y_1)$.
- Compute and normalize weights:

$$w_1^k = \frac{\mu_\theta(x_1^k)g_{1,\theta}(y_1|x_1^k)}{q_{1,\phi}(x_1^k|y_1)}, \qquad \bar{w}_1^k = \frac{w_1^k}{\sum_{\ell=1}^K w_1^\ell}.$$

- Initialize particle set: $\tilde{x}_1^k \leftarrow x_1^k$
- **for** $t = 2, 3, \ldots, T$ **do**
  - Sample ancestor index $a_{t-1}^k \sim \mathrm{Discrete}(\cdot|\bar{w}_{t-1}^1, \ldots, \bar{w}_{t-1}^K)$.
  - Sample particle value $x_t^k \sim q_{t,\phi}(\cdot|y_{1:t}, \tilde{x}_{1:t-1}^{a_{t-1}^k})$.
  - Update particle set $\tilde{x}_{1:t}^k \leftarrow (\tilde{x}_{1:t-1}^{a_{t-1}^k}, x_t^k)$.
  - Compute and normalize weights:

$$w_t^k = \frac{f_{t,\theta}(x_t^k|\tilde{x}_{1:t-1}^{a_{t-1}^k})g_{t,\theta}(y_t|\tilde{x}_{1:t}^k)}{q_{t,\phi}(x_t^k|y_{1:t}, \tilde{x}_{1:t-1}^{a_{t-1}^k})}, \qquad \bar{w}_t^k = \frac{w_t^k}{\sum_{\ell=1}^K w_t^\ell}.$$

- Compute marginal likelihood: $\hat{Z}_{\mathrm{SMC}} = \prod_{t=1}^T \frac{1}{K}\sum_{k=1}^K w_t^k$.

return *particles* $(\tilde{x}_{1:T}^k)_{k=1}^K$, *weights* $(w_T^k)_{k=1}^K$, *marginal likelihood estimate* $\hat{Z}_{SMC}$

The sequential nature of SMC and the resampling step are crucial in making SMC scalable to large T. The former makes it easier to design efficient proposal distributions as each step need only target the next set of variables x_t. The resampling step allows the algorithm to focus on promising particles in light of new observations, avoiding the exponential divergence between the weights of different samples that occurs for importance sampling as T increases. This can be demonstrated both empirically and theoretically (Del Moral, 2004, Chapter 9). We refer the reader to (Doucet & Johansen, 2009) for an in-depth treatment of SMC.

2.3 Importance Weighted Auto-Encoders

Given a dataset of observations $(y^{(n)})_{n=1}^N$, a generative model $p_\theta(x, y)$ and an inference network $q_\phi(x|y)$, IWAEs (Burda et al., 2016) maximize $\frac{1}{N}\sum_{n=1}^N \text{ELBO}_{\text{IS}}(\theta, \phi, y^{(n)})$ where, for a given observation y, the ELBO_{IS} (with K particles) is a lower bound on $\log p_\theta(y)$ by Jensen's inequality:

$$\text{ELBO}_{\text{IS}}(\theta, \phi, y) = \int Q_{\text{IS}}(x^{1:K}) \log \hat{Z}_{\text{IS}}(x^{1:K}) \, \mathrm{d}x^{1:K} \leq \log p_\theta(y), \text{ where} \tag{2}$$

$$Q_{\text{IS}}(x^{1:K}) = \prod_{k=1}^K q_\phi(x^k|y), \quad \hat{Z}_{\text{IS}}(x^{1:K}) = \frac{1}{K}\sum_{k=1}^K \frac{p_\theta(x^k, y)}{q_\phi(x^k|y)}. \tag{3}$$

Note that for $K = 1$ particle, this objective reduces to a VAE (Kingma & Welling, 2014; Rezende et al., 2014) objective to which we will refer to as

$$\text{ELBO}_{\text{VAE}}(\theta, \phi, y) = \int q_\phi(x|y)(\log p_\theta(x, y) - \log q_\phi(x|y)) \, \mathrm{d}x. \tag{4}$$

The IWAE optimization is performed using stochastic gradient ascent (SGA) where a sample from $\left(\prod_{k=1}^K q_\phi(x^k|y^{(n)})\right)$ is obtained using the reparameterization trick (Kingma & Welling, 2014) and the gradient $\frac{1}{N}\sum_{n=1}^N \nabla_{\theta,\phi} \log\left(\sum_{k=1}^K \frac{p_\theta(x^k, y^{(n)})}{q_\phi(x^k|y^{(n)})}\right)$ is used to perform an optimization step.

3 Auto-Encoding Sequential Monte Carlo

AESMC implements model learning, proposal adaptation, and inference amortization in a similar manner to the VAE and the IWAE: it uses SGA on an empirical average of the ELBO over observations. However, it varies in the form of this ELBO. In this section, we will introduce the AESMC ELBO, explain how gradients of it can be estimated, and discuss the implications of these changes.

3.1 Objective Function

Consider a family of SSMs $\{p_\theta(x_{1:T}, y_{1:T}) : \theta \in \Theta\}$ and a family of proposal distributions $\{q_\phi(x_{1:T}|y_{1:T}) = q_{1,\phi}(x_1|y_1)\prod_{t=2}^T q_{t,\phi}(x_t|x_{1:t-1}, y_{1:t}) : \phi \in \Phi\}$. AESMC uses an ELBO objective based on the SMC marginal likelihood estimator (1). In particular, for a given $y_{1:T}$, the objective is defined as

$$\text{ELBO}_{\text{SMC}}(\theta, \phi, y_{1:T}) := \int Q_{\text{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) \log \hat{Z}_{\text{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) \, \mathrm{d}x_{1:T}^{1:K} \, \mathrm{d}a_{1:T-1}^{1:K}, \tag{5}$$

where $\hat{Z}_{\text{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K})$ is defined in (1) and Q_{SMC} is the sampling distribution of SMC,

$$Q_{\text{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) = \left(\prod_{k=1}^K q_{1,\phi}(x_1^k)\right)\left(\prod_{t=2}^T\prod_{k=1}^K q_{t,\phi}(x_t^k|\tilde{x}_{1:t-1}^{a_{t-1}^k}) \cdot \text{Discrete}(a_{t-1}^k|w_{t-1}^{1:K})\right). \tag{6}$$

ELBO_{SMC} forms a lower bound to the log marginal likelihood $\log p_\theta(y_{1:T})$ due to Jensen's inequality. Hence, given a dataset $(y_{1:T}^{(n)})_{n=1}^N$, we can perform model parameter learning based on maximizing the lower bound of $\frac{1}{N}\sum_{n=1}^N \log p_\theta(y_{1:T}^{(n)})$:

$$\mathcal{J}(\theta, \phi) := \frac{1}{N}\sum_{n=1}^N \text{ELBO}_{\text{SMC}}(\theta, \phi, y_{1:T}^{(n)}). \tag{7}$$

For notational convenience, we will talk about optimizing ELBOs in the rest of this section. However, we note that the main intended use of AESMC is to amortize over datasets, for which we replace the ELBO gradient with an empirical sum as per (7).

3.2 Gradient Estimation

We describe a gradient estimator used for optimizing $\text{ELBO}_{\text{SMC}}(\theta, \phi, y_{1:T})$ using SGA. The SMC sampler in Algorithm 1 proceeds by sampling $x_1^{1:K}, a_1^{1:K}, x_2^{1:K}, \ldots$ sequentially from their respective distributions $\prod_{k=1}^K q_1(x_1^k)$, $\prod_{k=1}^K \text{Discrete}(a_1^k | w_1^{1:K})$, $\prod_{k=1}^K q_2(x_2^k | x_1^{a_1^k}), \ldots$ until the whole particle-weight trajectory $(x_{1:K}^{1:T}, a_{1:T-1}^{1:K})$ is sampled. From this trajectory, using equation (1), we can compute an estimation for the marginal likelihood.

