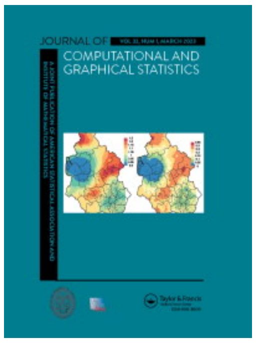


Distilling Importance Sampling for Likelihood Free Inference

Dennis Prangle & Cecilia Viscardi

JOURNAL OF COMPUTATIONAL AND GRAPHICAL STATISTICS
2023, VOL. 00, NO. 0, 1–11
https://doi.org/10.1080/10618600.2023.2175688

Taylor & Francis
Taylor & Francis Group

Distilling Importance Sampling for Likelihood Free Inference

Dennis Prangle[a] and Cecilia Viscardi[b]

[a]School of Mathematics, University of Bristol, Bristol, United Kingdom of Great Britain and Northern Ireland; [b]Department of Statistics, Computer Science, Applications, University of Florence, Florence, Italy

ABSTRACT
Likelihood-free inference involves inferring parameter values given observed data and a simulator model. The simulator is computer code which takes parameters, performs stochastic calculations, and outputs simulated data. In this work, we view the simulator as a function whose inputs are (1) the parameters and (2) a vector of pseudo-random draws. We attempt to infer all these inputs conditional on the observations. This is challenging as the resulting posterior can be high dimensional and involves strong dependence. We approximate the posterior using normalizing flows, a flexible parametric family of densities. Training data is generated by likelihood-free importance sampling with a large bandwidth value ϵ, which makes the target similar to the prior. The training data is "distilled" by using it to train an updated normalizing flow. The process is iterated, using the updated flow as the importance sampling proposal, and slowly reducing ϵ so the target becomes closer to the posterior. Unlike most other likelihood-free methods, we avoid the need to reduce data to low-dimensional summary statistics, and hence can achieve more accurate results. We illustrate our method in two challenging examples, on queuing and epidemiology. Supplementary materials for this article are available online.

ARTICLE HISTORY
Received April 2022
Accepted January 2023

KEYWORDS
Epidemics; Likelihood-free inference; Normalizing flows; Queues

1. Introduction

Many statistical models are specified by *simulators*, which can be used to generate data under the model given values of parameters. Probability calculations are often not possible for such models. In particular, it can be infeasible to evaluate the *likelihood function* $L(\theta)$: the probability (or density) of the observed data under parameters θ. This makes it difficult to use standard methods of statistical inference such as maximum likelihood and many Monte Carlo methods for Bayesian inference.

Likelihood-free inference (LFI)—also known as *simulation-based inference*—describes methods to perform inference using simulators without evaluating the likelihood function. One popular class of LFI methods is *approximate Bayesian computation* (ABC) (Marin et al. 2012). Here one runs the simulator under many θ values, and each time calculates a distance between the simulated data y and the observed data y_0. The distances are used to produce a sample (or weighted sample) of θs from an approximation to the posterior, for example, by selecting the θs with the smallest distances.

A drawback of ABC is that is suffers from a *curse of dimensionality*: the cost of producing an accurate posterior approximation rises rapidly with $\dim(y)$. An intuitive explanation is that, even under the best θ choices, close matches of y and y_0 are rare unless $\dim(y)$ is low. Therefore, it is common to use dimension reduction, replacing y with a vector of low dimensional *summary statistics* $s(y)$ when calculating distances. However, using summary statistics typically results in a loss of posterior accuracy.

An alternative class of LFI methods use *conditional density estimation* (Grazian and Fan 2019). These estimate a density based on simulated pairs of parameters and data. Then one can condition on the observed data to approximate its posterior distribution. These methods avoid the ABC curse of dimensionality, but typically still require dimension reduction of the data to summary statistics. So the associated loss of information remains a problem.

We aim to improve the efficiency of LFI algorithms by *parameter augmentation*. The idea is to infer not only the model parameters θ, but also all the random variables the simulator samples during its operation. Effectively this learns to control the simulator to produce output similar to the observations. Now it is no longer the case that outputting close matches will always be rare. In principle this can avoid the ABC curse of dimensionality without the need for summary statistics.

