Recursive Monte Carlo and Variational Inference with Auxiliary Variables

Abstract

A key challenge in applying Monte Carlo and variational inference (VI) is the design of *proposals* and *variational families* that are flexible enough to closely approximate the posterior, but simple enough to admit tractable densities and variational bounds. This paper presents *recursive auxiliary-variable inference* (RAVI), a new framework for exploiting flexible proposals, for example based on involved simulations or stochastic optimization, within Monte Carlo and VI algorithms. The key idea is to estimate intractable proposal densities via *meta-inference*: additional Monte Carlo or variational inference targeting the proposal, rather than the model. RAVI generalizes and unifies several existing methods for inference with expressive approximating families, which we show correspond to specific choices of meta-inference algorithm, and provides new theory for analyzing their bias and variance. We illustrate RAVI's design framework and theorems by using them to analyze and improve upon Salimans et al. [33]'s Markov Chain Variational Inference, and to design a novel sampler for Dirichlet process mixtures, achieving state-of-the-art results on a standard benchmark dataset from astronomy and on a challenging data-cleaning task with Medicare hospital data.

1 INTRODUCTION

Monte Carlo and variational inference algorithms are the workhorses of modern probabilistic inference, a fundamental problem with applications in many disciplines [25]. A key challenge in applying these algorithms is the design of *proposal distributions* (in VI, variational families), which can greatly affect their performance [6]. A good proposal should incorporate any knowledge the practitioner might have about the shape of the posterior; however, this goal is often in tension with the requirement that a proposal's marginal density be analytically tractable, in order to compute importance weights, MCMC acceptance probabilities, or gradient updates for VI. The challenge is that proposal distributions that are simple enough to admit tractable densities may not be flexible enough to solve real-world posterior inference problems.

In this paper, we present a new framework, called *Recursive Auxiliary-Variable Inference* (RAVI), for incorporating more complex proposals, without tractable densities, into standard Monte Carlo and VI algorithms. The key idea is to approximate the intractable proposal densities using *meta-inference* [10]: an additional layer of Monte Carlo or variational inference targeting the proposal, rather than the model. RAVI generalizes and unifies several existing methods for inference with expressive proposals [33, 32, 36], which we show correspond to specific choices of meta-inference algorithm (see Table 1 and Appendix B for 13 examples).

Contributions. Our key contributions are:

- the RAVI framework, including new recursive algorithms for IS, VI, SMC, and MH using proposals that lack tractable marginal densities (Sections 2 and 3);
- theorems that precisely characterize the impact of RAVI's approximations on overall inference quality (sampler variance, or tightness of the variational bound) (Section 4); and
- two extended examples of RAVI's application to algorithm design and analysis: (1) a novel variant of Salimans et al. [33]'s Markov Chain Variational Inference (MCVI) algorithm that, unlike vanilla MCVI, scales to handle proposals incorporating long MCMC chains; and (2) a novel sampler for Dirichlet process mixtures that uses a randomized agglomerative clustering algorithm as a proposal, outperforming strong baselines on a standard benchmark from astronomy [15] and a challenging data-cleaning problem using Medicare data [23], adapted from Lew et al. [18].

Submitted to the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022). **To be used for reviewing only**.

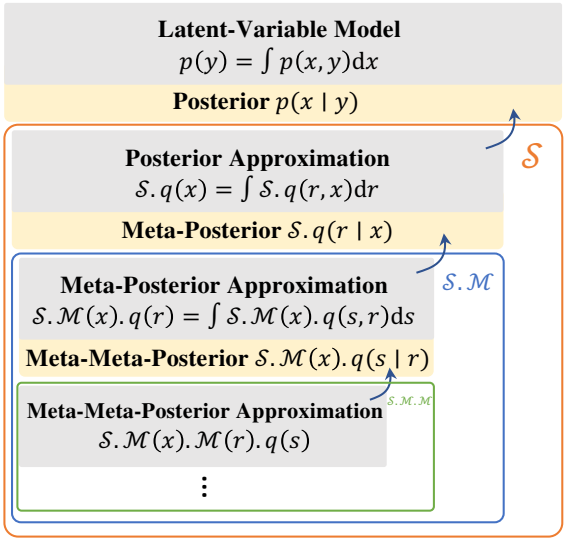

Figure 1: Structure of a RAVI inference strategy $\mathcal{S}$ targeting the posterior $p(x \mid y)$ of a latent-variable model. The proposal $\mathcal{S}.q(x) = \int \mathcal{S}.q(r, x)\mathrm{d}r$ has an intractable marginal density, so the strategy also specifies a *meta-inference* strategy $\mathcal{S}.\mathcal{M}$ that targets $\mathcal{S}.q(r \mid x)$. Nesting continues until the q approximation at some layer has a tractable density, at which point no further meta-inference is needed.

2 RECURSIVE AUXILIARY-VARIABLE INFERENCE

In this section, we introduce the RAVI framework in the context of a running example: we incorporate a chain of MCMC steps into a proposal, so that it can more accurately approximate a posterior distribution. Our approach generalizes Salimans et al. [33]'s Markov Chain Variational Inference (MCVI) algorithm, and fixes a flaw that prevents it from scaling to longer MCMC chains.

An expressive proposal with an intractable marginal density. Let $p(x, y)$ be a latent-variable model and y an observation. We are interested in approximating $p(x \mid y)$ using an expressive proposal $q(x)$, that generates an initial location x_0 from a simple parametric distribution q_0, then iterates M steps of an MCMC kernel T:

$$q(x) = \int q_0(x_0) \prod_{i=1}^{M} T(x_{i-1} \to x_i)\delta_{x_M}(x)\mathrm{d}x_{0:M}.$$

Even when q_0 is a poor approximation to $p(x \mid y)$, $q(x)$ may be close to the posterior, if M is sufficiently high. However, because the density $q(x)$ cannot be tractably evaluated, we cannot use $q(x)$ as a proposal within importance sampling (we have no way to evaluate the importance weight $\frac{p(x,y)}{q(x)}$), nor as a variational family in VI (we cannot estimate the ELBO $\mathcal{L} = \mathbb{E}_{x\sim q}[\log \frac{p(x,y)}{q(x)}]$ or its gradient, making it impossible to learn p's or q's parameters).

Approximating intractable densities with meta-inference. The goal of RAVI is to enable inference even when we cannot compute the marginal densities of our proposals and variational families. To apply RAVI, we must specify not just the proposal itself but a complete *inference strategy*:

Definition. An *inference strategy* $\mathcal{S}$ *targeting* $\pi(x)$ specifies:

- a posterior approximation $\mathcal{S}.q(x) \approx \pi(x)$[1] that either has a tractable density, or is the marginal distribution of a joint distribution with a tractable density, i.e. $\mathcal{S}.q(x) = \int \mathcal{S}.q(r, x)\mathrm{d}r$, and,
- if $\mathcal{S}.q$ has an intractable marginal density, a *meta-inference strategy* $\mathcal{S}.\mathcal{M}$, assigning to each value of x an inference strategy $\mathcal{S}.\mathcal{M}(x)$ targeting $\mathcal{S}.q(r \mid x)$.

Figure 1 illustrates the recursive structure of an inference strategy. The key novelty is the inclusion of *meta-posterior approximations*: additional proposals that the user must specify for inferring auxiliary variables introduced by existing proposal distributions. In our running example, we take $\mathcal{S}.q(x)$ to be our MCMC-based posterior approximation: it lacks a tractable density, but is the marginal of a tractable joint density $\mathcal{S}.q(x_{0:M}, x)$ over entire MCMC traces. A *meta-posterior approximation*, then, is a probability distribution $\mathcal{S}.\mathcal{M}(x).q(x_{0:M})$ that approximates the *meta-posterior* $\mathcal{S}.q(x_{0:M} \mid x)$: the distribution over traces of the MCMC chain, given the final location x.

The meta-posterior approximations enable RAVI to estimate the intractable marginal density of the top-level posterior approximation, to compute weights and gradients:

In Monte Carlo: If $\mathcal{S}.q(x) = \int \mathcal{S}.q(r, x)\mathrm{d}r$ is intended for use as a Monte Carlo proposal, RAVI uses meta-inference to obtain an unbiased estimate of $\frac{1}{\mathcal{S}.q(x)}$ (Algorithm 2), which is then multiplied by $p(x, y)$ to estimate the importance weight $\frac{p(x,y)}{\mathcal{S}.q(x)}$. This process relies on the *harmonic mean identity* [31], that for any meta-posterior approximation h,

$$\mathbb{E}_{\mathcal{S}.q(r|x)}\left[\frac{h(r)}{\mathcal{S}.q(r, x)}\right] = \frac{1}{\mathcal{S}.q(x)}\mathbb{E}\left[\frac{h(r)}{\mathcal{S}.q(r \mid x)}\right] = \frac{1}{\mathcal{S}.q(x)}.$$

(Harmonic mean estimators are infamous for having potentially infinite variance, but only when h is set to a broad prior; we give a general analysis of the variance of RAVI's importance weights in Section 4.)

In Variational Inference: If $\mathcal{S}.q(x) = \int \mathcal{S}.q(r, x)\mathrm{d}r$ is intended as a variational family, then RAVI uses the meta-posterior approximation to formulate an *upper bound* on $\log \mathcal{S}.q(x)$: for any meta-posterior approximation $h(r)$,

$$\log \mathcal{S}.q(x) \leq \mathcal{U}(x) := \mathbb{E}_{\mathcal{S}.q(r|x)}[\log \mathcal{S}.q(r, x) - \log h(r)].$$

[1] To simplify the exposition, we assume that if an inference strategy $\mathcal{S}$ targets π, then the approximation $\mathcal{S}.q$ is *mutually absolutely continuous* with π, i.e. the measure-zero events under π are exactly the same as those under $\mathcal{S}.q$. This requirement can be relaxed somewhat; see Appendix C.

Algorithm	Intractable posterior approximation $\mathcal{S}.q$	Meta-posterior approximation $\mathcal{S}.\mathcal{M}(x).q$
IWAE [5]	SIR with tractable proposal $q(x) = \sum_{j=1}^{K} \int \prod_{i=1}^{K} q(x_i) \frac{w_j}{\sum_{l=1}^{K} w_l} \delta_{x_j}(x) \mathrm{d}x_{1:K}$	Conditional SIR $h(x_{1:K}, j) = \frac{1}{K} \delta_x(x_j) \prod_{i \neq j} q(x_i)$
HVM [32]	Latent-variable proposal $q(x) = \int q(z, x) \mathrm{d}z$	Tractable proposal $h(z; x)$
MCVI [33]	MCMC chain initialized from tractable proposal $q(x) = \int q(x_0) \prod_{i=1}^{M} T(x_i; x_{i-1}) \delta_{x_M}(x) \mathrm{d}x_{0:M}$	Sequence of tractable proposals $h(x_{0:M}; x) = \delta_x(x_M) \prod_{i=0}^{M-1} r_i(x_i; x_{i+1})$
IWHVI [36]	Latent-variable proposal $q(x) = \int q(z, x) \mathrm{d}z$	SIR with tractable proposal $h(z; x) = \sum_{j=1}^{K} \int \prod_{i=1}^{K} h(z_i; x) \frac{w_j}{\sum_{l=1}^{K} w_l} \delta_{z_j}(z) \mathrm{d}z_{1:K}$
DIWHVI [36]	SIR with latent-variable proposal $q(x) = \sum_{j=1}^{K} \int \prod_{i=1}^{K} q(z_i, x_i) \cdot \frac{w_j}{\sum_{l=1}^{K} w_l} \delta_{x_j}(x) \mathrm{d}(z, x)_{1:K}$	Conditional SIR for $j, x_{1:K}, z_{-j}$ $h(z_{-j}, x_{1:K}, j; x) = \frac{1}{K} \delta_x(x_j) \prod_{i \neq j} q(z_i, x_i)$ SIR with tractable proposal for z_j $h(z_j; x) = \sum_{k=1}^{K} \int \prod_{i=1}^{K} h(z_j^{(i)}; x) \frac{w_k}{\sum_{m=1}^{K} w_m} \delta_{z_j^{(k)}}(z_j) \mathrm{d}z_j^{(1:K)}$

Table 1: Many existing approaches to inference with flexible proposals can be understood as instances of RAVI, with particular choices of inference strategy $\mathcal{S}$. In some cases, including IWHVI and DIWHVI, the meta-posterior approximation is itself intractable, necessitating meta-meta-inference $\mathcal{S}.\mathcal{M}(x).\mathcal{M}$. See Appendix B for eight additional examples.

This follows from Jensen's inequality, and the harmonic mean identity from above. With this upper bound in hand, we formulate a surrogate ELBO $\mathcal{L}_{\mathcal{S}} = \mathbb{E}_{\mathcal{S}.q(x)}[\log p(x, y) - \mathcal{U}(x)] \leq \mathcal{L}$, which we can tractably estimate and optimize via stochastic gradient descent (Algorithm 3).

In Section 3, we show how similar estimators can be built up recursively when the meta-posterior approximations themselves have intractable marginal densities.

A meta-inference strategy that recovers the MCVI objective [33]. In our running example, where the auxiliary randomness r is a trace $x_{0:M}$ of locations visited by MCMC, one option for meta-inference is to learn neurally parameterized reverse Markov kernels $R_i(x_{i+1} \to x_i)$, and apply them in sequence to infer a plausible trace of MCMC steps leading to the final location x:

$$\mathcal{S}.\mathcal{M}(x).q(x_{0:M}) = \delta_x(x_M) \prod_{i=0}^{M-1} R_i(x_{i+1} \to x_i).$$

This approximation to $\mathcal{S}.q(x_{0:M} \mid x)$ has a tractable density, and so completely specifies the meta-inference strategy $\mathcal{S}.\mathcal{M}$; there is no need to specify a *meta*-meta-inference strategy. Given $\mathcal{S}$, RAVI optimizes the surrogate objective $\mathcal{L}_{\mathcal{S}} = \mathbb{E}_{x \sim \mathcal{S}.q}[\log p(x, y) - \mathcal{U}_{\mathcal{S}.\mathcal{M}(x)}]$, where

$$\mathcal{U}_{\mathcal{S}.\mathcal{M}(x)} = \mathbb{E}_{x_{0:M} \sim \mathcal{S}.q(x_{0:M} \mid x)} \left[\log \frac{\mathcal{S}.q(x_{0:M}, x)}{\mathcal{S}.\mathcal{M}(x).q(x_{0:M})} \right].$$

For the above choice of $\mathcal{S}.\mathcal{M}$, the RAVI objective $\mathcal{L}_{\mathcal{S}}$ exactly coincides with the Markov Chain Variational Inference (MCVI) objective of Salimans et al. [33]. In fact, RAVI unifies and generalizes many existing methods; 13 examples are collected in Table 1 and Appendix B.

Analyzing MCVI within the RAVI framework. Framing MCVI as a RAVI algorithm lets us analyze it using general theory about RAVI objectives. For example, we have seen that the relative tightness of the bound $\mathcal{L}_{\mathcal{S}}$ is controlled by the quality of meta-inference:

$$\mathcal{L} - \mathcal{L}_{\mathcal{S}} = \mathbb{E}_{\mathcal{S}.q(x)}[KL(\mathcal{S}.q(x_{0:M} \mid x) || \mathcal{S}.\mathcal{M}(x).q(x_{0:M}))].$$

We can use this characterization to analyze the MCVI objective's behavior as M grows, i.e., as MCMC steps are added. Informally, as the MCMC chain begins to mix, the marginal distribution $\mathcal{S}.q(x)$ over the final location of the chain should grow closer to the posterior $p(x \mid y)$, tightening the (intractable) ELBO $\mathcal{L}$. Unfortunately, the meta-inference gap $\mathcal{L} - \mathcal{L}_{\mathcal{S}}$ *grows* with M, unless each kernel R_i exactly captures the local posterior $\mathcal{S}.q(x_i \mid x_{i+1})$. (This can be seen as an instance of the well-known *degeneracy problem* of sequential importance sampling [14, Proposition 1].) As MCMC converges, the rate of improvement in $\mathcal{L}$ slows, and the meta-inference penalty for increasing the chain's length eventually outweighs the benefit of improving the posterior approximation $\mathcal{S}.q$. The red curves in Figure 2 show this phenomenon playing out on two toy targets: we see that $\mathcal{L}_{\mathcal{S}}$ does become tighter as more MCMC steps are added, but only to a point, before the bound begins to *loosen*.