Assuming that the sampling of latent variables $x_{1:T}^{1:K}$ is reparameterizable, we can make their sampling independent of (θ, ϕ). In particular, assume that there exists a set of auxiliary random variables $\epsilon_{1:T}^{1:K}$ where $\epsilon_t^k \sim s_t$ and a set of reparameterization functions r_t. We can simulate the SMC sampler as follows: sample $\epsilon_1^{1:K} \sim \prod_{k=1}^K s_1$ and set $x_1^k = r_1(\epsilon_1^k)$, then sample $a_1^{1:K}$ from $\prod_{k=1}^K \text{Discrete}(a_1^k | w_1^{1:K})$, then sample $\epsilon_2^{1:K} \sim \prod_{k=1}^K s_2$ and set $x_2^k = r_2(\epsilon_2^k, x_1^{a_1^k})$, until we obtain $(x_{1:K}^{1:T}, a_{1:T-1}^{1:K})$. We use this reparameterized sample of $(x_{1:K}^{1:T}, a_{1:T-1}^{1:K})$ to evaluate the gradient estimator $\nabla_{\theta,\phi} \log \hat{Z}_{\text{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K})$.

To account for the discrete choices of ancestor indices a_t^k we can additionally use the REINFORCE (Williams, 1992) trick, however in practice, we find that the additional term in the estimator has problematically high variance. We explore various other possible gradient estimators and empirical assessments of their variances in Appendix A. This exploration confirms that including the additional reinforce terms leads to problematically high variance, justifying our decision to omit them, despite introducing a small bias into the gradient estimates.

3.3 Bias & Implications on the Proposals

In this section, we express the gap between ELBOs and the log marginal likelihood as a KL divergence and study implications on the proposal distributions. We present a set of claims and propositions whose full proofs are in Appendix B. These give insight into the behavior of AESMC, compared with alternatives and show the advantages, and disadvantages, of using our different ELBO. This insight motivates Section 4 which proposes an algorithm for improving proposal learning.

Definition 1. *Given an* unnormalized target density $\tilde{P} : \mathcal{X} \to [0, \infty)$ *with* normalizing constant $Z_P > 0$, $P := \tilde{P}/Z_P$, *and a* proposal density $Q : \mathcal{X} \to [0, \infty)$, *the* ELBO

$$\text{ELBO} = \int Q(x) \log \frac{\tilde{P}(x)}{Q(x)} \, \mathrm{d}x, \tag{8}$$

is a lower bound on $\log Z_P$ *and satisfies*

$$\text{ELBO} = \log Z_P - \text{KL}\left(Q || P\right). \tag{9}$$

This is a standard identity used in variational inference (Wainwright et al., 2008) and VAEs. In the case of VAEs, applying Definition 1 with P being $p_\theta(x|y)$, $\tilde{P}$ being $p_\theta(x, y)$, Z_P being $p_\theta(y)$, and Q being $q_\phi(x|y)$, we directly can rewrite (4) as $\text{ELBO}_{\text{VAE}}(\theta, \phi, y) = \log p_\theta(y) - \text{KL}\left(q_\phi(x|y) || p_\theta(x|y)\right)$.

The key observation for expressing such a bound for general ELBOs such as ELBO_{IS} and ELBO_{SMC} is that the target density P and the proposal density Q need not directly correspond to $p_\theta(x|y)$ and $q_\phi(x|y)$. This allows us to view the underlying sampling distributions of the marginal likelihood Monte Carlo estimators such as Q_{IS} in (3) and Q_{SMC} in (6) as proposal distributions on an extended space $\mathcal{X}$. The following claim uses this observation to express the bound between a general ELBO and the log marginal likelihood as KL divergence from the extended space sampling distribution to a corresponding target distribution.

Claim 1. *Given a non-negative unbiased estimator $\hat{Z}_P(x) \geq 0$ of the normalizing constant Z_P where x is distributed from the proposal distribution $Q(x)$, the following holds:*

$$\text{ELBO} = \int Q(x) \log \hat{Z}_P(x) \, \mathrm{d}x = \log Z_P - \text{KL}\left(Q || P\right), \tag{10}$$

$$\textit{where} \quad P(x) = \frac{Q(x)\hat{Z}_P(x)}{Z_P} \tag{11}$$

is a normalized target density.

In the case of IWAEs, we can apply Claim 1 with Q and $\hat{Z}_P$ being Q_{IS} and $\hat{Z}_{\text{IS}}$ defined in (3) and Z_P being $p_\theta(y)$. This yields

$$\text{ELBO}_{\text{IS}}(\theta, \phi, y) = \log p_\theta(y) - \text{KL}\left(Q_{\text{IS}} || P_{\text{IS}}\right), \text{ where} \tag{12}$$

$$P_{\text{IS}}(x^{1:K}) = \frac{1}{K} \sum_{k=1}^{K} \left(q(x^1|y) \cdots q(x^{k-1}|y) p(x^k|y) q(x^{k+1}|y) \cdots q(x^K|y) \right). \tag{13}$$

Similarly, in the case of AESMC, we obtain

$$\text{ELBO}_{\text{SMC}}(\theta, \phi, y_{1:T}) = \log p_\theta(y_{1:T}) - \text{KL}\left(Q_{\text{SMC}} || P_{\text{SMC}}\right), \text{ where} \tag{14}$$

$$P_{\text{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) = Q_{\text{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) \hat{Z}_{\text{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) / p_\theta(y_{1:T}). \tag{15}$$

Having expressions for the target distribution P and the sampling distribution Q for a given ELBO allows us to investigate what happens when we maximize that ELBO, remembering that the KL term is strictly non-negative and zero if and only if $P = Q$. For the VAE and IWAE cases then, provided the proposal is sufficiently flexible, one can always perfectly maximize the ELBO by setting $p_\theta(x|y) = q_\phi(x|y)$ for all x. The reverse implication also holds: if $\text{ELBO}_{\text{VAE}} = \log Z_P$ then it must be the case that $p_\theta(x|y) = q_\phi(x|y)$. However, for AESMC, achieving $\text{ELBO} = \log Z_P$ is only possible when one also has sufficient flexibility to learn a particular series of intermediate target distributions, namely the marginals of the final target distribution. In other words, it is necessary to learn a particular factorization of the generative model, not just the correct individual proposals, to achieve $P = Q$. These observations are formalized in Propositions 1 and 2 below.

Proposition 1. *$Q_{IS}(x^{1:K}) = P_{IS}(x^{1:K})$ for all $x^{1:K}$ if and only if $q(x|y) = p(x|y)$ for all x.*

Proposition 2. *If $K > 1$, then $P_{SMC}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) = Q_{SMC}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K})$ for all $(x_{1:T}^{1:K}, a_{1:T-1}^{1:K})$ if and only if*

1. *$\pi_t(x_{1:t}) = \int p(x_{1:T}|y_{1:T}) \, \mathrm{d}x_{t+1:T} = p(x_{1:t}|y_{1:T})$ for all $x_{1:t}$ and $t = 1, \ldots, T$, and*
2. *$q_1(x_1|y_1) = p(x_1|y_{1:T})$ for all x_1 and $q_t(x_t|x_{1:t-1}, y_{1:t}) = p(x_{1:t}|y_{1:T}) / p(x_{1:t-1}|y_{1:T})$ for $t = 2, \ldots, T$ for all $x_{1:t}$,*

where $\pi_t(x_{1:t})$ are the intermediate targets used by SMC.

Proposition 2 has the consequence that if the family of generative models is such that condition 1 does not hold, we will not be able to make the bound tight $\text{ELBO}_{\text{SMC}} = Z_P$. This means that, except for a very small class of models, then, for most convenient parameterizations, it will be impossible to learn a perfect proposal that gives a tight bound, i.e. there will be no θ and ϕ such that the above conditions can be satisfied. However, it also means that ELBO_{SMC} encodes important additional information about the implications the factorization of the generative model has on the inference—the model depends only on the final target $\pi_T(x_{1:T}) = p_\theta(x_{1:T}|y_{1:T})$, but some choices of the intermediate targets $\pi_t(x_{1:t})$ will lead to much more efficient inference than others. Perhaps more importantly, SMC is usually a far more powerful inference algorithm than importance sampling and so the AESMC setup allows for more ambitious model learning problems to be effectively tackled than the VAE or IWAE. After all, even though it is well known in the SMC literature that, unlike for importance sampling, most problems have no perfect set of SMC proposals which will generate exact samples from the posterior (Doucet & Johansen, 2009), SMC is still gives superior performance on most problems with more than a few dimensions. These intuitions are backed up by our experiments that show that using ELBO_{SMC} regularly learns better models than using ELBO_{IS}.