However, inference under parameter augmentation is challenging. The posterior is high dimensional and may have strong dependencies, for example, lying on a lower dimensional manifold. This makes it hard to design effective proposals for Monte Carlo methods. We use approximate versions of the posterior, which correspond to inference under noisy observation of the data, with a *bandwidth* parameter ϵ controlling the variance of the noise. The resulting distribution is the prior for $\epsilon \to \infty$, and the posterior for $\epsilon \to 0$. Thus, large ϵ produces a less challenging inference problem.

We approximate these distributions using *normalizing flows* (Papamakarios et al. 2021), a family of flexible and

CONTACT Dennis Prangle dennis.prangle@bristol.ac.uk University of Bristol, School of Mathematics, Bristol, BS8 1QU, United Kingdom of Great Britain and Northern Ireland.

Supplementary materials for this article are available online. Please go to www.tandfonline.com/r/JCGS.

computationally tractable distributions. They transform a random vector from a simple random distribution to a random vector from a complex distribution, by applying a sequence of learnable bijections. Normalizing flows can be trained using batches of data sampled from a target distribution.

We propose a method which alternates two steps. The first is importance sampling, using the current normalizing flow as the proposal. The target distribution is an approximate posterior with large ϵ, chosen so that importance sampling produces reasonably accurate results. The second step is to use the resulting weighted sampled to train the normalizing flow further. Following Li, Turner, and Liu (2017), we refer to this as *distilling* the importance sampling results. By iteratively distilling importance sampling results, we can target increasingly accurate posterior approximations, that is, reduce ϵ.

We demonstrate our algorithm on two challenging examples. In a queuing example, we obtain approximate inference results which are much more accurate than two ABC baselines: with and without summary statistics. In an epidemiology example, we are able to target the exact posterior.

In the remainder of the article, Section 2 presents background material. Section 3 discusses LFI parameter augmentation. Sections 4 and 5 describe our method. Section 6 illustrates it on a simple two dimensional inference task. Sections 7 and 8 present our main examples. Section 9 concludes with a discussion, including limitations and opportunities for extensions. Further details are available in the supplement, including a discussion of recommended tuning choices (Section E). Code for the examples can be found at *https://github.com/dennisprangle/DistillingImportanceSampling*, written using PyTorch (Paszke et al. 2019). All examples were run on a 16-core desktop PC.

1.1. Related Work and Novelty

Several recent articles (Müller et al. 2019; Arbel, Matthews, and Doucet 2021; Duan 2021; Naesseth, Lindsten, and Blei 2021) train a normalizing flow to use as an importance sampling proposal. Variations include training a Gaussian mixture (Jerfel et al. 2021) or a distribution transformed using polynomials (Cotter, Kevrekidis, and Russell 2020), rather than the usual neural network approaches in normalizing flows literature (reviewed in Section 2.3). The novelty of our work is an application to likelihood-free inference. To do so, we sequentially target a sequence of approximate distributions, unlike the prior work listed above.

Parameter augmentation methods for likelihood-free inference have been used previously. Graham and Storkey (2017) propose a method using constrained Hamiltonian Monte Carlo (HMC). This alternates HMC steps with projection steps to move the MCMC state back onto a target manifold. A limitation of this method is that it requires a differentiable simulator. Optimization Monte Carlo (OMC) (Meeds and Welling 2015) generates random seeds to use within the simulator, then uses optimization to find the corresponding θ values which minimize the distance of the resulting y to y_0, and computes appropriate weights. Robust OMC (Ikonomov and Gutmann 2020) produces multiple θs given a random seed. Rare event ABC (Prangle, Everitt, and Kypraios 2018) is a MCMC algorithm which, whenever a new θ is proposed, uses rare event methods to estimate the probability of the simulator producing a close match.

Let x denote the internal random variables used by the simulator (defined more precisely in Section 3). Then OMC methods essentially sample x from its prior, which could be inefficient if the posterior for x is concentrated compared to the prior. Rare event ABC infers $p(x|\theta, y)$ for each proposed θ, which becomes expensive for long MCMC chains. In contrast to these two methods, our approach aims to infer $p(\theta, x|y)$, which we argue is more efficient.