Resolving the issue with improved meta-inference. RAVI clarifies that the variational bound loosens with increasing M due to poor meta-inference: as the MCMC chain grows longer, error in the learned backward kernels accumulates. This analysis also points to a solution: use a meta-inference strategy $\mathcal{S}.\mathcal{M}$ that *can* scale to longer MCMC histories.

A standard approach to resolving the degeneracy problem when inferring sequences of latent variables is *sequential Monte Carlo* (SMC) [12]. SMC tracks K hypotheses about a latent sequence, periodically weighting the hypotheses and *resampling*, to clone promising particles and cull poor ones. Using RAVI, we can use SMC for *meta-inference*: we choose $\mathcal{S}.\mathcal{M}(x).q(x_{0:M})$ to generate a collection of K

RAVI Inference Strategy 1: RAVI-MCVI

Posterior Approx. `rmcvi(`M, K`).q()`

Target of inference : latent variable x

Auxiliary variables : MCMC trace $x_{0:M}$

$x_0 \sim q_0$

for $i \in 1, \ldots, M$ **do**

  $x_i \sim T(x_{i-1} \to \cdot)$

return x_M

Meta-Posterior Approx. `rmcvi(`M, K`).M(`x`).q()`

Target of inference : MCMC trace $x_{0:M}$

Auxiliary variables : SMC particles $x_{0:M}^{1:K}$, ancestor indices $a_0, a_{1:M}^{1:K}$

for $k \in 1, \ldots, K$ **do**

  $(x_M^k, w_M^k, t_k) \leftarrow (x, q_m(x), [x])$

for $i \in M-1, \ldots, 0$ **do**

  for $k \in 1, \ldots, K$ **do**

    $a_{i+1}^k \sim \text{Discrete}(w_{i+1}^{1:K})$

    $x_i^k \sim R_i(x_{i+1}^{a_{i+1}^k} \to \cdot)$ `// MCVI backward kernel`

    $w_i^k \leftarrow \frac{q_i(x_i^k) T(x_i \to x_{i+1}^{a_{i+1}^k})}{q_{i+1}(x_{i+1}^{a_{i+1}^k}) R_i(x_{i+1}^{a_{i+1}^k} \to x_i^k)}$

    $t_k \leftarrow [x_i^k, t_k^{a_{i+1}^k} \ldots]$

$a_0 \sim \text{Discrete}(w_0^{1:K})$

return t_{a^0}

Meta-Meta-Posterior Approx. `rmcvi(`M, K`).M(`x`).M(`$x_{0:M}$`).q()`

Target of inference : SMC particles $x_{0:M}^{1:K}$, ancestor indices $a_0, a_{1:M}^{1:K}$

Auxiliary variables : None

for $i \in 0, \ldots, M$ **do**

  $b_i \sim \text{Uniform}(1, K)$

for $k \in 1, \ldots, K$ **do**

  $(x_M^k, w_M^k) \leftarrow (x, q_m(x)])$

for $i \in M-1, \ldots, 0$ **do**

  for $k \in 1, \ldots, K$ **do**

    if $k = b_i$ **then**

      $(a_{i+1}^k, x_i^k) \leftarrow (b_{i+1}, x_i)$

    else

      $a_{i+1}^k \sim \text{Discrete}(w_{i+1}^{1:K})$

      $x_i^k \sim R_i(x_{i+1}^{a_{i+1}^k} \to \cdot)$

    $w_i^k \leftarrow \frac{q_i(x_i^k) T(x_i \to x_{i+1}^{a_{i+1}^k})}{q_{i+1}(x_{i+1}^{a_{i+1}^k}) R_i(x_{i+1}^{a_{i+1}^k} \to x_i^k)}$

$a_0 \leftarrow b_0$

return $(a_0, a_{1:M}^{1:K}, x_{0:M}^{1:K})$

possible backward MCMC trajectories, using SMC, before selecting one to return. This meta-posterior approximation is shown in RAVI Inference Strategy 1.

This algorithm does not itself have a tractable marginal density: computing $\mathcal{S}.\mathcal{M}(x).q(x_{0:M})$ would require large sums over the resampling variables and intractable integrals over the particle collection. But this is where RAVI's recursive structure comes into play: a meta-inference strategy may use an intractable meta-posterior approximation, so long as we attach a *meta*-meta-inference strategy $\mathcal{S}.\mathcal{M}(x).\mathcal{M}(x_{0:M}).q(a_0, a_{1:M}^{1:K}, x_{0:M}^{1:K})$. In this case meta-meta-inference must infer the auxiliary variables of SMC (ancestor variables and unchosen trajectories), given the final chosen trajectory $x_{0:M}$. For this we can use the *conditional SMC* algorithm [1], which runs SMC, with the same auxiliary variables, but constrained to ensure that one of the K particles traces the observed trajectory $x_{0:M}$. Because cSMC introduces no new auxiliary variables, it has a tractable density, and there is no need to specify a fourth layer of meta-inference. The full tower of posterior approximations is given in RAVI Inference Strategy 1.

In Section 5, we compare MCVI to `rmcvi`, for different K and M. Figure 2 shows that meta-inference error is greatly reduced by using SMC, so that the variational bound $\mathcal{L}_{\mathcal{S}}$ continues to tighten as the MCMC chain grows longer.

Using the inference strategy within a Monte Carlo algorithm, to estimate marginal likelihoods from MCMC results. Our inference strategy $\mathcal{S}$ can also be used as proposal within Monte Carlo algorithms, such as importance sampling. In the context of our example, where $\mathcal{S}.q$ incorporates M steps of a Markov chain, this allows us to assign an *importance weight* to each run of the Markov chain. The weight is an unbiased estimate of the marginal likelihood $p(y)$ of the model; thus, we can view the algorithm as a way to derive marginal likelihood estimates from MCMC runs, a task of long-standing interest in the Monte Carlo community [29]. In Section 5, we show that in some settings MCVI compares favorably a standard algorithm for the task, annealed importance sampling (AIS) [29].

3 ALGORITHMS

In this section, we present algorithms for using RAVI inference strategies within Monte Carlo and variational inference algorithms, as proposals and variational families.

RAVI for Importance Sampling and SMC. In importance sampling and SMC algorithms, proposals q are used to (1) generate proposed values $x \sim q$, and (2) compute importance weights $\frac{p(x)}{q(x)}$. But in both IS and SMC, it suffices to produce *unbiased estimates* of $\frac{p(x)}{q(x)}$ [20, 7]. RAVI exploits this degree of freedom to generate proper importance weights even when $q(x)$ is intractable. Suppose $\tilde{\pi} = Z\pi$ is an unnormalized target density, and $\mathcal{S}$ is a RAVI inference strategy targeting π. Algorithm 1 simulates $x \sim \mathcal{S}.q$ and computes an unbiased estimate $\hat{Z}$ of $\frac{\tilde{\pi}(x)}{\mathcal{S}.q(x)}$:

Theorem 1. *Given a strategy $\mathcal{S}$ targeting $\pi(x)$, Algorithm 1 generates $x \sim \mathcal{S}.q$ and a weight $\hat{Z}$ such that $\mathbb{E}[\hat{Z} \mid x] = \frac{\tilde{\pi}(x)}{\mathcal{S}.q(x)} = Z\frac{\pi(x)}{\mathcal{S}.q(x)}$.*

When $\mathcal{S}.q$ has a tractable marginal density, Algorithm 1 computes an exact importance weight. Otherwise, it calls Algorithm 2, which uses the meta-inference strategy $\mathcal{S}.\mathcal{M}(x)$ to estimate $\frac{1}{\mathcal{S}.q(x)}$. The proof of Theorem 1 is by induction on the level of nesting in the strategy (see Appendix A).

RAVI for MCMC. When models or proposals (or both) in a Metropolis-Hastings sampler do not have tractable closed-form densities, RAVI inference strategies enable computation of MH acceptance probabilities. We present the algorithm and its justification in Appendix A.5.

RAVI for Variational Inference. Let $p_\theta(x, y)$ be a latent-variable generative model with parameters θ, and $\mathcal{S}_\theta(y)$ is a family of strategies targeting $p_\theta(x \mid y)$. Given a dataset y, variational inference can be applied to maximize (a lower

Recursive Monte Carlo Estimation

Algorithm 1: RAVI Importance Sampling (`IMPORTANCE`)
Input: unnormalized target $\tilde{\pi}(x) = Z\pi(x)$
Input: inference strategy $\mathcal{S}$
Output: $(x, \hat{Z})$ properly weighted for $\pi(x)$, s.t. $\mathbb{E}[\hat{Z}] = Z$
if *$\mathcal{S}.q$ has a tractable marginal density* **then**
  $x \sim \mathcal{S}.q$
  $w \leftarrow \frac{1}{\mathcal{S}.q(x)}$
else if $\mathcal{S}.q(x) = \int \mathcal{S}.q(r, x)dr$ **then**
  $(r, x) \sim \mathcal{S}.q$
  $w \leftarrow \texttt{HME}(\mathcal{S}.q(\cdot \mid x), r, \mathcal{S}.\mathcal{M}(x))$
return $(x, w\tilde{\pi}(x))$

Algorithm 2: RAVI Harmonic Mean Estimation (`HME`)
Input: unnormalized target $\tilde{\pi}(x) = Z\pi(x)$
Input: exact sample $x \sim \pi$
Input: inference strategy $\mathcal{S}$
Output: unbiased estimate $\frac{1}{Z}$ of $\frac{1}{Z}$
if *$\mathcal{S}.q$ has a tractable marginal density* **then**
  $w \leftarrow \mathcal{S}.q(x)$
else if $\mathcal{S}.q(x) = \int \mathcal{S}.q(r, x)dr$ **then**
  $(r, w) \leftarrow \texttt{IMPORTANCE}(\mathcal{S}.q(\cdot, x), \mathcal{S}.\mathcal{M}(x))$
return $w/\tilde{\pi}(x)$

Recursive Variational Objectives and Gradient Estimation

Algorithm 3: RAVI ELBO and gradient estimator ($\texttt{ELBO}\nabla$)
Input: model $p(x, y)$
Input: data y
Input: inference strategy $\mathcal{S}$
Output: unbiased estimates of $\mathcal{L}(y)$ and of $\nabla_\theta \mathcal{L}(y)$
if *$\mathcal{S}.q$ has a tractable marginal density* **then**
  $x \sim \mathcal{S}.q$
  $(\hat{U}, \widehat{\nabla_\theta}) \leftarrow (\log \mathcal{S}.q(x), \nabla_\theta \log \mathcal{S}.q(x) \cdot (1 + \log \mathcal{S}.q(x)))$
  $\mathbf{g} \leftarrow \nabla_\theta \log \mathcal{S}.q(x)$
else if $\mathcal{S}.q(x) = \int \mathcal{S}.q(r, x)dr$ **then**
  $(r, x) \sim \mathcal{S}.q$
  $(\hat{U}, \widehat{\nabla_\theta}, \mathbf{g}) \leftarrow \texttt{EUBO}\nabla(\mathcal{S}.q, x, r, \mathcal{S}.\mathcal{M}(x))$
$\hat{L} \leftarrow \log p(x, y) - \hat{U}$
$\widehat{\nabla_\theta}' \leftarrow \nabla_\theta \log p(x, y) + \mathbf{g} \log p(x, y) - \widehat{\nabla_\theta}.$
return $(\hat{L}, \widehat{\nabla_\theta}')$

Algorithm 4: RAVI EUBO and gradient estimator ($\texttt{EUBO}\nabla$)
Input: model $p(x, y)$
Input: data y
Input: exact sample $x \sim p(x \mid y)$
Input: inference strategy $\mathcal{S}$
Output: unbiased estimates of $\mathcal{U}(y)$ and of $\nabla_\theta \mathcal{U}(y)$
Output: quantity $\mathbf{g}$ (see Thm. 2)
if *$\mathcal{S}.q$ has a tractable marginal density* **then**
  $(\hat{L}, \widehat{\nabla_\theta}) \leftarrow (\log \mathcal{S}.q(x), \nabla_\theta \log \mathcal{S}.q(x))$
else if $\mathcal{S}.q(x) = \int \mathcal{S}.q(r, x)dr$ **then**
  $(\hat{L}, \widehat{\nabla_\theta}) \leftarrow \texttt{ELBO}\nabla(\mathcal{S}.q, x, \mathcal{S}.\mathcal{M}(x))$
$\hat{U} \leftarrow \log p(x, y) - \hat{L}$
$\mathbf{g} \leftarrow \nabla_\theta \log p(x, y)$
$\widehat{\nabla_\theta}' \leftarrow \nabla_\theta \log p(x, y) + \mathbf{g} \cdot \hat{U} - \widehat{\nabla_\theta}$
return $(\hat{U}, \widehat{\nabla_\theta}', \mathbf{g})$

bound on) $\log p_\theta(y)$, and also to optimize parameters of the posterior approximations in $\mathcal{S}_\theta$, to bring them closer (in KL divergence) to their targets. Let

$$\mathcal{L}(y) := \mathbb{E}[\log \hat{Z}] \leq \log p(y)$$
$$\text{and } \mathcal{U}(y) := \mathbb{E}[\log \check{Z} | y] \geq \log p(y),$$

where $\hat{Z}$ is the estimate returned by Alg. 1 on $\mathcal{S}(y)$ targeting $p(x \mid y)$, and $\check{Z}$ is the inverse of the weight returned from Alg. 2 on the joint distribution $p(x, y)$ with family of strategies $\mathcal{S}$. Because $\hat{Z}$ is an unbiased estimate of $p_\theta(y)$, and $\frac{1}{\check{Z}}$ is an unbiased estimate of $\frac{1}{p_\theta(y)}$, we have by Jensen's inequality that $\mathcal{L}$ and $\mathcal{U}$ are lower and upper bounds (respectively) on $\log p_\theta(y)$. As such, we can fit the model parameters θ to data y by minimizing $\mathcal{U}(y)$ or maximizing $\mathcal{L}(y)$.

Recursive stochastic gradient estimation. Algs. 3 and 4 (above) give recursive procedures for estimating $\mathcal{L}(y)$ and its gradient $\nabla_\theta \mathcal{L}(y)$ with respect to all of the parameters θ that are used by any layer of the strategy, as well as $\mathbb{E}_{y \sim p}[\mathcal{U}(y)]$ and its gradient $\nabla_\theta \mathbb{E}_{y \sim p}[\mathcal{U}(y)]$. These procedures employ score function estimation of gradients, but it is straightforward to incorporate baselines within each procedure to reduce variance. Depending on the proposals in $\mathcal{S}$, variants of these procedures using the reparametrization trick may also be possible (Appendix E).

Theorem 2. *Given a model $p_\theta(x, y)$ and an inference strategy $\mathcal{S}_\theta$ targeting $p_\theta(x \mid y)$, Alg. 3 yields unbiased estimates of $\mathcal{L}(y)$ and of $\nabla_\theta \mathcal{L}(y)$. Furthermore, when $(x, y) \sim p_\theta$, Alg. 4 yields (i) $\hat{U}$ such that $\mathbb{E}[\hat{U} \mid y] = \mathcal{U}(y)$, (ii) $\widehat{\nabla_\theta}$ such that $\mathbb{E}_{y \sim p}[\mathbb{E}[\widehat{\nabla_\theta} | y]] = \nabla_\theta \mathbb{E}_{y \sim p_\theta}[\mathcal{U}(y)]$, and (iii) a value $\mathbf{g}$ such that for any function R that does not depend on θ, $\mathbb{E}_{y \sim p}[\mathbb{E}[\mathbf{g} \cdot R(y) | y]] = \nabla_\theta \mathbb{E}_{y \sim p_\theta}[R(y)]$ if $\nabla_\theta \mathbb{E}_{y \sim p_\theta}[R(y)]$ is defined.*

In Section 4, we show the tightness of the variational bounds $\mathcal{L}$ and $\mathcal{U}$ is given by sums of KL divergences between posterior approximations in $\mathcal{S}_\theta$ and their targets. Thus, optimizing these bounds improves the posterior approximations, either encouraging mass-capturing or mode-seeking behavior.

4 THEORETICAL ANALYSIS

We now present theorems characterizing the quality of RAVI inference: Thm. 3 concerns the variance of weights in a Monte Carlo sampler, and Thm. 4 the tightness of variational bounds. In both cases, error is related to each approximation in the RAVI strategy's divergence to its target posterior.