4 Improving Proposal Learning

Given the implications from the previous section, we now ask whether optimizing ELBO_{IS} and ELBO_{SMC} actually improves the proposal distribution? In other words, does the optimization procedure make $q_\phi(x_{1:T}|y_{1:T})$ closer to $p_\theta(x_{1:T}|y_{1:T})$ and how does the number of particles K affect this? In the VAE case, we are directly optimizing $\text{KL}\left(q_\phi(x|y)||p_\theta(x|y)\right)$ for fixed model parameters and so it is straight forward to see that we will induce proposal learning. In the IWAE and AESMC cases, such optimization only minimizes $\text{KL}\left(Q_{\text{IS}}||P_{\text{IS}}\right)$ and $\text{KL}\left(Q_{\text{SMC}}||P_{\text{SMC}}\right)$ respectively, which does not directly imply that $\text{KL}\left(q_\phi(x|y)||p_\theta(x|y)\right)$ is small.

Counter-intuitively, it transpires that the tighter bounds implied by using a larger K is often harmful to proposal learning for both IWAE and AESMC. At a high-level, this is because an accurate estimate for $\hat{Z}_P$ can be achieved for a wide range of proposal parameters ϕ and so the magnitude of ∇_ϕ ELBO reduces as K increases. Typically, this shrinkage happens faster than increasing K reduces the standard deviation of the estimate and so the signal-to-noise ratio (SNR) for the gradient estimate actually decreases, even though it is a lower variance estimate. This effect is demonstrated in Figure 1 which shows a kernel density estimator for the distribution of the ELBO gradient estimate for different K and the model given in Section 5.2. Here we see that as we increase K, both the expected gradient estimate (i.e. true gradient) and standard deviation of the estimate decrease. However, the former decreases faster and so the SNR decreases. This is perhaps easiest to appreciate by noting that for $K > 10$, there is a roughly equal probability of the estimate being positive or negative, such that we are equally likely to increase or decrease the parameter value at the next SGA iteration, inevitably leading to poor performance. On the other hand, when $K = 1$, it is far more likely that the gradient estimate is positive than negative, and so there is clear drift to the gradient steps. We add to the empirical evidence for this behavior is Section 5. Note the critical difference for model learning is that ∇_θ ELBO does not, in general, decrease in magnitude as K increases. Note also that using a larger K should always give better performance at test time—the implication of our result is that it may be better to learn ϕ using a smaller K.

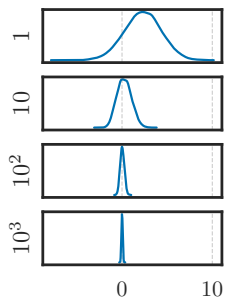


Figure 1: Density estimate of ∇_ϕ ELBO for different K

We can further demonstrate this result using an informal theoretical argument. Turner & Sahani (2011) previously showed that tighter bounds can be detrimental in some scenarios for variational inference. As well as its different context, our distinct analysis goes beyond their observations by showing that for the case of the IWAE, the gradient estimates for the proposal degrade arbitrarily for sufficiently large K. Our gradient estimate for the K particle IWAE is

$$I_K = \nabla_\phi \log\left(\frac{1}{K}\sum_{k=1}^{K}\frac{p_\theta(x^k, y)}{q_\phi(x^k|y)}\right), \tag{16}$$

where $I = \lim_{K\to\infty} I_K = 0$ because with infinite samples, the estimate is exact and thus independent of the proposal parameters. Now adapting the IWAE result of Rainforth et al. (2017) shows that

$$\mathbb{E}\left[I_K^2\right] = \mathbb{E}\left[\left(I_K - I\right)^2\right] \leq \frac{C_0^2\varsigma_1^4}{4K^2} + \frac{C_0^2\varsigma_1^4}{4K^2} + \frac{\kappa_0^2\varsigma_1^2}{K} + \frac{C_0\kappa_0\varsigma_1^3}{K^{3/2}} + O\left(\frac{1}{K^3}\right), \tag{17}$$

where C_0, κ_0 (K_0 in Rainforth et al. (2017)), and ς_1 are constants and we have set $N = 1$ in their formulation. Here the first $\frac{C_0^2\varsigma_1^4}{4K^2}$ and $O\left(\frac{1}{K^3}\right)$ are "bias terms", i.e. $\left(\mathbb{E}\left[I_K\right]\right)^2 = \left(\nabla_\phi \text{ELBO}\right)^2$, and the rest are variance terms. We can now define our SNR as follows

$$\text{SNR} = \frac{\nabla_\phi \text{ELBO}}{\sqrt{\text{Var}[I_K]}} \approx \sqrt{\frac{\frac{C_0^2\varsigma_1^4}{4K^2} + O\left(\frac{1}{K^3}\right)}{\frac{C_0^2\varsigma_1^4}{4K^2} + \frac{\kappa_0^2\varsigma_1^2}{K} + \frac{C_0\kappa_0\varsigma_1^3}{K^{3/2}}}} \approx \sqrt{\frac{C_0^2\varsigma_1^2}{4\kappa_0^2 K}} = O\left(\sqrt{\frac{1}{K}}\right), \tag{18}$$

where we have substituted in the bounds for the bias and variance from (17) and the approximations will, in general, become increasingly exact as K increases. We thus see that increasing K reduces the SNR and so a lower K is preferable for proposal learning.

4.1 Alternating ELBOs

To address these issues, we propose the alternating ELBOs (ALT) algorithm which updates (θ, ϕ) in a coordinate descent fashion using different ELBOs, and thus gradient estimates, for each. We pick a θ-optimizing pair and a ϕ-optimizing pair $(A_\theta, K_\theta), (A_\phi, K_\phi) \in \{\text{IS}, \text{SMC}\} \times \{1, 2, \dots\}$, corresponding to a inference type number of particles. In an optimization step, we obtain an estimator for $\nabla_\theta \, \text{ELBO}_{A_\theta}$ with K_θ particles and an estimator for $\nabla_\phi \, \text{ELBO}_{A_\phi}$ with K_ϕ particles which we call g_θ and g_ϕ respectively. We use g_θ to update the current θ and g_ϕ to update the current ϕ. The results from the previous sections suggest that using $A_\theta = \text{SMC}$ and $A_\phi = \text{IS}$ with a large K_θ and a small K_ϕ should perform better model and proposal learning than just fixing $(A_\theta, K_\theta) = (A_\phi, K_\phi)$ to (SMC, large) since using $A_\phi = \text{IS}$ with small K_ϕ helps learning ϕ and using $A_\theta = \text{SMC}$ with large K_θ helps learning θ. We experimentally find that this procedure in fact improves both model and proposal learning, leading to a new algorithm that improves the AESMC approach even further.

5 Experiments

We now present a series of experiments designed to answer the following questions: 1) Does reducing the gap by using more particles or a better inference procedure lead to an adverse effect on proposal learning? 2) Can AESMC, despite this effect, outperform IWAE? 3) Can we further improve the learned model and proposal by using our newly proposed algorithm ALT?

First we investigate a linear Gaussian state space model (LGSSM) for model learning and a latent variable model for proposal adaptation. This allows us to compare the learned parameters to the optimal ones. Doing so, we confirm our conclusions for this simple problem.

We then extend those results to more complex, high dimensional observation spaces that require models and proposals parameterized by neural networks. We do so by investigating the *Moving Agents* dataset, a set of partially occluded video sequences.

5.1 Linear Gaussian State Space Model

Given the following LGSSM

$$p(x_1) = \text{Normal}\left(x_1; 0, 1^2\right), \tag{19}$$

$$p(x_t|x_{t-1}) = \text{Normal}\left(x_t; \theta_1 x_{t-1}, 1^2\right), \qquad t = 2, \dots T, \tag{20}$$

$$p(y_t|x_t) = \text{Normal}\left(y_t; \theta_2 x_t, \sqrt{0.1}^2\right), \qquad t = 1, \dots, T, \tag{21}$$

we find that optimizing $\text{ELBO}_{\text{SMC}}(\theta, \phi, y_{1:T})$ w.r.t θ leads to better generative models than optimizing $\text{ELBO}_{\text{IS}}(\theta, \phi, y_{1:T})$. The same is true for using more particles.