Joint inference of θ and x can also sometimes be performed by sophisticated MCMC algorithms specialized to particular applications. For instance Shestopaloff and Neal (2014) give an MCMC algorithm for the queuing example we consider in Section 7. Our goal in this article is to provide a more generic approach.

More broadly, our approach has connections to several inference methods. Concentrating on its importance sampling component, it is closely related to *adaptive importance sampling* (Cornuet et al. 2012). Concentrating on training an approximate density, it is related to the *cross-entropy method* (Rubinstein 1999), an *estimation of distribution* algorithm (Larrañaga and Lozano 2002), and *reweighted wake-sleep* (Bornschein and Bengio 2014).

2. Background

2.1. Likelihood-Free Inference

Suppose we observe data y_0, assumed to be the output of a probability model with density $p(y|\theta)$ under some parameters θ, and we have access to a prior density $\pi(\theta)$. Bayesian inference aims to find the corresponding posterior density $p(\theta|y_0) \propto \pi(\theta)p(y_0|\theta)$. When the *likelihood function* $p(y|\theta)$ can be evaluated, many approaches to posterior inference are available.

However, many models are specified using a *simulator*, typically some computer code. The simulator's inputs are parameters θ, and the output is data y. This effectively defines a distribution[1] for y conditional on θ. However, probability calculations under this distribution are typically not feasible, including evaluation of the likelihood function. So standard likelihood-based methods of inference are not possible. *Likelihood-free inference* (LFI) methods attempt to perform approximate inference in this setting without directly evaluating the likelihood function.

One popular LFI approach is approximate Bayesian computation (ABC). The simplest ABC algorithm, *rejection ABC*, samples (θ, y) pairs from the prior and simulator, and accepts those where a distance between y and y_0 (e.g., the Euclidean distance $||y - y_0||$ if data is a vector of fixed length) is below some threshold ϵ. The accepted θs form a sample from an approximation to the posterior. More efficient ABC algorithms exist, for instance using more sophisticated methods of sampling θ values, or using importance sampling ideas to allow weighted samples.

Despite these improvements, close matches of y and y_0 are rare unless $\dim(y)$ is low. As a consequence, ABC suffers from

[1]If the simulator is deterministic the distribution is a point mass. However, it is more common to consider stochastic simulators with LFI.

a *curse of dimensionality*: in the asymptotic case $\epsilon \to 0$, the cost of ABC rapidly increases with $\dim(y)$. So ABC typically uses dimension reduction. That is, acceptance now occurs when $||s(y) - s(y_0)||$ is small, for some function s mapping data to a vector of *summary statistics*. The loss of information entailed by dimension reduction reduces inference accuracy (as sufficient statistics are rarely available).

See Prangle, Everitt, and Kypraios (2018) (end of Section 2.1) for a brief review of the ABC curse of dimensionality, and Marin et al. (2012) for a review of other relevant aspects of ABC. The supplement (Section C.1) gives more details of a specific ABC algorithm we use as a baseline in one of our examples.

As mentioned in Section 1, an alternative approach to LFI is through *conditional density estimation* methods (CDE), as reviewed by Grazian and Fan (2019). CDE methods estimate a density, often a normalizing flow, using simulated (θ, y) pairs. The density estimated could be the joint $\pi(\theta, y)$ or a conditional: $p(\theta|y)$ or $p(y|\theta)$. One can then condition on the observed data to approximate its posterior distribution. The method for conditioning depends on which density was estimated. CDE methods avoid the ABC curse of dimensionality, but typically still require dimension reduction of the data to summary statistics. So the resulting loss of information remains a problem.

2.2. Importance Sampling

Let $p(\xi) = \tilde{p}(\xi)/Z$ be a *target density* where only $\tilde{p}(\xi)$ can be evaluated, and the value of the normalizing constant $Z = \int \tilde{p}(\xi) d\xi$ is unknown. Importance sampling (IS) is a Monte Carlo method to estimate expectations of the form $I = \mathrm{E}_{\xi \sim p}[h(\xi)]$, for some function h. Here we review relevant aspects. For full details see, for example, Robert and Casella (2013) and Rubinstein and Kroese (2016).