Sampler variance in Monte Carlo. Let $\tilde{\pi} = Z\pi$ be an unnormalized target density, and $\mathcal{S}$ an inference strategy targeting π. We write $\hat{Z}(\tilde{\pi}, \mathcal{S})$ for the estimate of Z returned

by Algorithm 1, and $\mathrm{Var}_{\hat{Z}}(\pi, \mathcal{S})$ for the *relative variance* of the estimator, $\mathrm{Var}(\frac{\hat{Z}(\tilde{\pi},\mathcal{S})}{Z})$, which does not depend on Z (and therefore is a function of π, not $\tilde{\pi}$). Similarly, we write $\check{Z}(\tilde{\pi}, \mathcal{S})$ for the reciprocal of the estimator of $\frac{1}{Z}$ returned by Algorithm 2, run with an input $x \sim \pi$. $\mathrm{Var}_{\check{Z}}(\pi, \mathcal{S})$ is its relative variance, $\mathrm{Var}(\frac{Z}{\check{Z}(\tilde{\pi},\mathcal{S})})$.

Theorem 3. *Consider an unnormalized target distribution $\tilde{\pi}(x) = Z\pi(x)$ and an inference strategy $\mathcal{S}$ targeting $\pi(x)$. Then the relative variances of the estimators $\hat{Z}(\tilde{\pi}, \mathcal{S})$ and $\check{Z}(\tilde{\pi}, \mathcal{S})$ are given by the following recursive equations:*

$$\begin{aligned}
&\mathrm{Var}_{\hat{Z}}(\pi, \mathcal{S}) = \chi^2(\pi||\mathcal{S}.q) + \\
&\qquad \mathbb{E}_{x\sim\mathcal{S}.q}\left[\left(\frac{\pi(x)^2}{\mathcal{S}.q(x)^2}\right) \cdot \mathrm{Var}_{\check{Z}}(\mathcal{S}.q(\cdot \mid x), \mathcal{S}.\mathcal{M}(x))\right] \\
&\mathrm{Var}_{\check{Z}}(\pi, \mathcal{S}) = \chi^2(\mathcal{S}.q||\pi) + \\
&\qquad \mathbb{E}_{x\sim\pi}\left[\left(\frac{\mathcal{S}.q(x)^2}{\pi(x)^2}\right) \cdot \mathrm{Var}_{\hat{Z}}(\mathcal{S}.q(\cdot \mid x), \mathcal{S}.\mathcal{M}(x))\right]
\end{aligned}$$

When $\mathcal{S}.q$ is tractable, the second term of each sum is 0.

Tightness of variational bounds. In VI, the tightness of the variational bounds $\mathcal{L}$ and $\mathcal{U}$ can be characterized as a sum of a KL divergence and a term measuring meta-inference error. Let $\mathcal{L}(p, y, \mathcal{S})$ be the objective optimized by Algorithm 3 given target model $p(x, y)$, observed data y, and inference strategy $\mathcal{S}$ targeting $p(x \mid y)$—and similarly for $\mathcal{U}(p, y, \mathcal{S})$ and Algorithm 4. The random variables $\hat{\mathcal{L}}$ and $\hat{\mathcal{U}}$ can be viewed as biased estimators of $\log p(y)$; writing their bias as $\mathrm{Bias}_{\mathcal{L}}(p, y, \mathcal{S})$ (and similarly for $\mathcal{U}$), we have:

Theorem 4. *Consider a joint distribution $p(x, y)$ and an inference strategy $\mathcal{S}$ targeting $p(x \mid y)$. Then the following equations give the bias of $\hat{\mathcal{L}}$ and $\hat{\mathcal{U}}$ as estimators of $\log p(y)$:*

$$\begin{aligned}
\mathrm{Bias}_{\mathcal{L}}(p, y, \mathcal{S}) = &-\mathrm{KL}(\mathcal{S}.q||p(\cdot \mid y)) \\
&- \mathbb{E}_{x\sim\mathcal{S}.q}[\mathrm{Bias}_{\mathcal{U}}(\mathcal{S}.q, x, \mathcal{S}.\mathcal{M}(x))] \\
\mathrm{Bias}_{\mathcal{U}}(p, y, \mathcal{S}) = &\,\mathrm{KL}(p(\cdot \mid y)||\mathcal{S}.q) \\
&- \mathbb{E}_{x\sim p(\cdot|y)}[\mathrm{Bias}_{\mathcal{L}}(\mathcal{S}.q, x, \mathcal{S}.\mathcal{M}(x))]
\end{aligned}$$

where the second term in each equation is 0 when $\mathcal{S}.q$ has a tractable marginal density.

This means that maximizing $\mathcal{L}$, or minimizing $\mathcal{U}$, also minimizes these KL divergences. In particular, maximizing $\mathcal{L}(y)$ minimizes a 'mode-seeking' KL from $\mathcal{S}.q$ to the posterior, whereas minimizing $\mathbb{E}_{y\sim p}[\mathcal{U}(y)]$, e.g. by following the gradients of Alg. 4, implements amortized variational inference, and encourages $\mathcal{S}.q$ to cover the mass of the posterior.

Inference and Meta-Inference. In both Theorems 3 and 4, the first term of the sum is a divergence between $\mathcal{S}.q(x)$, the intractable posterior approximation, and the actual target posterior $p(x \mid y)$. The other term measures the *expected* quality of meta-inference. Thus the overall error of a RAVI algorithm can be understood as decomposing cleanly into (1) the mismatch between the posterior and the intractable proposal, and (2) the error introduced by meta-inference.

5 EXPERIMENTS

5.1 IMPROVING MCVI

In Section 2, we developed a variant of Salimans et al. [33]'s MCVI algorithm that used SMC for meta-inference. In Figure 2, we compare vanilla MCVI to the RAVI variant, with varying K (number of particles used for meta-inference) and M (number of MCMC steps in the variational family).

Experimental details. For the MCMC kernel T, we use Langevin ascent with step size 0.015. For the meta-inference proposals $R_i(x_{i+1} \rightarrow x_i)$, we use $\mathcal{N}(x_i; f_\mu(x_{i+1}, i), e^{f_{\log\sigma}(x_{i+1},i)})$, where f is a 4-layer MLP, the step number i is encoded as a one-hot vector, and f outputs the mean μ and log standard deviation $\log\sigma$ for a conditionally Gaussian proposal. The same f is used for each experiment, and is trained on forward rollouts of MCMC (equivalent to using Alg. 3 on `rmcvi` with $K = 1$). The unimodal model is Gaussian with $\sigma = 0.2$, and the multimodal model is a mixture of 3 Gaussians with standard deviations $0.2, 0.3, \&2.0$. The distributions q_i used for importance weighting during sequential Monte Carlo meta-inference are Gaussians with learned μ and σ.

Results. Figure 2 plots the gap $\log p(y) - \mathcal{L}$ for each algorithm's variational bound $\mathcal{L}$. By Theorem 4 this gap is the sum of two terms: $KL(\mathcal{S}.q||p(x \mid y))$ and the expected meta-inference divergence $\mathbb{E}_{x\sim\mathcal{S}.q}[KL(\mathcal{S}.q(x_{0:M} \mid x)||\mathcal{S}.\mathcal{M}(x).q(x_{0:M}))]$. The first term is constant across the algorithms, since they all use the same MCMC-based posterior approximation, so the plots primarily illustrate differences in the quality of meta-inference. MCVI's meta-inference steadily worsens as the chain's length grows, and after 15-25 steps, the meta-inference cost of adding new steps outweighs the benefits to $\mathcal{S}.q$, causing the bound $\mathcal{L}$ to loosen. Our variant, with SMC meta-inference, does not suffer the same penalty, and continues to improve as more steps are added. As discussed in Section 2, the same inference strategy (`rmcvi`) can be used within an importance sampler to derive unbiased marginal likelihood estimates from MCMC runs. The right-hand plot in Figure 2 shows that this technique can yield accurate estimates with less computation than AIS [29], at least on simple targets. (To fairly account for the computational cost of meta-inference, in the RAVI algorithm we multiply M by K when plotting the total number of MCMC steps.) Because the variance of AIS is bounded below by sums of divergences between subsequent pairs of intermediate target distributions, the MCMC chain must be long enough to support a very fine annealing schedule, without large jumps. By contrast, RAVI-MCVI requires only that the marginal distribution of the chain be a good approximation to the posterior, and that SMC meta-inference is sufficiently accurate. For some problems, this may be less expensive than the long chain required by AIS.

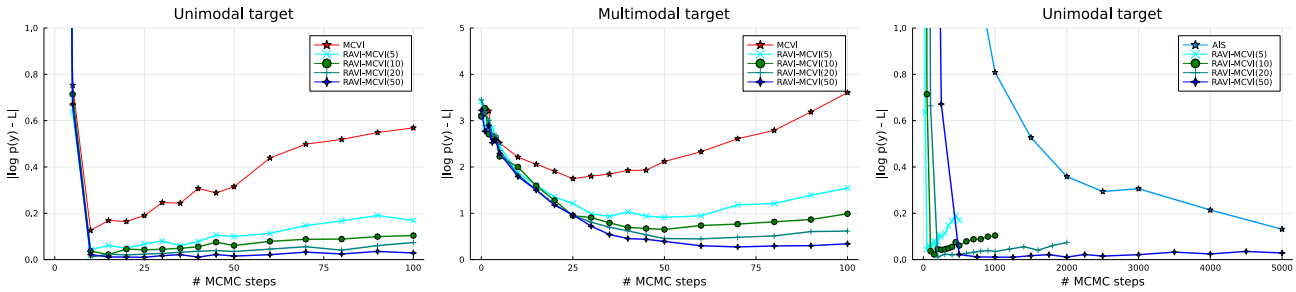

Figure 2: Improving Markov Chain Variational Inference with RAVI. **Left and Middle:** On unimodal and multimodal targets, MCVI begins to degrade after 15-25 steps of MCMC. RAVI-MCVI with sufficiently many particles continues to improve as more MCMC steps are added. **Right:** When MCMC converges quickly to a reasonable approximation of the posterior, RAVI-MCVI can give more accurate estimates of marginal likelihoods than standard techniques such as AIS. The x axis of this plot counts total MCMC steps simulated, whether as part of inference or meta-inference; for RAVI-MCVI(K), this is KM, where M is the length of the forward Markov chain and K is the number of SMC particles used for meta-inference.

RAVI Inference Strategy 2: Agglomerative Clustering

Posterior Approx. `agglom(`X, K`).q()`

  Target of inference : partition Π of dataset X

  Auxiliary variables : merge sequence $mv_{1:|X|-|\Pi|}$

  $\Pi \leftarrow \{\{x\} \mid x \in X\}$ `// Initial partition`

  for $l \in 1, \ldots, |X|$ **do**

    for *unordered pair* $\{i, j\}$ *of clusters in* Π **do**

      $w_{\{i,j\}} \leftarrow \pi((\Pi \setminus \{i, j\}) \cup \{i \cup j\})$

    $w_{\text{stop}} = \pi(\Pi)$

    $All \leftarrow \{\text{stop}\} \cup \{\{i, j\} \mid i, j \in \Pi\}$

    $mv_l \sim \text{Discrete}(\{m \Rightarrow w_m \mid m \in All\})$

    if $mv_l = stop$ **then break**

    $\Pi \leftarrow (\Pi \setminus mv_l) \cup (\cup mv_l)$ `// Perform the merge`

  return Π

Meta-Posterior Approx. `agglom(`X, K`).M(`Π`).q()`

  Target of inference : merge sequence $mv_{1:|X|-|\Pi|}$

  Auxiliary variables : particles $mv^{1:K}_{1:|X|-|\Pi|}$, ancestors $a^{1:K}_{1:|X|-|\Pi|}$

  for $k \in 1, \ldots, K$ **do** $\Pi^k_0, tr_k \leftarrow \{\{x\} \mid x \in X\}, []$

  for $l \in 1, \ldots, |X| - |\Pi|$ **do**

    for $k \in 1, \ldots, K$ **do**

      for *unordered pair* $\{i, j\}$ *in* Π^k_{l-1} **do**

        $w_{\{i,j\}} \leftarrow \pi((\Pi^k_{l-1} \setminus \{i, j\}) \cup \{i \cup j\})$

      $w_{\text{stop}} = \pi(\Pi^k_{l-1})$

      $All \leftarrow \{\text{stop}\} \cup \{\{i, j\} \mid i, j \in \Pi^k_{l-1}\}$

      $Ok \leftarrow \{\{i, j\} \in All \mid \exists c \in \Pi. i \cup j \subseteq c\}$

      $mv_l \sim \text{Discrete}(\{m \Rightarrow w_m \mid m \in Ok\})$

      $\Pi^k_{l-1} \leftarrow (\Pi^k_{l-1} \setminus mv_l) \cup (\cup mv_l)$

      $tr_k \leftarrow [tr_k \ldots, mv_l]$

      $W^k_l \leftarrow \frac{\sum_{m \in Ok} w_m}{\sum_{m \in All} w_m}$

    for $k \in 1, \ldots, K$ **do**

      $a^k_l \sim \text{Discrete}(W^{1:K}_l)$ `// resampling step`

      $\Pi^k_l, tr_k \leftarrow \Pi^{a^k_l}_{l-1}, tr_{a^k_l}$

  return $[tr_1 \ldots, stop]$

Meta-Meta-Posterior `agglom(`X, K`).M(`Π`).M(`$mv_{1:|X|-|\Pi|}$`).q()`

  `// Conditional SMC (omitted for space, but similar to that of rmcvi)`

5.2 AGGLOMERATIVE CLUSTERING FOR DIRICHLET PROCESS MIXTURES

A promising application of RAVI is to transform heuristic randomized algorithms into unbiased and consistent Monte Carlo estimators, by using them as proposal distributions. In this section, we design a RAVI inference strategy for clustering in Dirichlet process mixtures, based on a randomized agglomerative clustering algorithm (Inference Strategy 1).

Datasets and Models. We test our algorithm on three clustering problems. The first is a synthetic 1D dataset sampled from a Dirichlet process (DP) mixture prior. The second is a standard benchmark dataset of galaxy velocities [15, 10], which we model using a DP mixture with Gaussian likelihoods and $\alpha = 1$. The last is a data-cleaning task, correcting typos in 1k strings from Medicare records [23]. We adapt the generative model of Lew et al. [18]. Using an English character-level bigram model $H(s) = h(s_1) \prod_{i=2}^{|s|} h(s_i \mid s_{i-1})$, we model the data $\{y_i\}$ with a DP prior:

$$
\begin{aligned}
G &\sim DP(H, \alpha = 1.0) \\
x_i \mid G &\sim G \\
y_i \mid x_i &\sim f(\cdot \mid x_i)
\end{aligned}
$$

Here, the likelihood $f(y_i \mid x_i)$ models typos. We set f to be

$$
f(y_i \mid x_i) \propto \begin{cases} \mathbf{1}[x_i = y_i] & (x_i, y_i) \notin \mathcal{L} \times \mathcal{L} \\ \frac{\text{NegBin}(\tau(x_i, y_i); \lceil \frac{|s|}{5} \rceil, 0.9)}{(5.09|s|)^{\tau(x_i, y_i)}} & (x_i, y_i) \in \mathcal{L} \times \mathcal{L} \end{cases},
$$

where $\tau(x_i, y_i)$ is the Damerau-Levenshtein edit distance between x_i and y_i, and $\mathcal{L}$ is the set of all observed strings $\{y \mid \exists i. y = y_i\}$.[2] We perform inference in a collapsed version of the model, with the x_i marginalized out:

$$
\begin{aligned}
\Pi &\sim CRP(n = N, \alpha = 1.0) \\
y_I \mid \Pi &\sim F(y_I).
\end{aligned}
$$

Here Π is a partition, I ranges over the components of Π (each of which is a subset of data indices), and $F(y_I) = \sum_{x \in \mathcal{L}} h(x) \prod_{i \in I} f(y_i \mid x)$ is the marginal likelihood of y_I as a sequence of noisy observations of a latent string.

Baseline. We compare to an SMC baseline, inspired by PClean's inference [18], that targets a sequence of posteriors, where the tth posterior incorporates the first t datapoints.