To do so, we generate a sequence $y_{1:T}$ for $T = 200$ by sampling from the model with $\theta = (\theta_1, \theta_2) = (0.9, 1.0)$. We then optimize the different ELBOs w.r.t θ using the bootstrap proposal $q_1(x_1|y_1) = \mu_\theta(x_1)$ and $q_t(x_t|x_{1:t-1}, y_{1:t}) = f_{t,\theta}(x_t|x_{1:t-1})$. Because we use the bootstrap proposal, gradients w.r.t to θ are not backpropagated through q.

We use a fixed learning rate of 0.01 and optimize for 500 steps. Figure 2 shows that the convergence of both $\log p_\theta(y_{1:T})$ to $\max_\theta \log p_\theta(y_{1:T})$ and θ to $\text{argmax}_\theta \log p_\theta(y_{1:T})$ is faster when ELBO_{SMC} and more particles are used.

5.2 Proposal Learning

We now investigate how learning ϕ, i.e. the proposal, is affected by the the choice of ELBO and the number of particles.

Consider a simple, fixed generative model $p(\mu)p(x|\mu) = \text{Normal}(\mu; 0, 1^2)\text{Normal}(x; \mu, 1^2)$ where μ and x are the latent and observed variables respectively and a family of proposal distributions $q_\phi(\mu) = \text{Normal}(\mu; \mu_q, \sigma_q^2)$ parameterized by $\phi = (\mu_q, \log \sigma_q^2)$. For a fixed observation $x = 2.3$, we initialize $\phi = (0.01, 0.01)$ and optimize ELBO_{IS} with respect to ϕ. We investigate the quality of the learned parameter ϕ as we increase the number of particles K during training. Figure 3 (left) clearly demonstrates that the quality of ϕ compared to the analytic posterior decreases as we increase K.

Similar behavior is observed in Figure 3 (middle, right) where we optimize ELBO_{SMC} with respect to both θ and ϕ for the LGSSM described in Section 5.1. We see that using more

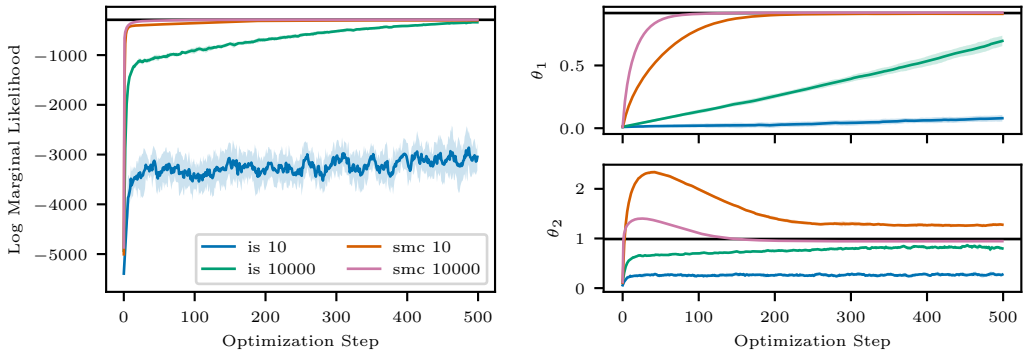


Figure 2: (Left) Log marginal likelihood analytically evaluated at every θ during optimization; the black line indicates $\max_\theta \log p_\theta(y_{1:T})$ obtained by the expectation maximization (EM) algorithm. (Right) learning of model parameters; the black line indicates $\text{argmax}_\theta \log p_\theta(y_{1:T})$ obtained by the EM algorithm.

particles helps model learning but makes proposal learning worse. Using our ALT algorithm alleviates this problem and at the same time makes model learning faster as it profits from a more accurate proposal distribution.

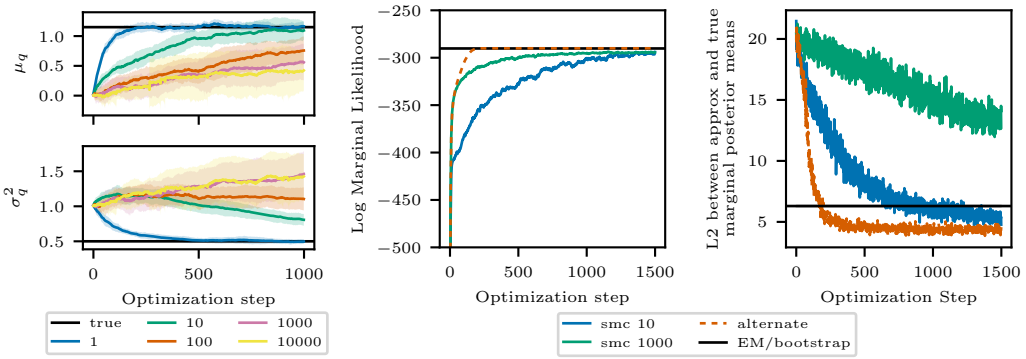


Figure 3: *(Left)* Optimizing ELBO_{IS} for the Gaussian unknown mean model with respect to ϕ results in worse ϕ as we increase number of particles K. *(Middle, right)* Optimizing ELBO_{SMC} with respect to (θ, ϕ) for LGSSM and using the ALT algorithm for updating (θ, ϕ) with $(A_\theta, K_\theta) = (\text{SMC}, 1000)$ and $(A_\phi, K_\phi) = (\text{IS}, 10)$. *Right* measures the quality of ϕ by showing $\sqrt{\sum_{t=1}^{T}(\mu_t^{\text{kalman}} - \mu_t^{\text{approx}})^2}$ where μ_t^{kalman} is the marginal mean obtained from the Kalman smoothing algorithm under the model with EM-optimized parameters and μ_t^{approx} is an marginal mean obtained from the set of 10 SMC particles with learned/bootstrap proposal.

5.3 Moving Agents

To show that our results are applicable to complex, high dimensional data we compare ALT, AESMC and IWAE on stochastic, partially observable video sequences. Figure 7 in the appendix shows an example of such a sequence.

The dataset consists of 5000 sequences of images $(y_{0:39}^{(i)})_1^{5000}$ of which 1000 are randomly held out as test set. Each sequence contains 40 images represented as a 3 dimensional array of size $1 \times 32 \times 32$. In each sequence there is one agent, represented as circle, whose starting position is sampled randomly along the top and bottom of the image. The dataset is inspired by (Ondruska & Posner, 2016), however with the crucial difference that the movement of the agent is *stochastic*. The agent performs a directed random walk through the image. At

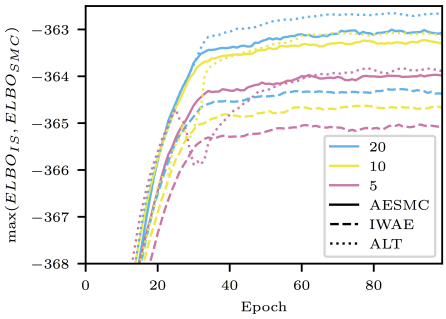
each timestep, it moves according to

$$\begin{aligned} y_{t+1} &\sim \mathcal{N}(y_{t+1}|y_t + 0.15, 0.02^2) \\ x_{t+1} &\sim \mathcal{N}(x_{t+1}|0, 0.02^2) \end{aligned} \tag{22}$$

where (x_t, y_t) are the coordinates in frame t in a unit square that is then projected onto 32×32 pixels. In addition to the stochasticity of the movement, half of the image is occluded, preventing the agent to be observed.

As generative model and proposal distribution we use a Variational Recurrent Neural Network (VRNN) (Chung et al., 2015). It extends recurrent neural networks (RNNs) by introducing a stochastic latent state x_t at each timestep t. Together with the observation y_t, this state conditions the deterministic transition of the RNN. By introducing this unobserved stochastic state, the VRNN is able to better model complex long range variability in stochastic sequences. Architecture and hyperparameter details are given in the appendix.