[2] We assume that the data $\mathcal{L}$ includes at least one example of every clean string. When $x_i \in \mathcal{L}$, we model a negative-binomially distributed number of typos, where the number of trials depends on the length of the string.

	$\hat{\mathcal{L}}$
Gaussian likelihood [10], synthetic data	
SMC + adapted proposals	-125.09 ± 0.38
RAVI agglomerative clustering	-125.97 ± 1.62
Gaussian likelihood [10], Galaxy data [15]	
SMC + adapted proposals	-426.20 ± 1.26
RAVI agglomerative clustering	$\mathbf{-423.03 \pm 0.94}$
PClean typos likelihood [18], Hospital data [23]	
SMC + adpated proposals	$-40,239 \pm 1,532$
RAVI agglomerative clustering	$\mathbf{-13,851.0 \pm 0.01}$

Table 2: RAVI agglomerative clustering vs. SMC baseline.

The SMC proposal is locally optimal, assigning the newest datapoint to an existing component I with probability proportional to $\frac{|I|}{t+\alpha-1} \cdot F(y_I \cup \{y_t\})$, or to a new component with probability proportional to $\frac{\alpha}{t+\alpha-1} \cdot F(\{y_t\})$. We do not compare to a Gibbs sampling baseline, as Gibbs sampling does not yield marginal likelihood estimates, but do perform a Gibbs rejuvenation sweep every 20 iterations of SMC.

RAVI algorithm. We apply Algorithm 1 to the inference strategy `agglom` (Inference Strategy 2). The strategy is based on a randomized agglomerative clustering algorithm: each datapoint begins in its own cluster (L1), and we repeatedly choose to either merge two clusters (L9) or stop and propose the current partition (L8). The sequence of merge decisions $mv_1, \ldots, mv_{|X|-|\Pi|}$ are the auxiliary variables of our proposal distribution; the final output is the proposed clustering Π. Our meta-inference $\texttt{agglom}(X, K).\mathcal{M}(\Pi).q$ infers the sequence of merges from the observed clustering Π, using K-particle SMC with proposals that mimic the forward process but choose only from a restricted set *Ok* of possible merges (L8), to avoid making any choices that disagree with Π. SMC introduces additional auxiliary variables, so we also include a conditional SMC meta-meta-posterior approximation (not shown, but nearly identical to `rmcvi`'s).

Results. Table 2 shows average log marginal likelihood estimates; higher is better. On synthetic Gaussian data, the algorithms perform comparably. On the galaxy data, RAVI agglomerative clustering finds modes that SMC misses, leading to a 3-nat improvement in the average log marginal likelihood. In the Medicare data example, SMC misses the ground-truth clustering and hypothesizes many unlikely typos to explain the data. The RAVI agglomerative clustering is less greedy, considering $O(N^2)$ possible merges at each step, rather than $O(N)$. As such, it is able to find the ground truth clustering, correctly identifying all typos (unlike PClean [18], which achieves only 90% accuracy on this dataset) and reporting a log marginal likelihood thousands of nats higher than the SMC algorithm.

6 RELATED WORK AND DISCUSSION

Related work. RAVI builds on and generalizes recent work from both the Monte Carlo and variational inference literatures. For example, Salimans et al. [33] and Ranganath et al. [32] showed how auxiliary variables could be used to construct and optimize variational bounds for specific families of expressive variational approximations (i.e., with intractable marginal densities). Sobolev and Vetrov [36] presented tighter bounds in a more general setting. RAVI is a further generalization, in two directions: first, we show that these bounds arise from particular choices of meta-inference strategy, and can be tightened by improving meta-inference; and second, we extend the results to the Monte Carlo setting, enabling learned variational families to be used as importance sampling, sequential Monte Carlo, or Metropolis-Hastings proposals. We also provide general theorems about the variance of RAVI samplers and the bias of RAVI variational bounds in Section 4, which can be applied to analyze both new and existing algorithms.

RAVI is also related to other compositional or unifying frameworks for thinking about broad classes of inference algorithms [21, 42, 34, 35, 22, 11, 37, 30, 41, 3, 38, 16, 17], some of which involve recursive constructions [27, 12, 13]. However, to our knowledge, RAVI's inference strategies are novel. For example, although RAVI and Nested IS (NIS) [27] are both approaches to inference with 'intractable proposals,' NIS *approximately samples* a proposal distribution with a *tractable* (unnormalized) density, whereas RAVI *approximates the density* of a proposal that *can* be simulated tractably, but whose marginal density (even unnormalized) is intractable. As another example, Domke and Sheldon [13]'s framework of *estimator-coupling pairs* constructs variational bounds and marginal likelihood estimators recursively, but unlike in RAVI, the posterior approximations cannot be used to formulate objectives for *amortized* VI or as components of Metropolis-Hastings proposals.

Outlook and Limitations. RAVI expands the design space for Monte Carlo and variational inference. It gives unifying correctness proofs for over a dozen methods from the literature, and novel theorems that characterize their behavior. Experiments show that RAVI helps to design algorithms that significantly improve accuracy over previously introduced Monte Carlo and variational inference methods. However, some difficulties remain. For example, the score function gradient estimators we present (Algs. 3 and 4) have high variance for some strategies $\mathcal{S}$; in Appendix E, we give estimators that exploit the reparameterization trick, but they only help when the posterior approximations in $\mathcal{S}$ can be reparameterized, which is not the case, e.g., for SMC-based approximations. In these cases, RAVI can still be used to derive objectives for optimization, but practitioners will need other ways of reducing the variance of gradient estimates; many results from the literature [24, 39] should apply.

Another difficulty is that the implementation of RAVI algorithms can be complex. We are currently exploring an automated implementation strategy based on probabilistic programming [11, 40]: if each posterior and meta-posterior approximation in a RAVI strategy $\mathcal{S}$ is implemented as a probabilistic program, Algorithms 1-4 can be provided as higher-order functions, which automatically compute the necessary joint densities and gradients.

References

[1] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

[2] Christophe Andrieu, Arnaud Doucet, Sinan Yıldırım, and Nicolas Chopin. On the utility of metropolis-hastings with asymmetric acceptance ratio. *arXiv preprint arXiv:1803.09527*, 2018.

[3] Christophe Andrieu, Anthony Lee, and Sam Livingstone. A general perspective on the metropolis-hastings kernel. *arXiv preprint arXiv:2012.14881*, 2020.

[4] Philip Bachman and Doina Precup. Training deep generative models: Variations on a theme. In *NIPS Approximate Inference Workshop*, 2015.

[5] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[6] Sourav Chatterjee, Persi Diaconis, et al. The sample size required in importance sampling. *The Annals of Applied Probability*, 28(2):1099–1135, 2018.

[7] Nicolas Chopin and Omiros Papaspiliopoulos. *An introduction to sequential Monte Carlo*. Springer, 2020.

[8] Nicolas Chopin, Pierre E Jacob, and Omiros Papaspiliopoulos. Smc2: an efficient algorithm for sequential analysis of state space models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):397–426, 2013.

[9] Chris Cremer, Quaid Morris, and David Duvenaud. Reinterpreting importance-weighted autoencoders. *arXiv preprint arXiv:1704.02916*, 2017.

[10] Marco F Cusumano-Towner and Vikash K Mansinghka. Aide: An algorithm for measuring the accuracy of probabilistic inference algorithms. *arXiv preprint arXiv:1705.07224*, 2017.

[11] Marco F Cusumano-Towner, Feras A Saad, Alexander K Lew, and Vikash K Mansinghka. Gen: a general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 221–236, 2019.

[12] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

[13] Justin Domke and Daniel Sheldon. Divide and couple: Using monte carlo variational objectives for posterior approximation. *arXiv preprint arXiv:1906.10115*, 2019.

[14] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3): 197–208, 2000.

[15] Michael J Drinkwater, Quentin A Parker, Dominique Proust, Eric Slezak, and Hernán Quintana. The large scale distribution of galaxies in the shapley supercluster. *Publications of the Astronomical Society of Australia*, 21(1):89–96, 2004.

[16] Axel Finke. *On extended state-space constructions for Monte Carlo methods*. PhD thesis, University of Warwick, 2015.

[17] Axel Finke and Alexandre H Thiery. On importance-weighted autoencoders. *arXiv preprint arXiv:1907.10477*, 2019.

[18] Alexander Lew, Monica Agrawal, David Sontag, and Vikash Mansinghka. Pclean: Bayesian data cleaning at scale with domain-specific probabilistic programming. In *International Conference on Artificial Intelligence and Statistics*, pages 1927–1935. PMLR, 2021.

[19] Fredrik Lindsten, Adam M Johansen, Christian A Naesseth, Bonnie Kirkpatrick, Thomas B Schön, JAD Aston, and Alexandre Bouchard-Côté. Divide-and-conquer with sequential monte carlo. *Journal of Computational and Graphical Statistics*, 26(2):445–458, 2017.

[20] Jun S Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

[21] Vikash Mansinghka, Daniel Selsam, and Yura Perov. Venture: a higher-order probabilistic programming platform with programmable inference. *arXiv preprint arXiv:1404.0099*, 2014.

[22] Vikash K Mansinghka, Ulrich Schaechtle, Shivam Handa, Alexey Radul, Yutian Chen, and Martin Rinard. Probabilistic programming with programmable inference. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 603–616, 2018.

[23] Medicare. Hospital compare, 2012.

[24] Andriy Mnih and Danilo Rezende. Variational inference for monte carlo objectives. In *International Conference on Machine Learning*, pages 2188–2196. PMLR, 2016.

[25] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[26] Saeid Naderiparizi, Adam Ścibior, Andreas Munk, Mehrdad Ghadiri, Atılım Güneş Baydin, Bradley Gram-Hansen, Christian Schroeder de Witt, Robert Zinkov, Philip H. S. Torr, Tom Rainforth, Yee Whye Teh, and Frank Wood. Amortized rejection sampling in universal probabilistic programming, 2019.

[27] Christian Naesseth, Fredrik Lindsten, and Thomas Schon. Nested sequential monte carlo methods. In *International Conference on Machine Learning*, pages 1292–1301. PMLR, 2015.

[28] Christian Naesseth, Scott Linderman, Rajesh Ranganath, and David Blei. Variational sequential monte carlo. In *International Conference on Artificial Intelligence and Statistics*, pages 968–977. PMLR, 2018.

[29] Radford M Neal. Annealed importance sampling. *Statistics and computing*, 11(2):125–139, 2001.

[30] Kirill Neklyudov, Max Welling, Evgenii Egorov, and Dmitry Vetrov. Involutive mcmc: a unifying framework. In *International Conference on Machine Learning*, pages 7273–7282. PMLR, 2020.

[31] Michael A Newton and Adrian E Raftery. Approximate bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(1):3–26, 1994.

[32] Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *International conference on machine learning*, pages 324–333. PMLR, 2016.

[33] Tim Salimans, Diederik Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226. PMLR, 2015.

[34] Adam Ścibior, Ohad Kammar, Matthijs Vákár, Sam Staton, Hongseok Yang, Yufei Cai, Klaus Ostermann, Sean K Moss, Chris Heunen, and Zoubin Ghahramani. Denotational validation of higher-order bayesian inference. *arXiv preprint arXiv:1711.03219*, 2017.

[35] Adam Ścibior, Ohad Kammar, and Zoubin Ghahramani. Functional programming for modular bayesian inference. *Proceedings of the ACM on Programming Languages*, 2(ICFP):1–29, 2018.

[36] Artem Sobolev and Dmitry P Vetrov. Importance weighted hierarchical variational inference. *Advances in Neural Information Processing Systems*, 32, 2019.

[37] Sam Stites, Heiko Zimmermann, Hao Wu, Eli Sennesh, et al. Learning proposals for probabilistic programs with inference combinators. *arXiv preprint arXiv:2103.00668*, 2021.

[38] Geir Storvik. On the flexibility of metropolis–hastings acceptance probabilities in auxiliary variable proposal generation. *Scandinavian Journal of Statistics*, 38(2): 342–358, 2011.

[39] George Tucker, Dieterich Lawson, Shixiang Gu, and Chris J Maddison. Doubly reparameterized gradient estimators for monte carlo objectives. *arXiv preprint arXiv:1810.04152*, 2018.

[40] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An introduction to probabilistic programming. *arXiv preprint arXiv:1809.10756*, 2018.

[41] Heiko Zimmermann, Hao Wu, Babak Esmaeili, and Jan-Willem van de Meent. Nested variational inference. *arXiv preprint arXiv:2106.11302*, 2021.

[42] Robert Zinkov and Chung-chieh Shan. Composing inference algorithms as program transformations. *arXiv preprint arXiv:1603.01882*, 2016.

SUPPLEMENTARY MATERIAL FOR "RECURSIVE MONTE CARLO AND VARIATIONAL INFERENCE WITH AUXILIARY VARIABLES"

This document and the accompanying code files contain supplementary material for the submission "Recursive Monte Carlo and Variational Inference with Auxiliary Variables." In particular, we provide:

1. In Section A, **proofs** of Theorems 1-4.
2. In Section B, **RAVI inference strategies for many existing algorithms**.
3. In Section C, a further discussion of the **absolute continuity requirements** for RAVI and how they can be relaxed.
4. In Section D, **other applications** of RAVI inference strategies, to parameterize rejection sampling and KL divergence estimation algorithms.

NOTATION

Throughout the appendix, we use a notation that fully *unrolls* an inference strategy into a sequence $p_0, p_1, \ldots, p_n$, where $p_0(x_1, x_0)$ is the model, $p_1(x_2, x_1; x_0)$ is $\mathcal{S}.q$, $p_2(x_3, x_2; x_1, x_0)$ is $\mathcal{S}.\mathcal{M}(x_1).q$, and so on. We call n the *length* of an inference strategy.

A OMITTED PROOFS.

A.1 PROOF OF THEOREM 1.

Theorem 1. *If p_0 is a model and $p_1, \ldots, p_n$ encodes an inference strategy, then Alg. 1 is properly weighted for $p_0(x_1 \mid x_0)$. Alg. 2 generates $x_0 \sim p_0$ and a weight $\frac{1}{Z}$ such that $\mathbb{E}[\frac{1}{Z} \mid x_0] = \frac{1}{p_0(x_0)}$.*

Proof. The proof is by induction on the length of the inference strategy.

First consider $n = 0$. In this case Algorithm 2 terminates on its first step, producing $x_0 \sim p_0$ and returning $\frac{1}{p_0(x_0)}$ exactly. For Algorithm 1, as there is no latent variable x_1, we view x_1 as deterministically equal to a null value $\emptyset$. In this case, Algorithm 1 returns $(x_1 = \emptyset, w = p_0(x_0))$ on its first step, and $\mathbb{E}[wf(x_1)] = p_0(x_0) \cdot f(\emptyset)$ as required.

Now consider the inductive step. Assume the theorem proven for inference strategies of length $n < k$ and consider an inference strategy of length k.

We first consider Algorithm 2. The variable x_0 is generated from p_0 in Step 2, so we have directly that $x_0 \sim p_0$. By induction, the call to Algorithm 1 generates a weight w' whose expectation (given x_0 and x_1) is $p_1(x_1; x_0)$. Then the conditional expectation of $w = \frac{w'}{p_0(x_1, x_0)}$, given x_0, is:

$$\mathbb{E}_{x_1,\ldots,x_k}[w \mid x_0] = \mathbb{E}_{x_1}\left[\frac{\mathbb{E}_{x_2,\ldots,x_k}[w' \mid x_1, x_0]}{p_0(x_1, x_0)} \;\middle|\; x_0\right] = \mathbb{E}_{x_1}\left[\frac{p_1(x_1; x_0)}{p_0(x_1, x_0)} \;\middle|\; x_0\right].$$

This last expression can be viewed as an ordinary importance sampling estimator, with proposal distribution $p_0(x_1 \mid x_0)$ and target $\frac{p_1(x_1;x_0)}{p_0(x_0)}$:

$$\mathbb{E}_{x_1}\left[\frac{p_1(x_1; x_0)}{p_0(x_1, x_0)} \;\middle|\; x_0\right] = \int p_0(x_1 \mid x_0) \cdot \frac{p_1(x_1; x_0)}{p_0(x_0)p_0(x_1 \mid x_0)} dx_1 = \int \frac{p_1(x_1; x_0)}{p_0(x_0)} dx_1 = \frac{1}{p_0(x_0)}.$$

The last equations use the absolute continuity assumption: the integral is only over the support of $p_0(x_1 \mid x_0)$, and so for $\int p_1(x_1; x_0) dx_1$ to equal 1, we must have that $p_1(x_1; x_0) \ll p_0(x_1 \mid x_0)$.