One can see in Figure 4 that models trained using the alternating training schedule ALT outperform AESMC which in turn outperform IWAE. Using more particles improves the ELBO, but more so for AESMC and ALT as the differences in estimators and training methods become more pronounced with higher particle numbers. In the appendix we inspect different learned generative models by using them for prediction, confirming the results presented here.



Particles	Method	Moving Agents
5	IWAE	-365.1
	AESMC	-364.0
	ALT	**-363.9**
10	IWAE	-364.6
	AESMC	-363.3
	ALT	**-363.1**
20	IWAE	-364.4
	AESMC	-363.06
	ALT	**-362.6**

Figure 4: *(Left)* Rolling mean over 5 epochs of $\max(\text{ELBO}_{\text{SMC}}, \text{ELBO}_{\text{IS}})$ on the test set. The color indicates the number of particles, the line style the used algorithm. *(Right)* The table shows the final $\max(\text{ELBO}_{\text{SMC}}, \text{ELBO}_{\text{IS}})$ for each learned model.

6 Conclusions

We have developed AESMC—a method for performing model learning using a new ELBO objective which is based on the SMC marginal likelihood estimator. This ELBO objective is optimized using SGA and the reparameterization trick. Our approach utilizes the efficiency of SMC in models with intermediate observations and hence is suitable for a highly structured models. We experimentally demonstrated that this objective has a tighter gap to the log marginal likelihood than the IWAE objective and that it works well on structured problems such as learning the generative model of moving balls which can be used for tasks such as reconstruction or prediction.

Additionally, in Claim 1, we provide a simple way to express the bias of objectives induced by log of marginal likelihood estimators as a KL divergence. In Propositions 1 and 2, we investigate the implications of these KLs being zero in the case of IWAE and AESMC. In the latter case, we find that we can achieve zero KL only if we are able to learn SMC intermediate target distributions corresponding to marginals of the target distribution. We then built on these results to show that *tighter in not necessarily better* when it comes to variational bounds and used this insight to develop a new method, alternating ELBOs, that address some of the issues of AESMC to improve proposal learning.

References

Anon. Auto-encoding sequential Monte Carlo. *arXiv preprint arXiv:1705.*****v1*, 2017.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *ICLR*, 2016.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pp. 2980–2988, 2015.

P Del Moral. Feynman-Kac formulae: genealogical and interacting particle systems with applications. *Probability and its applications*, 2004.

Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *ICLR*, 2014.

Chris J Maddison, Dieterich Lawson, George Tucker, Nicolas Heess, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Whye Teh. Filtering variational objectives. *arXiv preprint arXiv:1705.09279*, 2017.

Christian A Naesseth, Scott W Linderman, Rajesh Ranganath, and David M Blei. Variational sequential Monte Carlo. *arXiv preprint arXiv:1705.11140*, 2017.

Peter Ondruska and Ingmar Posner. Deep tracking: Seeing beyond seeing using recurrent neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Tom Rainforth, Robert Cornish, Hongseok Yang, Andrew Warrington, and Frank Wood. On the opportunities and pitfalls of nesting monte carlo estimators. *arXiv preprint arXiv:1709.06181*, 2017.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

Richard E Turner and Maneesh Sahani. Two problems with variational expectation maximisation for time-series models. *Bayesian Time series models*, pp. 115–138, 2011.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

A Gradients

The goal is to obtain an unbiased estimator for the gradient

$$\nabla_{\theta,\phi} \int Q_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) \log \hat{Z}_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) \, \mathrm{d}x_{1:T}^{1:K} \, \mathrm{d}a_{1:T-1}^{1:K}. \tag{23}$$

A.1 Full Reinforce

We express the required quantity as

$$\nabla_{\theta,\phi} \int Q_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) \log \hat{Z}_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) \, \mathrm{d}x_{1:T}^{1:K} \, \mathrm{d}a_{1:T-1}^{1:K} \tag{24}$$

$$= \int \nabla_{\theta,\phi} Q_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) \log \hat{Z}_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) + \tag{25}$$

$$Q_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) \nabla_{\theta,\phi} \log \hat{Z}_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) \, \mathrm{d}x_{1:T}^{1:K} \, \mathrm{d}a_{1:T-1}^{1:K} \tag{26}$$

$$= \int Q_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) \Big[\nabla_{\theta,\phi} \log Q_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) \log \hat{Z}_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) + \tag{27}$$

$$\nabla_{\theta,\phi} \log \hat{Z}_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) \Big] \, \mathrm{d}x_{1:T}^{1:K} \, \mathrm{d}a_{1:T-1}^{1:K}, \tag{28}$$

which we can estimate by sampling $(x_{1:T}^{1:K}, a_{1:T-1}^{1:K})$ directly from Q_{SMC} and evaluating $\Big[\nabla_{\theta,\phi} \log Q_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) \log \hat{Z}_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) + \nabla_{\theta,\phi} \log \hat{Z}_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K})\Big]$.

A.2 Reinforce & Reparameterization

We express the required quantity as

$$\nabla_{\theta,\phi} \int Q_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) \log \hat{Z}_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) \, \mathrm{d}x_{1:T}^{1:K} \, \mathrm{d}a_{1:T-1}^{1:K} \tag{29}$$

$$= \nabla_{\theta,\phi} \int \left(\prod_{k=1}^{K} q_1(x_1^k) \right) \left(\prod_{t=2}^{T} \prod_{k=1}^{K} q_t(x_t^k | x_{t-1}^{a_{t-1}^k}) \cdot \mathrm{Discrete}(a_{t-1}^k | w_{t-1}^{1:K}) \right)$$
$$\log \hat{Z}_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) \, \mathrm{d}x_{1:T}^{1:K} \, \mathrm{d}a_{1:T-1}^{1:K} \tag{30}$$

$$= \nabla_{\theta,\phi} \int \left(\prod_{k=1}^{K} s_1(\epsilon_1^k) \right) \left(\prod_{t=2}^{T} \prod_{k=1}^{K} s_t(\epsilon_t^k) \cdot \mathrm{Discrete}(a_{t-1}^k | w_{t-1}^{1:K}) \right)$$
$$\log \hat{Z}_{\mathrm{SMC}}(r(\epsilon_{1:T}^{1:K}), a_{1:T-1}^{1:K}) \, \mathrm{d}\epsilon_{1:T}^{1:K} \, \mathrm{d}a_{1:T-1}^{1:K} \tag{31}$$

$$= \int \left(\prod_{t=1}^{T} \prod_{k=1}^{K} s_t(\epsilon_t^k) \right) \Bigg[\nabla_{\theta,\phi} \prod_{t=2}^{T} \prod_{k=1}^{K} \mathrm{Discrete}(a_{t-1}^k | w_{t-1}^{1:K}) \log \hat{Z}_{\mathrm{SMC}}(r(\epsilon_{1:T}^{1:K}), a_{1:T-1}^{1:K}) +$$
$$\left(\prod_{t=2}^{T} \prod_{k=1}^{K} \mathrm{Discrete}(a_{t-1}^k | w_{t-1}^{1:K}) \right) \nabla_{\theta,\phi} \log \hat{Z}_{\mathrm{SMC}}(r(\epsilon_{1:T}^{1:K}), a_{1:T-1}^{1:K}) \Bigg] \, \mathrm{d}\epsilon_{1:T}^{1:K} \, \mathrm{d}a_{1:T-1}^{1:K} \tag{32}$$

$$= \int \left(\prod_{t=1}^{T} \prod_{k=1}^{K} s_t(\epsilon_t^k) \right) \left(\prod_{t=2}^{T} \prod_{k=1}^{K} \mathrm{Discrete}(a_{t-1}^k | w_{t-1}^{1:K}) \right) \cdot$$
$$\Bigg[\nabla_{\theta,\phi} \log \left(\prod_{t=2}^{T} \prod_{k=1}^{K} \mathrm{Discrete}(a_{t-1}^k | w_{t-1}^{1:K}) \right) \log \hat{Z}_{\mathrm{SMC}}(r(\epsilon_{1:T}^{1:K}), a_{1:T-1}^{1:K}) +$$
$$\nabla_{\theta,\phi} \log \hat{Z}_{\mathrm{SMC}}(r(\epsilon_{1:T}^{1:K}), a_{1:T-1}^{1:K}) \Bigg] \, \mathrm{d}\epsilon_{1:T}^{1:K} \, \mathrm{d}a_{1:T-1}^{1:K}, \tag{33}$$

where $r(\epsilon_{1:T}^{1:K})$ denotes a sample with identical distribution as $x_{1:T}^{1:K}$ obtained by passing the auxiliary samples $\epsilon_{1:T}^{1:K}$ through the reparameterization function. We can thus estimate the gradient by sampling $\epsilon_{1:T}^{1:K}$ from the auxiliary distribution, reparameterizing and evaluating $\Big[\nabla_{\theta,\phi} \log \Big(\prod_{t=2}^{T} \prod_{k=1}^{K} \mathrm{Discrete}(a_{t-1}^k | w_{t-1}^{1:K}) \Big) \log \hat{Z}_{\mathrm{SMC}}(r(\epsilon_{1:T}^{1:K}), a_{1:T-1}^{1:K}) + \nabla_{\theta,\phi} \log \hat{Z}_{\mathrm{SMC}}(r(\epsilon_{1:T}^{1:K}), a_{1:T-1}^{1:K})\Big]$.