We now turn to Algorithm 1. We have, for any measurable test function f:

$$\begin{aligned}
\mathbb{E}\left[w \cdot p_0(x_1, x_0) \cdot f(x_1)\right] &= \mathbb{E}_{x_1 \sim p_1(x_1;x_0)}\left[\mathbb{E}\left[w \;\middle|\; x_1\right] \cdot p_0(x_1, x_0) \cdot f(x_1)\right] \\
&= \mathbb{E}_{x_1 \sim p_1(x_1;x_0)}\left[\frac{p_0(x_1, x_0)}{p_1(x_1; x_0)} \cdot f(x_1)\right] \\
&= p_0(x_0) \cdot \mathbb{E}_{x_1 \sim p_0(x_1 \mid x_0)}[f(x_1)],
\end{aligned}$$

where the second equation holds due to the proof for Algorithm 2 and the third is the ordinary importance sampling argument.

A.2 PROOF OF THEOREM 2

Lemma A.1. *For an inference strategy $p_0, \ldots, p_n$:*

$$\mathcal{L}_0(x_0) = \mathbb{E}_{x_1 \sim p_1(\cdot; x_0)}[\log p_0(x_0, x_1) - \mathcal{U}_1(x_1; x_0)]$$

(where the dependence of each distribution in the inference strategy $p_1, \ldots, p_n$ on the value x_0 manifests as an additional parameter in $\mathcal{U}_1$). Similarly, for a valid inference strategy $p_0, \ldots, p_n$:

$$\mathcal{U}_0(x_0) = \mathbb{E}_{x_1 \sim p_0(\cdot|x_0)}[\log p_0(x_0, x_1) - \mathcal{L}_1(x_1; x_0)]$$

Proof.

$$\begin{aligned}
\mathcal{L}_0(x_0) &= \mathbb{E}_{x_1 \sim p_1(\cdot; x_0)}[\mathbb{E}_{p_1, p_3, \ldots}[\log \hat{Z}]] && (1)\\
&= \mathbb{E}_{x_1 \sim p_1(\cdot; x_0)}\left[\mathbb{E}_{p_1, p_3, \ldots}\left[\sum_{i=0}^{n-1} (-1)^i \log p_i(x_{i+1}, x_i; x_{0:i-1}) + (-1)^n \log p_n(x_n; x_{0:n-1})\right]\right] && (2)\\
&= \mathbb{E}_{x_1 \sim p_1(\cdot; x_0)}\left[\log p_0(x_0, x_1) - \mathbb{E}_{p_1, p_3, \ldots}\left[\sum_{i=1}^{n-1} (-1)^{i-1} \log p_i(x_{i+1}, x_i; x_{0:i-1}) + (-1)^{n-1} \log p_n(x_n; x_{0:n-1})\right]\right] && (3)\\
&= \mathbb{E}_{x_1 \sim p_1(\cdot; x_0)}[\log p_0(x_0, x_1) - \mathcal{U}_1(x_1; x_0)]. && (4)
\end{aligned}$$

The same approach, but with $\mathbb{E}[\log \check{Z}]$, can be used to prove the other conclusion.

Theorem 2. *For the inference strategy $p_0, \ldots, p_n$, Alg. 3 yields unbiased estimates of $\mathcal{L}_0(x_0)$ and of $\nabla_\theta \mathcal{L}_0(x_0)$; and Alg. 4 yields (i) $x_0 \sim p_0$, (ii) $\hat{U}$ such that $\mathbb{E}[\hat{U}|x_0] = \mathcal{U}_0(x_0)$, (iii) $\widehat{\nabla_\theta}$ such that $\mathbb{E}_{x_0 \sim p_0}[\mathbb{E}[\widehat{\nabla_\theta}|x_0]] = \nabla_\theta \mathbb{E}_{x_0 \sim p_0}[\mathcal{U}_0(x_0)]$, and (iv) a value $\mathbf{g}$ such that for any function R that does not depend on θ, $\mathbb{E}_{x_0 \sim p_0}[\mathbb{E}[\mathbf{g} \cdot R(x_0)|x_0]] = \nabla_\theta \mathbb{E}_{x_0 \sim p_0}[R(x_0)]$ if $\nabla_\theta \mathbb{E}_{x_0 \sim p_0}[R(x_0)]$ is defined.*

Proof. The proof is by induction on the length of the inference strategy.

First consider $n = 0$. In this case Algorithm 3 returns $\log p_n(x_n) = \mathcal{L}_n(x_n)$ and $\nabla_\theta \log p_n(x_n) = \nabla_\theta \mathcal{L}_n(x_n)$, which are zero-variance unbiased estimates. In this case Algorithm 4 returns (i) x_n with $x_n \sim p_n$, (ii) $\hat{U} = \log p_n(x_n) = \mathcal{U}_n(x_n)$ (for which $\mathbb{E}[\hat{U}|x_n] = \mathcal{U}_n(x_n)$), (iii) $\widehat{\nabla_\theta} = \nabla_\theta \log p_n(x_n) + (\log p_n(x_n)) \cdot \mathcal{U}_n(x_n)$, and (iv) $\mathbf{g} = \log p_n(x_n)$. The condition for $\widehat{\nabla_\theta}$ is satisfied because:

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{x_n \sim p_n}[\mathcal{U}_n(x_n)] &= \mathbb{E}_{x_n \sim p_n}[(\nabla_\theta \log p_n(x_n)) \cdot \mathcal{U}_n(x_n) + \nabla_\theta \mathcal{U}_n(x_n)]\\
&= \mathbb{E}_{x_n \sim p_n}[(\nabla_\theta \log p_n(x_n)) \cdot \mathcal{U}_n(x_n) + \nabla_\theta \log p_n(x_n)]\\
&= \mathbb{E}_{x_n \sim p_n}[\widehat{\nabla_\theta}]
\end{aligned}$$

The condition for $\mathbf{g}$ is satisfied because

$$\nabla_\theta \mathbb{E}_{x_n \sim p_n}[R(x_n)] = \mathbb{E}_{x_n \sim p_n}[(\nabla_\theta \log p_n(x_n)) R(x_n)] = \mathbb{E}_{x_n \sim p_n}[\mathbf{g} \cdot R(x_n)]$$

Now consider the inductive step. Assume the theorem is proven for inference strategies of length $n < k$ and now consider an inference strategy of length k. We first consider Algorithm 3. It samples $x_1 \sim p_1$, and by induction, the call to Algorithm 4 returns $(\hat{U}, \widehat{\nabla_\theta}, \mathbf{g})$ such that:

1. $\mathbb{E}[\hat{U}|x_1] = \mathcal{U}_1(x_1; x_0)$
2. $\mathbb{E}_{x_1 \sim p_1}[\mathbb{E}[\widehat{\nabla_\theta}|x_1]] = \nabla_\theta \mathbb{E}_{x_1 \sim p_1}[\mathcal{U}_1(x_1; x_0)]$

3. $\mathbb{E}_{x_1\sim p_1}[\mathbb{E}[\mathbf{g}\cdot R(x_1)|x_1]] = \nabla_\theta \mathbb{E}_{x_1\sim p_1}[R(x_1)]$ for all valid R.

Consider $\mathbb{E}[\hat{L}]$:

$$\begin{aligned}
\mathbb{E}[\hat{L}] &= \mathbb{E}_{x_1\sim p_1}[\mathbb{E}[\hat{L}|x_1]] && (5)\\
&= \mathbb{E}_{x_1\sim p_1}[\mathbb{E}[\log p_0(x_0,x_1) - \hat{U}|x_1]] && (6)\\
&= \mathbb{E}_{x_1\sim p_1}[\log p_0(x_0,x_1) - E[\hat{U}|x_1]] && (7)\\
&= \mathbb{E}_{x_1\sim p_1}[\log p_0(x_0,x_1) - \mathcal{U}_1(x_1)] && (8)\\
&= \mathcal{L}_0(x_0) && (9)
\end{aligned}$$

Consider $\mathbb{E}[\widehat{\nabla_\theta}']$:

$$\begin{aligned}
\mathbb{E}[\widehat{\nabla_\theta}'] &= \mathbb{E}\left[\nabla_\theta \log p_0(x_0,x_1) + \mathbf{g}\cdot \log p_0(x_0,x_1) - \widehat{\nabla_\theta}\right] && (10)\\
&= \mathbb{E}_{x_1\sim p_1}\left[\nabla_\theta \log p_0(x_0,x_1)\right] + \nabla_\theta \mathbb{E}_{x_1\sim p_1}[\log \boxed{p_0}(x_0,x_1)] - \mathbb{E}_{x_1\sim p_1}\left[\mathbb{E}[\widehat{\nabla_\theta}|x_1]\right] && (11)\\
&= \mathbb{E}_{x_1\sim p_1}\left[\nabla_\theta \log p_0(x_0,x_1)\right] + \nabla_\theta \mathbb{E}_{x_1\sim p_1}[\log \boxed{p_0}(x_0,x_1)] - \nabla_\theta \mathbb{E}_{x_1\sim p_1}\left[\mathcal{U}_1(x_1;x_0)\right] && (12)\\
&= \nabla_\theta \mathbb{E}_{x_1\sim p_1}[\log p_0(x_0,x_1) - \mathcal{U}_1(x_1;x_0)] && (13)\\
&= \nabla_\theta \mathcal{L}_0(x_0) && (14)
\end{aligned}$$

(where $\boxed{p_0}$ denotes the distribution p_0 but without a dependence on θ for the purposes of differentiation with respect to θ).

Now, consider Algorithm 4. By induction, the call to Algorithm 3 returns $(\hat{L}, \widehat{\nabla_\theta})$ such that:

1. $\mathbb{E}[\hat{L}] = \mathcal{L}_1(x_1;x_0)$
2. $\mathbb{E}[\widehat{\nabla_\theta}] = \nabla_\theta \mathcal{L}_1(x_1;x_0)$

The property $x_0 \sim p_0$ is immediately read from Algorithm 4. Consider $\mathbb{E}[\hat{U}|x_0]$:

$$\begin{aligned}
\mathbb{E}[\hat{U}|x_0] &= \mathbb{E}_{x_1\sim p_1(\cdot|x_0)}[\mathbb{E}[\log p_0(x_0,x_1) - \hat{L}|x_0,x_1]] && (15)\\
&= \mathbb{E}_{x_1\sim p_1(\cdot|x_0)}[\log p_0(x_0,x_1) - \mathcal{L}_1(x_1;x_0)] && (16)\\
&= \mathcal{U}_0(x_0) && (17)
\end{aligned}$$

Consider $\mathbb{E}_{x_0\sim p_0}[\mathbb{E}[\widehat{\nabla_\theta}'|x_0]]$:

$$\begin{aligned}
&= \mathbb{E}_{x_0,x_1\sim p_0}\left[\mathbb{E}\left[\nabla_\theta \log p_0(x_0,x_1) + (\nabla_\theta \log p_0(x_0,x_1))\cdot \hat{U} - \widehat{\nabla_\theta}|x_0,x_1\right]\right] && (18)\\
&= \mathbb{E}_{x_0,x_1\sim p_0}\left[\nabla_\theta \log p_0(x_0,x_1) + (\nabla_\theta \log p_0(x_0,x_1))\cdot \mathbb{E}\left[\hat{U}|x_0,x_1\right] - \mathbb{E}\left[\widehat{\nabla_\theta}|x_0,x_1\right]\right] && (19)\\
&= \mathbb{E}_{x_0,x_1\sim p_0}\left[\nabla_\theta \log p_0(x_0,x_1) + (\nabla_\theta \log p_0(x_0,x_1))\cdot \mathbb{E}\left[\log p_0(x_0,x_1) - \hat{L}|x_0,x_1\right] - \nabla_\theta \mathcal{L}_1(x_1;x_0)\right] && (20)\\
&= \mathbb{E}_{x_0,x_1\sim p_0}\left[\nabla_\theta \log p_0(x_0,x_1) - \nabla_\theta \mathcal{L}_1(x_1;x_0) + (\nabla_\theta \log p_0(x_0,x_1))\cdot(\log p_0(x_0,x_1) - \mathcal{L}_1(x_1;x_0))\right] && (21)\\
&= \nabla_\theta \mathbb{E}_{x_0,x_1\sim p_0}\left[\log p_0(x_0,x_1) - \mathcal{L}_1(x_1;x_0)\right] && (22)\\
&= \nabla_\theta \mathbb{E}_{x_0\sim p_0}[\mathbb{E}_{x_1\sim p_0(\cdot|x_0)}[\log p_0(x_0,x_1) - \mathcal{L}_1(x_1;x_0]] && (23)\\
&= \nabla_\theta \mathbb{E}_{x_0\sim p_0}[\mathcal{U}_0(x_0)] && (24)
\end{aligned}$$

Consider $\mathbb{E}_{x_0\sim p_0}[\mathbb{E}[\mathbf{g}\cdot R(x_0)|x_0]]$ (and recall that $R(x_0)$ is not to be treated as a function of θ):

$$\begin{aligned}
\mathbb{E}_{x_0\sim p_0}[\mathbb{E}[\mathbf{g}\cdot R(x_0)|x_0]] &= \mathbb{E}_{x_0\sim p_0}\left[\mathbb{E}[(\nabla_\theta \log p_0(x_0,x_1))\cdot R(x_0)|x_0]\right] && (25)\\
&= \mathbb{E}_{x_0,x_1\sim p_0}\left[(\nabla_\theta \log p_0(x_0,x_1))\cdot R(x_0)\right] && (26)\\
&= \nabla_\theta \mathbb{E}_{x_0,x_1\sim p_0}\left[R(x_0)\right] && (27)\\
&= \nabla_\theta \mathbb{E}_{x_0\sim p_0}[R(x_0)] && (28)
\end{aligned}$$

A.3 PROOF OF THEOREM 3

Theorem 3. *Let $p_0, \dots, p_n$ be an inference strategy. As above, let $Z = p_0(x_0)$ for some fixed observation x_0, and let $\hat{Z}$ denote the recursive estimator for Z corresponding to the inference strategy $p_0, \dots, p_n$. Further, let $\hat{Z}_i$ be the estimator corresponding to the inference strategy $p_0, \dots, p_i$ (which will typically be intractable to compute, as it requires a marginal density for $p_i(x_i)$). Then $\frac{\hat{Z}}{Z}$ has variance*

$$\mathrm{Var}\left(\frac{\hat{Z}}{Z}\right) = \sum_{i=0}^{n-1} \mathbb{E}\left[\frac{\hat{Z}_i^2}{Z^2} \cdot \chi^2(p_{2\lceil \frac{i}{2} \rceil}(x_{i+1} \mid x_i, \dots, x_0) \mid\mid p_{2\lfloor \frac{i}{2} \rfloor + 1}(x_{i+1} \mid x_i, \dots, x_0))\right],$$

where the expectation is taken over $x_1, \dots, x_i$.

Proof. The proof is by induction on the length of the inference strategy.

When $n = 0$, $\frac{\hat{Z}}{Z} = 1$ deterministically, and so has variance $\sum_{i=0}^{-1}(\dots) = 0$, as desired.