In Figure 5, we demonstrate that the estimator in (33) has much higher variance if we include the first term.

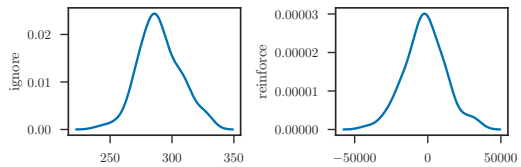


Figure 5: $T = 200$ model described in Section 5.1. Kernel density estimation (KDE) of $\nabla_{\theta_1} \text{ELBO}_{\text{SMC}}$ evaluated at $\theta_1 = 0.1$ with $K = 16$ using 100 samples.

B Proofs for Bias & Implications on the Proposals

Derivation of (9).

$$\text{ELBO} = \int Q(x) \log \frac{Z_P P(x)}{Q(x)} \, \mathrm{d}x \tag{34}$$

$$= \int Q(x) \log Z_P \, \mathrm{d}x - \int Q(x) \log \frac{Q(x)}{P(x)} \, \mathrm{d}x \tag{35}$$

$$= \log Z_P - \text{KL}\left(Q || P\right). \tag{36}$$

□

Proof of Claim 1. Since $\hat{Z}_P(x) \geq 0$, $Q(x) \geq 0$ and $\int Q(x)\hat{Z}_P(x) \, \mathrm{d}x = Z_P$, we can let the unnormalized target density in Definition 1 be $\tilde{P}(x) = Q(x)\hat{Z}_P(x)$. Hence, the normalized target density is $P(x) = Q(x)\hat{Z}_P(x)/Z_P$. Substituting these quantities into (8) and (9) yields the two equalities in (10). □

Proof of Proposition 1. ($\Longrightarrow$) Substituting for $Q_{\text{IS}}(x^{1:K}) = P_{\text{IS}}(x^{1:K})$, we obtain

$$\prod_{k=1}^{K} q(x^k|y) = \frac{1}{K} \sum_{k=1}^{K} \frac{\prod_{\ell=1}^{K} q(x^\ell|y)}{q(x^k|y)} p(x^k|y) = \frac{1}{K} \sum_{k=1}^{K} \left[q(x^1|y) \cdots q(x^{k-1}|y) p(x^k|y) q(x^{k+1}|y) \cdots q(x^K|y) \right]. \tag{37}$$

Integrating both sides with respect to $(x^2, \ldots, x^K)$ over the whole support (i.e. marginalize out everything except x^1), we obtain:

$$q(x^1|y) = \frac{1}{K} \left[p(x^1|y) + \sum_{k=2}^{K} q(x^1|y) \right]. \tag{38}$$

Rearranging gives us $q(x^1|y) = p(x^1|y)$ for all x^1.

($\Longleftarrow$) Substituting $p(x^k|y) = q(x^k|y)$, we obtain

$$P_{\text{IS}}(x^{1:K}) = \frac{1}{K} \sum_{k=1}^{K} \frac{Q_{\text{IS}}(x^{1:K})}{q(x^k|y)} p(x^k|y) \tag{39}$$

$$= \frac{1}{K} \sum_{k=1}^{K} Q_{\text{IS}}(x^{1:K}) \tag{40}$$

$$= Q_{\text{IS}}(x^{1:K}). \tag{41}$$

□

Proof of Proposition 2. We consider the general sequence of target distributions $\pi_t(x_{1:t})$ ($p_\theta(x_{1:t}|y_{1:t})$ in the case of SSMs), their unnormalized versions $\gamma_t(x_{1:t})$ ($p_\theta(x_{1:t}, y_{1:t})$ in the case of SSMs), their normalizing constants $Z_t = \int \gamma_t(x_{1:t}) \, \mathrm{d}x_{1:t}$ ($p_\theta(y_{1:t})$ in the case of SSMs), where $Z = Z_T = p(y_{1:T})$.

($\Longrightarrow$) It suffices to show that $\hat{Z}_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) = Z$ for all $(x_{1:T}^{1:K}, a_{1:T-1}^{1:K})$ implies 1 and 2 in Proposal 2 due to equation (11).

We first prove that $\hat{Z}_{\mathrm{SMC}}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) = Z$ for all $(x_{1:T}^{1:K}, a_{1:T-1}^{1:K})$ implies that the weights

$$w_1(x_1) := \frac{\gamma_1(x_1)}{q_1(x_1)} \tag{42}$$

$$w_t(x_{1:t}) := \frac{\gamma_t(x_{1:t})}{\gamma_{t-1}(x_{1:t-1})q_t(x_t|x_{1:t-1})} \qquad \text{for } t = 2, \dots, T \tag{43}$$

are constant with respect to $x_{1:t}$.

Pick $t \in \{1, \dots, T\}$ and distinct $k, \ell \in \{1, \dots, K\}$. Also, pick $x_{1:t}$ and $x'_{1:t}$. Now, consider two sets of particle sets $(\bar{x}_{1:T}^{1:K}, \bar{a}_{1:T-1}^{1:K})$ and $(\tilde{x}_{1:T}^{1:K}, \tilde{a}_{1:T-1}^{1:K})$, illustrated in Figure 6, such that

$$\bar{x}_\tau^\kappa = \begin{cases} x'_\tau & \text{if } \kappa = \ell \text{ and } \tau < t \\ x'_\tau & \text{if } (\kappa, \tau) = (k, t) \\ x_\tau & \text{if } \kappa = k \text{ and } \tau < t \\ x_\tau^\kappa & \text{otherwise} \end{cases} \qquad \text{for } \tau = 1, \dots, T, \kappa = 1, \dots, K, \tag{44}$$

$$\bar{a}_\tau^\kappa = \begin{cases} \ell & \text{if } (\kappa, \tau) = (k, t-1) \text{ or } (k, t) \\ \kappa & \text{otherwise} \end{cases} \qquad \text{for } \tau = 1, \dots, T-1, \kappa = 1, \dots, K, \tag{45}$$

$$\tilde{x}_\tau^\kappa = \begin{cases} x'_\tau & \text{if } \kappa = \ell \text{ and } \tau < t \\ x_\tau & \text{if } (\kappa, \tau) = (k, t) \\ x_\tau & \text{if } \kappa = k \text{ and } \tau < t \\ x_\tau^\kappa & \text{otherwise} \end{cases} \qquad \text{for } \tau = 1, \dots, T, \kappa = 1, \dots, K, \tag{46}$$

$$\tilde{a}_\tau^\kappa = \begin{cases} \ell & \text{if } (\kappa, \tau) = (k, t) \\ \kappa & \text{otherwise} \end{cases} \qquad \text{for } \tau = 1, \dots, T-1, \kappa = 1, \dots, K. \tag{47}$$

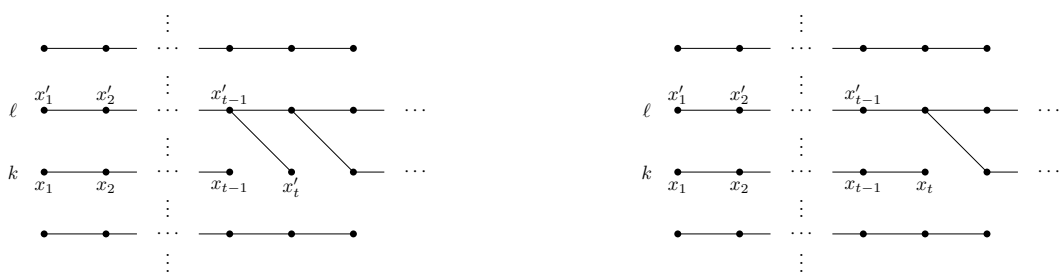


Figure 6: (Left) particle set $(\bar{x}_{1:T}^{1:K}, \bar{a}_{1:T-1}^{1:K})$ and (right) particle set $(\tilde{x}_{1:T}^{1:K}, \tilde{a}_{1:T-1}^{1:K})$. Lines indicate ancestor indices.