Now assume the theorem proven for $n = k - 1$ and consider $n = k$. Note that $\hat{Z} = \hat{Z}_k = \hat{Z}_{k-1} \cdot \frac{p_{2\lfloor \frac{k}{2} \rfloor}(x_k \mid x_{0:k-1})}{p_{2\lceil \frac{k}{2} \rceil - 1}(x_k \mid x_{0:k-1})}$. Therefore, we have:

$$\begin{aligned}
\mathrm{Var}\left(\frac{\hat{Z}}{Z}\right) &= \mathbb{E}\left[\frac{\hat{Z}^2}{Z^2}\right] - \mathbb{E}\left[\frac{\hat{Z}}{Z}\right]^2 \\
&= \mathbb{E}\left[\frac{\hat{Z}^2}{Z^2} - 1\right] \\
&= \mathbb{E}\left[\frac{\hat{Z}_{k-1}^2}{Z^2}\left(\frac{p_{2\lfloor \frac{k}{2} \rfloor}(x_k \mid x_{0:k-1})^2}{p_{2\lceil \frac{k}{2} \rceil - 1}(x_k \mid x_{0:k-1})^2}\right) - 1\right] \\
&= \mathbb{E}\left[\frac{\hat{Z}_{k-1}^2}{Z^2}\left(\mathbb{E}_{x_k \sim p_{2\lceil \frac{k}{2} \rceil - 1}}\left[\frac{p_{2\lfloor \frac{k}{2} \rfloor}(x_k \mid x_{0:k-1})^2}{p_{2\lceil \frac{k}{2} \rceil - 1}(x_k \mid x_{0:k-1})^2} \,\middle|\, x_{0:k-1}\right]\right) - 1\right] \\
&= \mathbb{E}\left[\frac{\hat{Z}_{k-1}^2}{Z^2}\left(\chi^2(p_{2\lfloor \frac{k}{2} \rfloor}(x_k \mid x_{0:k-1}) \,||\, p_{2\lceil \frac{k}{2} \rceil - 1}(x_k \mid x_{0:k-1})) + 1\right) - 1\right] \\
&= \mathbb{E}\left[\frac{\hat{Z}_{k-1}^2}{Z^2}\left(\chi^2(p_{2\lfloor \frac{k}{2} \rfloor}(x_k \mid x_{0:k-1}) \,||\, p_{2\lceil \frac{k}{2} \rceil - 1}(x_k \mid x_{0:k-1}))\right) + \frac{\hat{Z}_{k-1}^2}{Z^2} - 1\right] \\
&= \mathbb{E}\left[\frac{\hat{Z}_{k-1}^2}{Z^2}\left(\chi^2(p_{2\lfloor \frac{k}{2} \rfloor}(x_k \mid x_{0:k-1}) \,||\, p_{2\lceil \frac{k}{2} \rceil - 1}(x_k \mid x_{0:k-1}))\right)\right] + \mathrm{Var}\left(\frac{\hat{Z}_{k-1}}{Z}\right) \\
&= \sum_{i=0}^{k-1} \mathbb{E}\left[\frac{\hat{Z}_i^2}{Z^2} \cdot \chi^2(p_{2\lceil \frac{i}{2} \rceil}(x_{i+1} \mid x_i, \dots, x_0) \mid\mid p_{2\lfloor \frac{i}{2} \rfloor + 1}(x_{i+1} \mid x_i, \dots, x_0))\right],
\end{aligned}$$

where the last step uses the inductive hypothesis to rewrite $\mathrm{Var}(\frac{\hat{Z}_{k-1}}{Z})$ as a sum from $i = 0$ to $k - 2$.

A.4 PROOF OF THEOREM 4.

Theorem 4. *If $p_0, \dots, p_n$ is an inference strategy, then*

$$\mathcal{U}_0(x_0) = \log p_0(x_0) + \sum_{i=0}^{n-1} \mathbb{E}_{x_{1:i-1} \sim p^*(\cdot \mid x_0)}\left[\mathrm{KL}(p_{2\lceil \frac{i}{2} \rceil}(x_{i+1} \mid x_{0:i}) \mid\mid p_{2\lfloor \frac{i}{2} \rfloor + 1}(x_{i+1} \mid x_{0:i}))\right],$$

and if $p_0, \dots, p_n$ is an inference strategy, then

$$\mathcal{L}_0(x_0) = \log p_0(x_0) - \sum_{i=0}^{n-1} \mathbb{E}_{x_{1:i-1} \sim q^*(\cdot; x_0)}\left[\mathrm{KL}(p_{2\lfloor \frac{i}{2} \rfloor + 1}(x_{i+1} \mid x_{0:i}) \mid\mid p_{2\lceil \frac{i}{2} \rceil}(x_{i+1} \mid x_{0:i}))\right].$$

Proof.

When $n = 0$, we have $\mathcal{L}_0(x_0) = \log p_0(x_0) = \mathcal{U}_0(x_0)$, as required. Now consider $n > 0$. If we have an inference strategy, we have by Lemma A.1 that

$$\mathcal{L}_0(x_0) = \mathbb{E}_{x_1 \sim p_1(\cdot; x_0)}[\log p_0(x_0, x_1) - \mathcal{U}_1(x_1; x_0)].$$

Since $p_0, \ldots, p_n$ is an inference strategy, $p_1, \ldots, p_n$ is an inference strategy, and by induction,

$$\mathcal{U}_1(x_1; x_0) = \log p_1(x_1; x_0) + \sum_{i=1}^{n-1} \mathbb{E}_{x_{1:i-1} \sim q^*(\cdot; x_0)} \left[\mathrm{KL}(p_{2\lfloor \frac{i}{2} \rfloor + 1}(x_{i+1} \mid x_{0:i}) \mid\mid p_{2\lceil \frac{i}{2} \rceil}(x_{i+1} \mid x_{0:i})) \right].$$

Substituting, we get

$$\begin{aligned}
\mathcal{L}_0(x_0) &= \mathbb{E}_{x_1 \sim p_1(\cdot; x_0)} \left[\log p_0(x_0, x_1) - \log p_1(x_1; x_0) - \sum_{i=1}^{n-1} \mathbb{E} \left[\mathrm{KL}(p_{2\lfloor \frac{i}{2} \rfloor + 1}(x_{i+1} \mid x_{0:i}) \mid\mid p_{2\lceil \frac{i}{2} \rceil}(x_{i+1} \mid x_{0:i})) \right] \right] \\
&= \log p_0(x_0) + \mathbb{E}_{x_1 \sim p_1(\cdot; x_0)} \left[\log \frac{p_0(x_1 \mid x_0)}{p_1(x_1; x_0)} - \sum_{i=1}^{n-1} \mathbb{E} \left[\mathrm{KL}(p_{2\lfloor \frac{i}{2} \rfloor + 1}(x_{i+1} \mid x_{0:i}) \mid\mid p_{2\lceil \frac{i}{2} \rceil}(x_{i+1} \mid x_{0:i})) \right] \right] \\
&= \log p_0(x_0) - \mathrm{KL}(p_1(x_1; x_0) || p_0(x_1 \mid x_0)) - \sum_{i=1}^{n-1} \mathbb{E} \left[\mathrm{KL}(p_{2\lfloor \frac{i}{2} \rfloor + 1}(x_{i+1} \mid x_{0:i}) \mid\mid p_{2\lceil \frac{i}{2} \rceil}(x_{i+1} \mid x_{0:i})) \right] \\
&= \log p_0(x_0) - \sum_{i=0}^{n-1} \mathbb{E} \left[\mathrm{KL}(p_{2\lfloor \frac{i}{2} \rfloor + 1}(x_{i+1} \mid x_{0:i}) \mid\mid p_{2\lceil \frac{i}{2} \rceil}(x_{i+1} \mid x_{0:i})) \right].
\end{aligned}$$

The same proof applies to inference strategies, flipping the necessary signs.

A.5 STATIONARITY OF MCMC ALGORITHM

In Section 3, we mention that RAVI can be used to run Metropolis-Hastings kernels with proposals that have intractable densities. Here, we present and justify the algorithm.

Let $\tilde{\pi}(x) = \int \tilde{\pi}(r, x) \mathrm{d}r = Z \int \pi(r, x) \mathrm{d}r$ be a possibly unnormalized target density, and let $q(x'; x) = \int q(s, x'; x) \mathrm{d}s$ be a proposal kernel mapping previous state x to new state x'. We note that (1) both $\tilde{\pi}$ and q have intractable marginal densities, and (2) the target marginal $\tilde{\pi}(x)$ itself may be unnormalized. As is typical in pseudomarginal MCMC, even this unnormalized target density cannot be evaluated pointwise, due to the additional nuisance variables r.

Now suppose we have an inference strategy $\mathcal{S}$ targeting $\pi(r \mid x)$, and a family of inference strategies $\mathcal{M}(x, x')$ targeting $q(s \mid x'; x)$. Let x be a starting position for our Markov chain. We can run Algorithm 1 on $\mathcal{S}$, targeting $\pi(r \mid x)$, to obtain an initial estimate $\hat{Z}_x$ of the unnormalized marginal density $\tilde{\pi}(x)$. Then Algorithm 5 defines a stationary MCMC kernel for the target distribution $\pi(x)$, starting at input point x:

When q's marginal density is known exactly, the above algorithm recovers variants of Particle-Marginal MH [1], except instead of using SMC to marginalize r, any RAVI algorithm can be applied. When q's marginal density is unavailable, however, the algorithm instead becomes a pseudo-marginal *ratio* algorithm [2], because not just p but also q is estimated unbiasedly. In general, it is *not* valid to use arbitrary unbiased estimates of p *and* q, or even of $\alpha = \frac{p(x')q(x;x')}{p(x)q(x';x)}$, within an MH algorithm. However, the added structure of the RAVI strategy ensures that the above procedure is sound.

To see why our MCMC kernel is stationary, consider the extended target distribution

$$p_0(x_0) \cdot \prod_{i=1}^{n} p_{2\lfloor \frac{i}{2} \rfloor}(x_i \mid x_{0:i-1}) \cdot \prod_{i=0}^{m} q_{2\lfloor \frac{i}{2} \rfloor}(y_i \mid y_{0:i-1}; x_0) \cdot \prod_{i=1}^{m} q_{2\lceil \frac{i}{2} \rceil - 1}(y'_i; y'_{1:i-1}, x_0; y_0).$$

Our algorithm can be understood as sequencing two stationary kernels for this target (which admits our target of interest, p_0, as a marginal). The first is a Gibbs update on the y_i and y'_i variables: the y_i variables are generated while running

Algorithm 5: RAVI Metropolis-Hastings
Input: model $\tilde{\pi}(x) = Z \int \pi(r, x) \mathrm{d}r$
Input: proposal $q(x'; x) = \int q(s, x'; x) \mathrm{d}s$
Input: inference strategy $\mathcal{S}$ targeting $\pi(r \mid x)$
Input: family $\mathcal{M}(x, x')$ of inference strategies targeting $q(s \mid x'; x)$
Input: initial position x and estimate $\hat{Z}_x$ of $\tilde{\pi}(x)$
Output: next position x' and estimate $\hat{Z}_{x'}$ of $\tilde{\pi}(x')$

1 $(s, x') \sim q(s, x'; x)$
2 $w_{x'} \leftarrow \texttt{HME}(q(\cdot, x'; x), s, \mathcal{M}(x, x'))$
3 $(_, w_x) \leftarrow \texttt{IMPORTANCE}(q(\cdot \mid x; x'), \mathcal{M}(x', x))$
4 $(_, \hat{Z}_{x'}) \leftarrow \texttt{IMPORTANCE}(\pi(\cdot \mid x'), \mathcal{S})$
5 $u \sim \text{Uniform}(0, 1)$
6 **if** $u < \mathit{min}(1, \frac{\hat{Z}_{x'}}{\hat{Z}_x} w_{x'} w_x)$ **then**
7 | **return** $(x', \hat{Z}_{x'})$
8 **else**
9 | **return** $(x, \hat{Z}_x)$

Algorithm 2 on the q inference strategy, and the y'_i variables are generated while running Algorithm 1 on the q inference strategy. As a Gibbs update, this move is accepted with probability 1. The second kernel is a Metropolis-Hastings proposal that simultaneously: (i) swaps x_0 with y_0 (the proposed update of the 'main' state variable x_0), (ii) swaps each y_i with y'_i, for $1 \leq i \leq m$, and (iii) proposes updated $x_{1:n}$ values by running Algorithm 1 on the p inference strategy. The usual Metropolis-Hastings acceptance probability for this kernel, computed on the extended state space, is precisely the formula from Step (4) of the algorithm given above.

One consequence of this justification is that the *same* inference strategy for q must be used in steps 1 and 2 of the algorithm. Furthermore, from iteration to iteration, the inference strategy used in Step 3 for p must also be kept constant.

B FURTHER EXAMPLES

This appendix lists examples of popular Monte Carlo and variational inference algorithms, and explains how they can be viewed as inference strategies. In addition, some of these algorithms can be viewed as *inference strategy combinators*, because they feature user-chosen proposal distributions or variational families that can themselves be instantiated with inference strategies.[3]

B.1 N-PARTICLE IMPORTANCE SAMPLING

Consider the N-particle importance sampling estimator

$$\hat{Z} = \frac{1}{N} \sum_{i=1}^{N} \frac{p_0(x_2^{(i)}, x_0)}{q(x_2^{(i)})}, \text{ for } x_2^{(i)} \sim q.$$

The same estimator can be recovered as a *one-particle* estimate (Alg. 1) with a particular RAVI inference strategy, whose top layer is the generative model $p_0(x_1, x_0)$, and whose remaining layers work as follows:

Layer 1. The distribution $p_1(x_2, x_1; x_0)$ approximates $p_0(x_1 \mid x_0)$ using N-particle sampling/importance-resampling (SIR):

[3]This 'combinator' viewpoint evokes earlier work by [34] and [37]. For example, [37] introduce combinators for creating properly weighted samplers compositionally, with parameters that can be optimized using standard or nested variational objectives. Some of their combinators have equivalents in this section, e.g. their `propose` combinator is similar to the construction we present for Nested Importance Sampling in Section B.6. However: (1) the fundamental compositional operation in RAVI, of building inference strategy-based samplers $(p_0, \ldots, p_n)$ from individual distributions p_i, cannot be achieved using their combinators; (2) as such, some of the algorithms that RAVI covers cannot be constructed using their combinators; and (3) their combinators produce properly weighted samplers, which contain 'less information' than inference strategies: an inference strategy can be used, e.g., as a proposal distribution in Metropolis-Hastings, whereas properly weighted samplers cannot in general be used this way.

$$p_1(x_2, x_1; x_0) = \prod_{i=1}^{N} q(x_2^{(i)}) \cdot \sum_{i=1}^{N} \frac{w_i}{\sum_{j=1}^{N} w_j} \delta_{x_2^{(i)}}(x_1),$$

where $w_i = p_0(x_2^{(i)})/q(x_2^{(i)})$.

Layer 2. At the next layer, $p_2(x_3, x_2; x_1, x_0)$ infers the values of the N particles $x_2^{(i)}$ based on the single sample x_1 returned by p_1. It does this by randomly generating a chosen particle index x_3, and generating the unchosen particles from q:

$$p_2(x_3, x_2; x_1, x_0) = \frac{1}{N} \cdot \prod_{i \neq x_3} q(x_2^{(i)}) \cdot \delta_{x_1}(x_2^{(x_3)}).$$

Layer 3. The final layer of the inference strategy infers x_3 from x_2 and x_1, by finding particles $x_2^{(i)}$ that are equal to the chosen particle x_1 and selecting one at random:

$$p_3(x_3; x_2, x_1, x_0) = \frac{\mathbf{1}[x_2^{(x_3)} = x_1]}{\sum_{j=1}^{N} \mathbf{1}[x_2^{(j)} = x_1]}.$$

This inference strategy recovers $\hat{Z}$ as the derived estimator: for $x_2, x_1 \sim p_1(x_2, x_1; x_0)$, and $x_3 \sim p_3(x_3; x_2, x_1)$,

$$\hat{Z} = \frac{p_0(x_1, x_0) \cdot p_2(x_3, x_2; x_1, x_0)}{p_1(x_2, x_1; x_0) \cdot p_3(x_3; x_2, x_1, x_0)} = \frac{1}{N} \sum_{i=1}^{N} \frac{p_0(x_2^{(i)}, x_0)}{q(x_2^{(i)})}.$$

RAVI clarifies that this is a suboptimal choice of p_2; improved estimates can be obtained by improving p_2, either by incorporating problem-specific domain knowledge or via learning. However, in many cases, improved meta-inference may not be worth the computation required; it remains to be seen whether techniques such as amortized learning can be applied to deliver accuracy gains at low computational cost.

B.1.1 Instantiating the proposal q as its own inference strategy.

Suppose that one wishes to perform IS using a proposal that itself has auxiliary variables. Let $q_0(r_1, r_0), \ldots, q_n(r_n; r_{n-1}, \ldots, r_0)$ be an inference strategy, where the top layer q_0's marginal output distribution (over r_0) is the proposal distribution targeting $p_0(x_1 \mid x_0)$. Then we can modify the inference strategy presented above as follows:

1. Instead of generating $x_2^{(i)}$ from an atomic proposal q, p_1 runs Alg. 2 N times on the inference strategy $q_0, \ldots, q_n$, to obtain N weighted particles $(r_0^{(i)}, w_i)$, one of which is selected to return as x_1. In the process, auxiliary variables $r_{1,\ldots,n}^{(i)}$ are generated for each of the N particles. Denoting by x_2 the collection of all of these auxiliary r variables, we have:
$$p_1(x_2, x_1; x_0) = \prod_{i=1}^{N} \prod_{j=0}^{n} q_{2\lfloor \frac{j}{2} \rfloor}(r_j^{(i)} \mid r_{j-1}^{(i)}, \ldots, r_0^{(i)}) \cdot \sum_{i=1}^{N} \frac{w_i p_0(r_0^{(i)}, x_0)}{\sum_{j=1}^{N} w_j p_0(r_0^{(j)}, x_0)} \delta_{r_0^{(i)}}(x_1),$$
where w_i is the weight returned by Algorithm 2 in its ith run.
2. As in Layer 2 above, the new p_2 also generates x_3 uniformly at random in $\{1, \ldots, N\}$. For $i \neq x_3$, it generates $r_0^{(i)}, \ldots, r_n^{(i)}$ from $\prod_{j=0}^{n} q_{2\lfloor \frac{j}{2} \rfloor}(r_j^{(i)} \mid r_{j-1}^{(i)}, \ldots, r_0^{(i)})$, the same distribution as p_1. However, for the chosen particle $r^{(x_3)}$, $r_0^{(x_3)}$ is set deterministically to x_1, and the remaining $r^{(x_3)}$ variables are generated using Alg. 1 on the q inference strategy:
$$p_2(x_3, x_2; x_1, x_0) = \frac{1}{N} \prod_{i \neq x_3} \prod_{j=0}^{n} q_{2\lfloor \frac{j}{2} \rfloor}(r_j^{(i)} \mid r_{j-1}^{(i)}, \ldots, r_0^{(i)}) \cdot \prod_{j=0}^{n-1} q_{2\lfloor \frac{j}{2} \rfloor + 1}(r_{j+1}^{(x_3)} \mid r_j^{(x_3)}, \ldots, r_0^{(x_3)}) \cdot \delta_{x_1}(r_0^{(x_3)}).$$
3. Layer 3 is largely the same, except instead of comparing $x_2^{(x_3)}$ to x_1 it compares $r_0^{(x_3)}$ to x_1.

One way to think about this construction is that, given an inference strategy $p_0, q_0, \ldots, q_n$, where q_0 is a posterior approximation to p_0, it's possible to build a new inference strategy p_0, p_1, p_2, p_3, where p_1 is a better posterior approximation derived by running Algorithm 1 on the inference strategy $(p_0, q_0, \ldots, q_n)$ multiple times, and selecting one of the resulting particles to return (based on the relative $\hat{Z}$'s returned by Algorithm 1). That is, this construction is a way to improve any existing inference strategy by 'adding replicates.' The resulting estimator of Z is the mean of N independent $\hat{Z}$ estimates from the original inference strategy.

B.2 IMPORTANCE-WEIGHTED AUTOENCODERS

The importance-weighted auto-encoder arises by considering the same inference strategy as in Section B.1, but as a variational inference procedure (Alg. 3) rather than a Monte Carlo procedure.

Because p_1 of this inference strategy corresponds to SIR, it has been argued that IWAE is in fact 'vanilla' variational inference, but with a variational family that uses SIR to more closely approximate the posterior [4]. However, [9] have shown that deriving the ELBO for that variational family gives rise to a different objective, and that IWAE gives a looser lower bound on $\log Z$ than this idealized (but generally intractable) objective.

In the RAVI framework, these two objectives arise from two different inference strategies, which share the same p_1 (which does indeed correspond to SIR in both cases), but use a different p_2. IWAE uses the simple p_2 introduced in Section B.1, whereas [9]'s idealized objective can be derived by using the optimal choice of p_2—the exact posterior of p_1. The looser bound obtained by IWAE can be seen as a result of its p_2 performing poorer 'meta-inference': inference about the auxiliary variables of the SIR inference algorithm used in p_1.

B.3 N-PARTICLE SEQUENTIAL MONTE CARLO

The sequential Monte Carlo family of algorithms [7, 12] evolve a population of *weighted particles* to approximate a sequence of target distributions. SMC can be viewed as a three-layer inference strategy in which Layer 0 is the model, Layer 1 is the sampling distribution for SMC, and Layer 2 is the conditional SMC algorithm [1].

Consider the goal of generating a properly weighted collection of particles for a target distribution $p(x \mid y)$. At the top layer of the inference strategy, we choose a proposal that first generates a weighted collection of particles for an intermediate target $\pi(z)$, perhaps recursively using sequential Monte Carlo to do so. Having generated the weighted collection, $\{(z_1, w_1), \ldots, (z_M, w_M)\}$ a *resampling* step is performed. We consider a multi-particle resampling strategy $\Psi(\mathbf{z}, d\mathbf{j})$ that takes as input K particles and outputs M indices $j_1, \ldots, j_M$. For example, we might set $\Psi(\mathbf{z}, d\mathbf{j}) = \prod_{i=1}^{M} \frac{w_{j_i}}{\sum_{l=1}^{M} w_l}$ to perform *multinomial resampling*, selecting each j_i independently according to the weights of the z particles. Each resampled z_{j_i} is then transformed independently into a proposed x_i using a kernel $K(z, dx)$, and then reweighted to target p: $w'_i = w_{j_i} \frac{p(x_i|y)L(z_{j_i};x_i)}{\pi(z_{j_i})K(x_i;z_{j_i})}$. As described in Del Moral et al. [12], L is a user-chosen meta-inference kernel that attempts to reverse K, recovering z from x.

Meta-inference requires three ingredients. Given an x, we can use L to propose a z that was used to generate it. We then need to propose a possible weighted collection of z values that includes that z at some index. This can be done in two steps: first choosing an index where the distinguished z will end up (using an index distribution $\hat{\psi}$), and then using a distribution $\hat{G}$ to perform meta-infernece about the sampling process that generated the collection of z particles. When that generating process is itself an SMC algorithm, $\hat{G}$ will involve running another L kernel on the distinguished z particle, sampling another ancestor index using $\hat{\psi}$, and running another $\hat{G}$ to generate the previous step's weighted collection. Repeating this process yields the conditional SMC algorithm of [1]; variants can be designed to accommodate different resampling strategies Ψ or more general SMC-like algorithms, as in Lindsten et al. [19] or our Agglomerative Monte Carlo algorithm.

B.4 VARIATIONAL SEQUENTIAL MONTE CARLO

Variational Sequential Monte Carlo [28] corresponds to Alg. 3, with the same RAVI inference strategy as in Appendix B.3.

B.5 ANNEALED IMPORTANCE SAMPLING

In annealed importance sampling, the practitioner chooses a sequence of target distributions $\pi_0(x_1), \ldots, \pi_T(x_1)$, where $\pi_T(x_1) = p_0(x_1 \mid x_0)$ is the posterior distribution of interest. Typically π_0 is chosen to be a distribution that is easy to approximate with a proposal q_0, and each π_i is slightly closer to the true target π_T than the last. The user also chooses a sequence of kernels $K_i(x^{(i)}; x^{(i-1)})$, where K_i is stationary for π_i. The algorithm begins by sampling an initial point $x_2^{(0)} \sim q_0$, transforming it through the sequence of kernels to obtain $x_2^{(1)}, \ldots, x_2^{(T-1)}$, and returning $x_1 := x_2^{(T-1)}$ as the inferred value of x_1. The associated weight is

$$w = \frac{\pi_0(x_2^{(0)}) \cdot \pi_1(x_2^{(1)}) \cdot \cdots \cdot \pi_T(x_2^{(T-1)})}{q_0(x_2^{(0)}) \cdot \pi_0(x_2^{(1)}) \cdot \cdots \cdot \pi_{T-1}(x_2^{(T-1)})}.$$

We justify this as an inference strategy as follows. Layer 0 is, as usual, the target model, $p_0(x_1, x_0)$. Layer 1 represents the forward process:

$$p_1(x_2, x_1; x_0) = q_0(x_2^{(0)}) \prod_{i=1}^{T-1} K_i(x_2^{(i)}; x_2^{(i-1)}) \delta_{x_2^{(T-1)}}(x_1).$$

Layer 2's job is to infer the sequence $x_2^{(0)}, \ldots, x_2^{(T-1)}$ from the observed value x_1. It generates the hypothesized sequence in reverse, beginning with $x_2^{(T-1)} = x_1$ and repeatedly applying kernels $L_i(x_{T-i-1}; x_{T-i}) = \frac{\pi_{T-i}(x_2^{(T-i-1)}) \cdot K_{T-i}(x_2^{(T-i)}; x_2^{(T-i-1)})}{\pi_{T-i}(x_2^{(T-i)})}$:

$$p_2(x_2; x_1, x_0) = \delta_{x_1}(x_2^{(T-1)}) \prod_{i=1}^{T-1} L_i(x_2^{(T-i-1)}; x_2^{T-i}).$$

Note that if K is a stationary kernel for π_i, so is K^m for any natural number m. With sufficient computation (increasing m), we can ensure that the AIS top-level inference distribution p_1 is arbitrarily close to the target posterior $p_0(x_1 \mid x_0)$. However, doing so will not necessarily lead to lower-variance weights: RAVI makes clear that it is also necessary to consider the quality of meta-inference.

Consider the job of L_1, which is to infer $x_2^{(T-2)}$ from $x_2^{(T-1)}$. AIS's choice of L_1 is the exact posterior *assuming that, in the forward direction,* $x_2^{(T-2)}$ *was distributed according to* π_{T-1}. However, in the forward direction, if each K is run sufficiently many times to ensure mixing at each step, $x_2^{(T-2)}$ will in fact be distributed according to π_{T-2}. This gap—between the optimal L kernels and the actual L kernels—is partly responsible for the variance of the AIS estimator, and can be mitigated by using a finer annealing schedule that brings successive target distributions closer together. It could also be mitigated by learning a better reverse annealing chain than the default L kernels yield.

B.6 NESTED SEQUENTIAL MONTE CARLO

We first consider Nested Importance Sampling. As in RAVI-MC, Nested Importance Sampling is concerned with importance sampling when the proposal distribution q cannot be tractably evaluated. But RAVI and NIS take different approaches:

1. RAVI assumes q can be simulated, but that the (normalized) density cannot be evaluated. RAVI generates proposals exactly distributed according to the user's desired proposal q (called p_1 in our exposition), and generates approximations to the ideal importance weights.
2. NIS does not assume q can even be simulated, only that its unnormalized density is available. As such, proposals are not simulated from q, but rather from a Sampling/Importance-Resampling (SIR) approximation to q.

The NIS procedure with an intractable proposal q corresponds exactly to a special case of the RAVI algorithm, with the RAVI proposal p_1 set *not* to q but rather to an SIR sampling distribution targeting q using some tractable proposal h. The remainder of the inference strategy is then the same as in Section B.1. The only difference is that in NIS's p_1, the particle index c is chosen according to weights $w_i = q(x_1^{(i)})/h(x_1^{(i)})$ that correct a tractable distribution h toward an intractable but desired proposal q, whereas in the version from Section B.1, the weights $w_i = p_0(x_1^{(i)}, x_0)/h(x_1^{(i)})$ correct the tractable proposal directly toward the target distribution, and there is no intractable proposal q.

More generally, [27] consider procedures other than SIR for approximating q, arguing that any properly weighted sampler for the intractable proposal q will do. If we let $q_0, q_1, \ldots, q_n$ be a RAVI inference strategy representing the properly weighted sampler for the intractable proposal $q = q_0$ (where here, we take $q_0(x_1)$ to be an arbitrary unnormalized density, rather than a latent-variable model's posterior), then the Nested IS procedure that uses this properly weighted proposal to perform inference in $p_0(x_0, x_1)$ is simply the inference strategy $p_0, q_1, \ldots, q_n$, i.e., only the top layer (the target distribution) is changed, from the intractable proposal to the target model. Indeed, the weight formula that [27] offer is $w = \hat{Z} p_0(x_0, x_1)/q_0(x_1)$, where $(x_1, \hat{Z})$ is a properly weighted sample for q. In our setting, the estimator $\hat{Z}$ computed by Alg. 1 contains a factor for the top-layer density of the inference strategy, in this case q_0, and multiplying by $p_0(x_0, x_1)/q_0(x_1)$ is equivalent to replacing that top layer with p_0.

Nested SMC is similar, except that instead of using properly weighted samplers for a single intractable importance proposal q, they can be used for every transition proposal within an SMC algorithm. In RAVI, this can be expressed by constructing an inference strategy for SMC, where the proposal distributions used at each time step are themselves inference strategies. (See end of Section B.1 for a discussion and example of how to incorporate RAVI inference strategies as proposals in larger algorithms.)

B.7 SMC^2

[8] introduced the SMC^2 algorithm for inferring both static parameters θ and time-varying latents $z_1, \ldots, z_T$ in state-space models. The algorithm performs sequential Monte Carlo on a sequence of target distributions, where the target π_i is defined over both θ and *all the random variables generated by a particle filter* inferring $z_1, \ldots, z_i$. We can encode the algorithm as an inference strategy as follows:

Layer 0. As usual, the top layer $p_0(x_1, x_0)$ is the model. In SMC^2, we assume that $x_1 = (\theta, z_{1,\ldots,T})$.

Layer 1. Running SMC^2 yields a set of weighted particles, each of which contains an estimate of θ and a *weighted collection* of estimates of $z_{1,\ldots,T}$. Denote by x_2 the set of all random variables generated by the SMC^2 algorithm, and by $p_1(x_2; x_0)$ their sampling distribution. We extend this with $p_1(x_1 \mid x_2; x_0)$, which chooses one of the θ-particles at random based on the relative particle weights, and then, using that chosen particle's z-particle filter, chooses a particular $z_{1,\ldots,T}$ trajectory. The selected $(\theta, z_1, \ldots, z_T)$ sequence is returned as x_1.

Layer 2. The next layer, p_2, sees the final $x_1 = (\theta, z_{1,\ldots,T})$ inferred by Layer 1, and must infer the SMC^2 algorithm's trace x_2. To do this, we use a 'conditional SMC^2' procedure, which begins by sampling two independent ancestor trajectories of length T, one for the θ particle filter, and one for the z particle filter. Then SMC^2 is run forward, but fixing these ancestor indices for the chosen θ and z trajectories in x_1.

B.8 AMORTIZED REJECTION SAMPLING

Consider a generative model $p_0(x_1, x_0)$ where the latent variables x_1 to be marginalized or inferred represent the trace of a rejection sampling loop, $x_1 = (m, x_1^{(1)}, \ldots, x_1^{(m)})$. Here, the $x_1^{(i)}$ are drawn independently from a distribution q, until some predicate P holds of the most recent particle $x_1^{(m)}$, at which point the loop stops. The observation x_0 depends on the final sample $x_1^{(m)}$, but not the earlier, rejected samples or the number of samples m.

[26] proposed a technique called *Amortized Rejection Sampling* for performing inference in this model. Their technique corresponds to a particularly elaborate, five-layer RAVI inference strategy (including p_0), and is parameterized by two natural numbers N and M that control the amount of computation used by the algorithm.

Layer 1. The first layer of the inference strategy, $p_1(x_2, x_1; x_0)$, generates a proposed x_1 in stages:

1. First, a proposal $h_\theta(x_1; x_0)$ attempts to generate possible $x_1^{(m)}$ values that both satisfy the predicate P and explain the data x_0. It may take multiple samples $x_1'^{(i)} \sim h$ before the predicate P is satisfied. Let m' be the number of samples required. Then once the loop is finished, $x_1'^{(m')}$ is proposed as the value of $x_1^{(m)}$ from the model.
2. Second, it's necessary to generate the rejected particles $x_1^{(1\ldots m-1)}$. To do this, p_1 first runs M of its own rejection sampling loops with the sampling distribution q, to produce $x_1''^{(i,j)}$ for $i \in \{1, \ldots, M\}$, $j \in \{1, \ldots, m_i\}$, where m_i is the total number of samples in the ith rejection loop. Now, p_1 chooses one of those loops $c \in \{1, \ldots, M\}$, where

$p_1(c = i) \propto m_i - 1$. Within that loop, a *number of rejections* $l \in \{0, \ldots, m_c - 1\}$ is chosen uniformly at random. Finally, the rejected samples $x_1''^{(c,1\ldots l)}$ are proposed as the values of $x_1^{(1\ldots l)}$, and m is proposed as $l + 1$.