The weights $\bar{w}_\tau^\kappa$ and $\tilde{w}_\tau^\kappa$ for the respective particle sets are identical except when $(\tau, \kappa) = (t, k)$ where

$$\bar{w}_t^k = w_t(x'_{1:t}), \tag{48}$$

$$\tilde{w}_t^k = w_t(x_{1:t}). \tag{49}$$

Since $\hat{Z}(\bar{x}_{1:T}^{1:K}, \bar{a}_{1:T-1}^{1:K}) = \hat{Z}(\tilde{x}_{1:T}^{1:K}, \tilde{a}_{1:T-1}^{1:K})$, we have $w_t(x'_{1:t}) = w_t(x_{1:t})$. We have proven that $w_t(x_{1:t})$ is constant with respect to $x_{1:t}$ for all $t = 1, \dots, T$.

Now, for $x_{1:t}$, consider the proposal by rearranging (42) and (43)

$$q_1(x_1) = \frac{\gamma_1(x_1)}{w_1} \tag{50}$$

$$q_t(x_t|x_{1:t-1}) = \frac{\gamma_t(x_{1:t})}{\gamma_{t-1}(x_{1:t-1})w_t} \qquad \text{for } t = 2, \dots, T, \tag{51}$$

where $w_t := w_t(x_{1:t})$ is constant. For this to be a normalized density with respect to x_t, we must have

$$w_1 = \int \gamma_1(x_1)\,\mathrm{d}x_1 = Z_1, \tag{52}$$

and for $t = 2, \dots, T$:

$$\begin{aligned} w_t &= \int \frac{\gamma_t(x_{1:t})}{\gamma_{t-1}(x_{1:t-1})}\,\mathrm{d}x_t && (53)\\ &= \frac{\int \gamma_t(x_{1:t})\,\mathrm{d}x_t}{\gamma_{t-1}(x_{1:t-1})} && (54)\\ &= \frac{Z_t}{Z_{t-1}} \cdot \frac{\int \pi_t(x_{1:t})\,\mathrm{d}x_t}{\pi_{t-1}(x_{1:t-1})}. && (55) \end{aligned}$$

Since $\int \pi_{t+1}(x_{1:t+1})\,\mathrm{d}x_{t+1}$ and $\pi_t(x_{1:t})$ are both normalized densities, we must have $\pi_t(x_{1:t}) = \int \pi_{t+1}(x_{1:t+1})\,\mathrm{d}x_{t+1}$ for all $t = 1, \dots, T-1$ for all $x_{1:t}$. For a given $t \in \{1, \dots, T-1\}$ and $x_{1:t}$, applying this repeatedly yields

$$\pi_t(x_{1:t}) = \int \pi_{t+1}(x_{1:t+1})\,\mathrm{d}x_{t+1} = \int\int \pi_{t+2}(x_{1:t+2})\,\mathrm{d}x_{t+2}\,\mathrm{d}x_{t+1} = \cdots = \int \pi_T(x_{1:T})\,\mathrm{d}x_{t+1:T}. \tag{56}$$

We also have

$$\begin{aligned} w_1(x_1) &= Z_1, && && (57)\\ w_t(x_{1:t}) &= \frac{Z_t}{Z_{t-1}}, && t = 2, \dots, T, && (58)\\ q_1(x_1) &= \pi_1(x_1) = \pi_T(x_1), && && (59)\\ q_t(x_t|x_{1:t-1}) &= \frac{\pi_t(x_{1:t})}{\pi_{t-1}(x_{1:t-1})} = \frac{\pi_T(x_{1:t})}{\pi_T(x_{1:t-1})}, && t = 2, \dots, T. && (60) \end{aligned}$$

($\Longleftarrow$) We substitute identities in 1 and 2 of Proposal 2 back to the expression of $\hat{Z}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K})$ to obtain $\hat{Z}(x_{1:T}^{1:K}, a_{1:T-1}^{1:K}) = Z$. □

C VRNN

In the following we give the details of our VRNN architecture. The generative model is given by:

$$p(x_{1:T}, h_{0:T}, y_{1:T}) = p(h_0) \prod_t p(x_t|h_{t-1}) p(y_t|h_{t-1}, x_t) p(h_t|h_{t-1}, x_t, y_t) \tag{61}$$

where

$$\begin{aligned} p(h_0) &= \mathcal{N}(h_0; 0, \mathbb{1})\\ p(x_t|h_{t-1}) &= \mathcal{N}(x_t; \mu_\theta^x(h_{t-1}), \sigma_\theta^x(h_{t-1})^2)\\ p(y_t|h_{t-1}, x_t) &= \text{Bernoulli}(y_t; \mu_\theta^y(\varphi_\theta^x(x_t), h_{t-1}))\\ p(h_t|h_{t-1}, x_t, y_t) &= \delta_{f(h_{t-1}, \varphi_\theta^x(x_t), \varphi_\theta^y(y_t))}(h_t) \end{aligned} \tag{62}$$

and the proposal distribution is given by

$$p(x_t|y_t, h_{t-1}) = \mathcal{N}(x_t; \mu_\phi^p(\varphi_\phi^y(y_t), h_{t-1}), \sigma_\phi^{p\,2}(\varphi_\phi^y(y_t), h_{t-1})) \tag{63}$$

The functions μ_θ^x and σ_θ^x are computed by networks with two fully connected layers of size 128 whose first layer is shared. φ_θ^x is one fully connected layer of size 128.

For visual input, the encoding φ_θ^y is a convolutional network with conv-4x4-2-1-32, conv-4x4-2-1-64, conv-4x4-2-1-128 where conv-wxh-s-p-n denotes a convolutional network with n filters of size $w \times h$, stride s, padding p. Between convolutions we use leaky ReLUs with

slope 0.2 as nonlinearity and batch norms. The decoding μ_θ^y uses transposed convolutions of the same dimensions but in reversed order, however with stride $s = 1$ and padding $p = 0$ for the first layer.

A Gated Recurrent Unit (GRU) is used as RNN and if not stated otherwise ReLUs are used in between fully connected layers.

For the proposal distribution, the functions μ_ϕ^p and σ_ϕ^p are neural networks with three fully connected layers of size 128 that are sharing the first two layers. Sigmoid and softplus functions are used where values in $(0, 1)$ or $\mathbb{R}^+$ are required. We use a minibatch size of 40.

For the moving agents dataset we use SGA with a learning rate of $1e - 5$.

A specific feature of the VRNN architecture is that the proposal and the generative model share the component $\varphi_{\phi,\theta}^y$. Consequently, we set $\phi = \theta$ for the parameters belonging to this module and train it using gradients for both θ and ϕ.

D Experiments

D.1 Moving Agents

In figure 7 we investigate the quality of the generative model by comparing visual predictions. We do so for models learned by IWAE *(top)* and AESMC *(bottom)*. The models were learned using ten particles but for easier visualization we only predict using five particles.

The first row in each graphic shows the ground truth. The second row shows the averaged predictions of all five particles. The next five rows show the predictions made by each particle individually.

The observations (i.e. the top row) up to $t = 19$ are shown to the model. Up to this timestep the latent values $x_{0:19}$ are drawn from the proposal distribution $q(x_t|y_t, h_{t-1})$. From $t = 20$ onwards the latent values $x_{20:37}$ are drawn from the generative model $p(x_t|x_{t-1})$. Consequently, the model predicts the partially occluded, stochastic movement over 17 timesteps into the future.

We note that most particles predict a viable future trajectory. However, the model learned by IWAE is not as consistent in the quality of its predictions, often 'forgetting' the particle. This does not happen in every predicted sequence but the behavior shown here is very typical. Models learned by AESMC are much more consistent in the quality of their predictions.

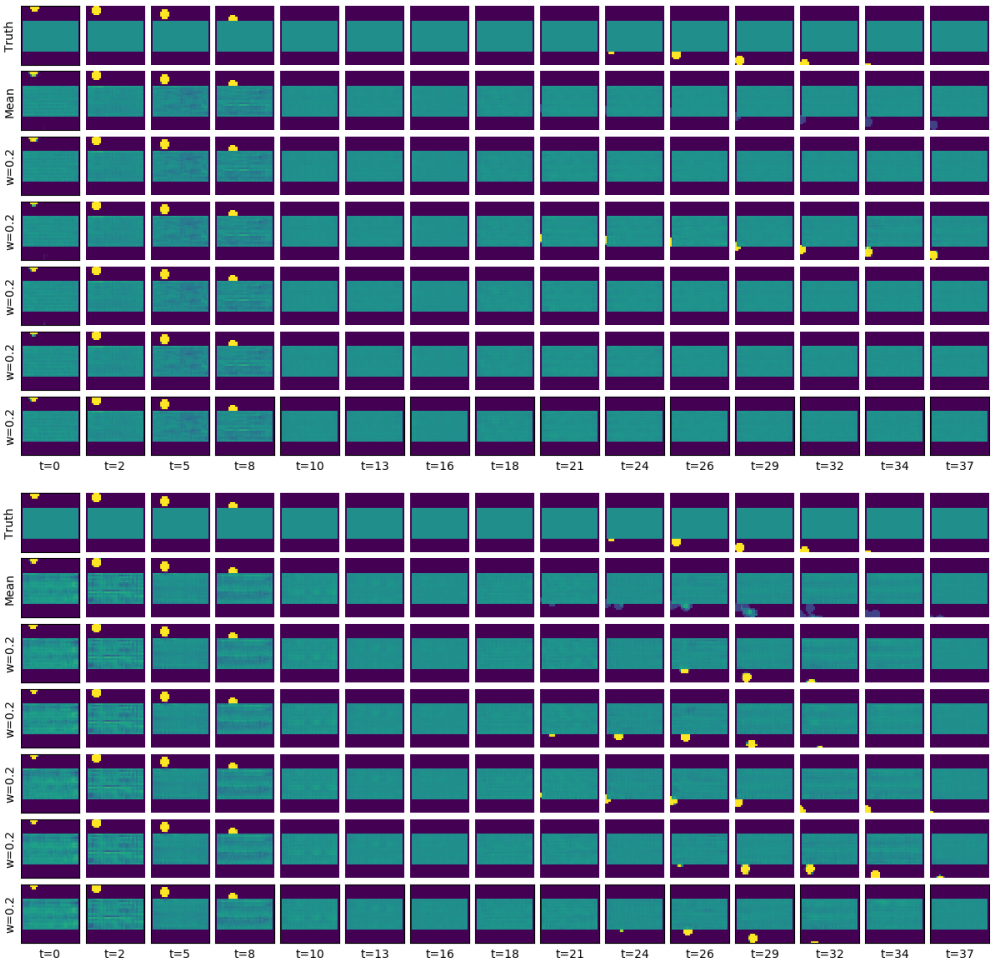


Figure 7: Visualisation of the learned model. Ground truth observations (top row in each sub figure) are only revealed to the algorithm up until t=19 inclusive. The second row shows the prediction averaged over all particles, all following rows show the prediction made by a single particle. *(Top)* IWAE. *(Bottom)* AESMC.

E Additional Experiments

E.1 Optimizing Only Proposal Parameters

We have run experiments where we optimize various ELBO objectives with respect to ϕ with θ fixed in order to see how various objectives have an effect on proposal learning. In particular, we train ELBO_{IS} and ELBO_{SMC} with number of particles $K \in \{10, 100, 1000\}$. Once the training is done, we use the trained proposal network to perform inference using both IS and SMC with number of particles $K_{\text{test}} \in \{10, 100, 1000\}$.

In Figure 8, we see experimental results for the LGSSM described in Section 5.1. We measure the quality of the inference network using a proxy $\sqrt{\sum_{t=1}^{T}(\mu_t^{\text{kalman}} - \mu_t^{\text{approx}})^2}$ where μ_t^{kalman} is the true marginal mean $\mathbb{E}_{p(x_{1:T}|y_{1:T})}[x_t]$ obtained from the Kalman smoothing algorithm and $\mu_t^{\text{approx}} = \left(\sum_{k=1}^{K} w_T^k x_t\right) / \left(\sum_{k=1}^{K} w_T^k\right)$ is an approximate marginal mean obtained from the proposal parameterized by ϕ.

We see that if we train using ELBO_{SMC} with $K_{\text{train}} = 1000$, the performance for inference using SMC (with whichever $K_{\text{test}} \in \{10, 100, 1000\}$) is worse than if we train with ELBO_{IS} with any number of particles $K_{\text{train}} \in \{10, 100, 1000\}$. Examining the other axes of variation:

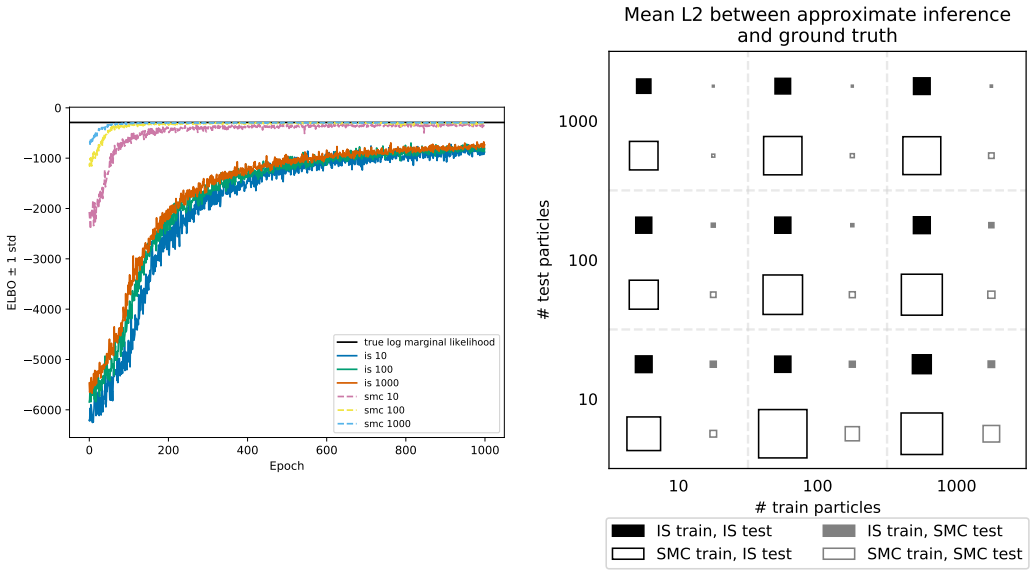


Figure 8: *(Left)* Optimizing ELBO with respect to ϕ for LGSSM. *(Right)* The lengths of the squares are proportional (with a constant factor) to $\sqrt{\sum_{t=1}^{T}(\mu_t^{\text{kalman}} - \mu_t^{\text{approx}})^2}$ which is a proxy for inference quality of ϕ described in the main text. The larger the square, the worse the inference.

- Increasing K_{test} (moving up in Figure 8 (Right)) improves inference.
- Increasing K_{train} (moving to the right in Figure 8 (Right)) worsens inference.
- Among different possible combinations of (training algorithm, testing algorithm), (IS, SMC) $\succ$ (SMC, SMC) $\succ$ (IS, IS) $\succ$ (SMC, IS), where we use "$a \succ b$" to denote that the combination a results in better inference than combination b.