In this way, p_1 proposes all the components of x_1, namely the l rejected samples and the single accepted sample. However, in doing so, it introduces many auxiliary variables of its own: the $m' - 1$ rejected samples $x_1'^{(1\ldots m'-1)}$ from Step (1), the $M - 1$ entire rejection sampling loops $x_1''^{(i,1\ldots m_i)}$ for $i \neq c$, and the partial rejection sampling loop $x_1''^{(c,l+1\ldots m_c)}$. We collect these into the latent variable x_2; they must be inferred at Layer 2.

Layer 2. Now, $p_2(x_3, x_2; x_1, x_0)$ needs to infer the auxiliary variables in x_2:

1. To infer the $m' - 1$ samples $x_1'^{(i)}$ from h that were rejected in Step 1 of p_1's generative process, we simply run h until an accepted sample is reached. Call these samples $x_2^{(i)}$ for $i = 1, \ldots, m'$, where $x_2^{(m')}$ is an accepted sample. The first $m' - 1$ samples are proposed as the values of $x_1'^{(i)}$.
2. To infer the traces of the $M - 1$ rejection loops *other* than loop c, just run $M - 1$ fresh rejection loops. To infer the suffix of rejection loop c (i.e., the variables $x_1''^{(c,l+1\ldots m_c)}$), run a final rejection loop.

This layer introduces only one new auxiliary variable: the accepted sample $x_2^{(m')}$. This becomes the latent variable x_3.

Layer 3. We also require a $p_3(x_4, x_3; x_2, x_1, x_0)$, to infer x_3, the discarded accepted sample from Step 1 of Layer 2:

1. To infer the accepted sample $x_3 = x_2^{(m')}$ that was discarded in p_2, run N-particle SIR with proposal $h_\theta(x)$ and 0-1 target density $\mathbf{1}[P(x)]$. That is, draw N samples $x_4^{(1\ldots N)}$ independently from h_θ and weight them either 0 or 1 according to whether they satisfy the predicate P. Then, choose a particle index c' where $p_3(c' = i)$ is proportional to the weight of particle i. Return $x_3 = x_4^{(c')}$ as the proposed accepted sample. Note that it's possible all N weights are 0; in this case, c' is chosen uniformly, and the returned x_3 is not in the support of the Layer 2 posterior. As such, the odd-numbered marginal $p(x_3; x_2, x_1, x_0)$ is *not* absolutely continuous w.r.t. the even-numbered posterior $p_2(x_3 \mid x_2; x_1, x_0)$, and so although this RAVI inference strategy is a valid *wide* inference strategy, it is not a valid narrow inference strategy (see Section C).

This algorithm introduces $N - 1$ new auxiliary variables $x_4^{(i)}$ for $i \neq c'$, as well as the variable c' indicating the index of the chosen particle. Together these form x_4.

Layer 4. Finally, $p_4(x_4; x_3, x_2, x_1, x_0)$ recovers the unchosen particles $x_4^{(i)}$ from the SIR procedure run at Layer 3. It does this by randomly sampling c' uniformly from $\{1, \ldots, N\}$, and then generating the $N - 1$ unchosen particles independently from h_θ—as in Section B.1.

This completes the inference strategy $p_0, \ldots, p_4$ corresponding to Amortized Rejection Sampling. Algorithm 4 can be used to learn the parameters θ—this ends up reducing to a procedure whereby samples $(x_1, x_0) \sim p_0(x_1, x_0)$ are repeatedly generated, from which $(x_1^{(m)}, x_0)$ pairs are extracted and used to train $h_\theta(x_1^{(m)}; x_0)$ in a supervised manner. This is the amortization referred to by the name of the technique. Furthermore, Algorithm 1 can be applied to the inference strategy above to generate estimates with the same weights as those computed by [26].

B.9 HAMILTONIAN VARIATIONAL INFERENCE

Hamiltonian Variational Inference [33] is a hybrid of Hamiltonian Monte Carlo and variational inference. It corresponds to the following four-layer tower:

Layer 0: The top layer is the generative model $p_0(x_1, x_0)$ in which inference is performed. We assume $x_1 \in \mathbb{R}^d$ is a continuous vector.

Layer 1: To approximate $p_0(x_1 \mid x_0)$, we use the following HMC-inspired distribution:

$$p_1(x_2, x_1; x_0) = q_0(x_2^{init}) q_v(x_2^v) \delta_{(\pi_1 \circ \mathrm{LF})(x_2^{init}, x_2^v)}(x_1),$$

which generates an initial point $x_2^{init} \sim q_0$, samples a momentum vector $x_2^v \sim q_v$, and uses a leapfrog integrator LF to simulate Hamiltonian dynamics for some fixed number of steps with the Hamiltonian $H(x, v) = \log p(x_1, x_0) + \log q_v(v)$. The final position variable is then set as the proposed x_1. (Here, π_1 is the projection operator that extracts the position element of the (position, momentum) pair.)

Layer 2: Given an observed x_1, p_2 attempts to infer the initial point x_2^{init} and the momentum v. To do so, it samples a random momentum x_3, and runs the leapfrog integrator to generate both a final momentum (the negated value of which will be proposed as x_2^v) and the final position (which will be proposed as x_2^{init}:

$$p_2(x_3, x_2; x_1, x_0) = r_v(x_3; x_1)\delta_{\mathrm{LF}(x_1,x_3)}((x_2^{init}, -x_2^v)).$$

Layer 3: Given x_1 and x_2, the third layer deterministically sets the backward momentum x_3:

$$p_3(x_3; x_2, x_1, x_0) = \delta_{(\pi_2 \circ LF)(x_2^{init}, x_2^v)}(-x_3).$$

The distributions q_0, q_v, and r_v can all be chosen from families of distributions parmaeterized by some continuous parameters θ, in which case Algorithm 3 can be used for optimization.

B.10 ANTITHETIC SAMPLING

Consider a model $p_0(x_1, x_0)$ and a proposal $q(x_1)$ that approximates $p_0(x_1 \mid x_0)$. Further suppose q is invariant under some bijective transformation T:

$$\forall x_1, q(x_1) = q(T(x_1)).$$

For example, a univariate Gaussian proposal with mean μ is invariant under $T(x_1) = 2\mu - x_1$. Antithetic sampling generates a sample x_1 from q, but instead of using the estimator $\hat{Z} = p_0(x_1, x_0)/q(x_1)$, it uses

$$\hat{Z} = \frac{p_0(x_1, x_0) + p_0(T(x_1), x_0)}{2q(x_1)}.$$

This can be justified as a RAVI-MC inference strategy with p_0 as before and

$$p_1(x_2, x_1; x_0) = q(x_2) \cdot \frac{p_0(x_1, x_0)}{p_0(x_2, x_0) + p_0(T(x_2), x_0)} \cdot \mathbf{1}\left[x_1 \in \{x_2, T(x_2)\}\right].$$

Intuitively, p_1 generates a proposal $x_2 \sim q$, and then either chooses to return $x_1 = x_2$ or $x_1 = T(x_2)$, based on their relative probabilities under the model p_0. Finally, we have the third layer of the inference strategy,

$$p_2(x_2; x_1, x_0) = \frac{1}{2}\mathbf{1}[x_1 \in \{x_2, T(x_2)\}].$$

This layer is responsible for inferring which choice p_1 made: is $x_2 = x_1$, or is $x_2 = T^{-1}(x_1)$? Our p_2 makes this choice via a fair coin flip, which is optimal when $T = T^{-1}$, i.e., when T is an involution. In the general case a lower-variance estimator could be derived by setting p_2 to the exact posterior of p_1. Antithetic sampling can also be generalized to the case where a finite family of bijective transformations T_i are available.

We note that although the final expression for $\hat{Z}$ falls out of this inference strategy only when $q(x) = q(T(x))$ for all x, nothing in the inference strategy itself exploits this assumption, and the same inference strategy could be applied to T without this property, to derive other estimators that—intuitively—simultaneously consider a proposal x_2 and a deterministic function of it $T(x_2)$ as possible locations.

C ABSOLUTE CONTINUITY

When we defined inference strategies $\mathcal{S}$ targeting π, we required that $\mathcal{S}.q$ and π be *mutually* absolutely continuous, a stronger requirement than in importance sampling. We now consider relaxing this requirement, by requiring only *one-sided* absolute continuity. We define two *kinds* of inference strategy, depending on which direction of absolute continuity holds:

1. An inference strategy $\mathcal{S}$ targeting π is *wide* if π is absolutely continuous with respect to $\mathcal{S}.q$, and either $\mathcal{S}.q$ has a tractable marginal density or $\mathcal{S}.\mathcal{M}(x)$ is a narrow inference strategy targeting $\mathcal{S}.q(\cdot \mid x)$ for all x.

2. An inference strategy $\mathcal{S}$ targeting π is *narrow* if $\mathcal{S}.q$ is absolutely continuous with respect to π, and either $\mathcal{S}.q$ has a tractable marginal density or $\mathcal{S}.\mathcal{M}(x)$ is a wide inference strategy targeting $\mathcal{S}.q(\cdot \mid x)$ for all x.

Then an inference strategy as defined in the main paper is one that is both wide and narrow.

Narrow inference strategies can serve as variational families within variational inference algorithms. Wide inference strategies can be used as importance sampling and SMC proposals, as well as variational families for *amortized* variational inference. Inference strategies used as MCMC proposals must be both wide and narrow.

D OTHER APPLICATIONS OF RAVI INFERENCE STRATEGIES

D.1 REJECTION SAMPLING WITH RAVI

As in any properly weighted sampler, if the weights produced by Alg. 1 can be bounded above by a constant M, a RAVI inference strategy can be used for exact inference via rejection sampling: a sample (x, w) is drawn using Alg. 1, and then accepted with probability $\frac{w}{M}$. The weight w for an inference strategy can be viewed as a product of the normalizing constant $Z = p_0(x_0)$ with the normalized importance weights $w_i = \frac{p_{2\lfloor \frac{i}{2} \rfloor}(x_i|x_{0:i-1})}{p_{2\lceil \frac{i}{2} \rceil - 1}(x_i|x_{0:i-1})}$ for $i = 1, \ldots, n$. As such, if upper bounds M_Z and $M_{1:n}$ can be found for these quantities (that hold for all $x_{1:n}$), the product of these bounds, $M = M_Z \prod_{i=1}^{n} M_i$, is a bound on w. Thus, as in properly weighted sampling and in variational inference with RAVI, it is possible to reason about the RAVI inference strategy in terms of pairs of consecutive layers.

D.2 ESTIMATING KL DIVERGENCES BETWEEN MODELS WITH RAVI INFERENCE STRATEGIES EQUIPPED

Suppose $p_0(x_1, x_0), \ldots, p_n(x_n)$ and $q_0(r_1, r_0), \ldots, q_m(r_m)$ are two RAVI inference strategies, and that the marginal distributions $p_0(x_0)$ and $q_0(r_0)$ are defined on the same space and are mutually absolutely continuous. Then the AIDE algorithm [10] can be adapted to give a stochastic upper bound on the symmetric KL divergence between p_0 and q_0. First, (x_0, w_{px}) and (r_0, w_{qr}) are generated using Algorithm 2 on each inference strategy. Then, Algorithm 1 is run on the p inference strategy and r_0 to yield w_{pr}, and on the q inference strategy and x_0 to yield w_{qx}. Summing the logs of the four weights gives an estimate $\hat{D}$ whose expectation is:

$$\mathbb{E}[\hat{D}] = \mathbb{E}_{x_0 \sim p_0}[\mathcal{U}_0^p(x_0) - \mathcal{L}_0^q(x_0)] + \mathbb{E}_{r_0 \sim q_0}[\mathcal{U}_0^q(r_0) - \mathcal{L}_0^p(r_0)].$$

As the marginal likelihood bounds $\mathcal{U}_0$ and $\mathcal{L}_0$ become tighter, this expectation approaches the true symmetric KL between p_0 and q_0, i.e., $D = KL(p_0||q_0) + KL(q_0||p_0)$. Theorem 4 allows us to characterize the tightness of these bounds, and thus of the stochastic upper bound $\hat{D}$ on the symmetric KL, in terms of KL divergences between successive layers of each inference strategy. Improving inference at any layer of the inference strategy tightens the bound $\hat{D}$, yielding less biased estimates of D.

E REPARAMETERIZATION TRICK GRADIENT ESTIMATORS

In this section, we present a version of Algorithms 3 and 4 that utilize reparameterization gradients, rather than score function gradients. Using these algorithms requires that an inference strategy be *reparameterizable*.

Definition: A *reparameterizable inference strategy $\mathcal{S}$ with arguments θ* specifies:

- A reparameterizable posterior approximation $\mathcal{S}.q$, which is one of:
  - a tractable proposal: a tuple $(\mathcal{S}.q(x;\theta), \mathcal{S}.q.g(\epsilon), \mathcal{S}.q.f(\epsilon, \theta)$, such that q is the pushforward of g by f; or
  - an intractable proposal: a tuple $(\mathcal{S}.q(r, x;\theta), \mathcal{S}.q.g(\epsilon_r, \epsilon_x), \mathcal{S}.q.f_r(\epsilon_r, \theta), \mathcal{S}.q.f_x(\epsilon_x, \theta))$, such that q is the pushforward of g by $\lambda(\epsilon_r, \epsilon_x).(f_r(\epsilon_r, \theta), f_x(\epsilon_x, \theta))$.
- If the latter, a reparameterizable meta-inference strategy $\mathcal{S}.\mathcal{M}$, with arguments (x, θ), that given argument (x, θ), targets $\mathcal{S}.q(r \mid x;\theta)$.

Now, reparameterized estimators can be derived by applying standard automatic differentiation to the following algorithm, which only samples from distributions that do not depend on parameters:

Algorithm 6: RAVI ELBO estimator (`ELBO`)
Input: unnormalized model $\tilde{p}(x)$
Input: inference strategy $\mathcal{S}$ with arguments
Input: arguments θ
Output: unbiased estimates of $\mathcal{L}$ (differentiable w.r.t. θ)

if $\mathcal{S}.q$ *has a tractable marginal density* **then**
  $\epsilon_x \sim \mathcal{S}.q.g$
  $x \leftarrow \mathcal{S}.q.f(\epsilon_x, \theta)$
  $\hat{U} \leftarrow \log \mathcal{S}.q(x; \theta)$
else if $\mathcal{S}.q(x;\theta) = \int \mathcal{S}.q(r, x; \theta) dr$ **then**
  $(\epsilon_r, \epsilon_x) \sim \mathcal{S}.q.g$
  $(x, r) \leftarrow (\mathcal{S}.q.f_x(\epsilon_x, \theta), \mathcal{S}.q.f_r(\epsilon_r, \theta))$
  $\hat{U} \leftarrow \texttt{EUBO}(\mathcal{S}.q(\cdot, x; \theta), r, \mathcal{S}.\mathcal{M}, (x, \theta))$
return $\log \tilde{p}(x) - \hat{U}$

Algorithm 7: RAVI EUBO estimator (`EUBO`)
Input: unnormalized model $\tilde{p}(x)$
Input: exact sample $x \sim p(x)$
Input: inference strategy $\mathcal{S}$ with arguments
Input: arguments θ
Output: unbiased estimate of $\mathcal{U}$ (differentiable w.r.t. θ)

if $\mathcal{S}.q$ *has a tractable marginal density* **then**
  $\hat{L} \leftarrow \log \mathcal{S}.q(x; \theta)$
else if $\mathcal{S}.q(x;\theta) = \int \mathcal{S}.q(r, x; \theta) dr$ **then**
  $\hat{L} \leftarrow \texttt{ELBO}(\mathcal{S}.q(\cdot, x; \theta), \mathcal{S}.\mathcal{M}, (x, \theta))$
return $\log \tilde{p}(x) - \hat{L}$

Note that in fact only every *other* posterior approximation in the unrolled strategy requires a reparameterized version: Algorithm 7 never samples from its $\mathcal{S}.q$, only evaluates the densities.

It would be interesting to develop variants of these algorithms that allow users to combine score-function and reparameterization estimation at different layers of nesting, or exploit other variance reduction tactics compositionally.