IS requires a *proposal density* $\lambda(\xi)$ which can easily be sampled from, and must satisfy

$$\mathrm{supp}(p) \subseteq \mathrm{supp}(\lambda), \tag{1}$$

where supp denotes support. Then

$$I = \mathrm{E}_{\xi \sim \lambda}\left[\frac{p(\xi)}{\lambda(\xi)} h(\xi)\right]. \tag{2}$$

So an unbiased Monte Carlo estimate of I is

$$\hat{I}_1 = \frac{1}{NZ} \sum_{i=1}^{N} w_i h(\xi^{(i)}), \tag{3}$$

where $\xi^{(1)}, \xi^{(2)}, \dots, \xi^{(N)}$ are independent samples from λ, and $w_i = \tilde{p}(\xi^{(i)})/\lambda(\xi^{(i)})$ is an *importance weight*.

Typically Z is estimated as $\frac{1}{N}\sum_{i=1}^{N} w_i$ giving

$$\hat{I}_2 = \sum_{i=1}^{N} w_i h(\xi^{(i)}) \Big/ \sum_{i=1}^{N} w_i, \tag{4}$$

a biased, but consistent, estimate of I. Equivalently

$$\hat{I}_2 = \sum_{i=1}^{N} s_i h(\xi^{(i)}),$$

for *normalized* importance weights $s_i = w_i / \sum_{i=1}^{N} w_i$.

A drawback of IS is that it can produce estimates with large, or infinite, variance if λ is a poor approximation to p. Hence, diagnostics for the quality of the results are useful. A popular diagnostic is *effective sample size* (ESS),

$$N_{\mathrm{ESS}} = \left(\sum_{i=1}^{N} w_i\right)^2 \Big/ \sum_{i=1}^{N} w_i^2. \tag{5}$$

Under various assumptions, $\mathrm{Var}(\hat{I}_2)$ roughly equals the variance of an idealized Monte Carlo estimate based on N_{ESS} independent samples from $p(\xi)$ (Liu 1996). Elvira, Martino, and Robert (2022), among others, note that ESS can be an unreliable diagnostic in practice and research is needed to propose better alternatives.

2.3. Normalizing Flows

A *normalizing flow* represents a random vector ξ with a complicated distribution as an invertible transformation of a random vector z with a simple *base distribution*, typically $\mathcal{N}(0, \mathrm{I})$.

Recent research has developed flexible learnable families of normalizing flows. See Papamakarios et al. (2021) for a review. We focus on autoregressive neural spline flows (Durkan et al. 2019), which we will refer to as "spline flows" as a shorthand. See Section 5.5 for comments on exploratory work with an alternative choice.

Spline Flows Description. An autoregressive transformation transforms input vector u to output vector v using $v_i = \tau(u_i, h(u_{<i}))$ (where $u_{<i}$ refers to the u_j values with $j < i$). Here τ is a monotonic and invertible transformation of its first argument. The particular transformation used depends on the second argument. So v_i is a transformation of u_i, and the transformation used depends only on $u_{<i}$. Therefore, the overall transformation has a triangular Jacobian matrix, facilitating quick density calculations. Furthermore, a masked autoregressive neural network (Germain et al. 2015; Nash and Durkan 2019) is typically used for h. This allows $h(u_{<i})$ to be computed in parallel for all i values.

Durkan et al. (2019) propose using a spline transformation for τ. This transformation is a piecewise function based on partitioning a bounding box $[-B, B]$ into several bins. The type of function used is chosen so it can be fully defined by: the location of knots (bin boundaries); the function values at the knots; and (in some cases) derivatives at the knots. All these details should be the output of $h(u_{<i})$. We use piecewise rational quadratic transformations, following Durkan et al. (2019). Outside the bounding box the function is defined to be the identity.

Spline Flows Properties. Some relevant properties of spline flows are as follows. See Papamakarios et al. (2021) for full discussion of the points made here.

- *Universality.* An autoregressive flow can represent any distribution meeting some simple conditions, if any τ and h functions can be used. In practice this means the expressiveness of spline flows is only limited by the complexity of the splines and neural networks used.

- *Sampling.* Samples of ξ can rapidly be drawn from $q(\xi;\phi)$.
- *Gradient calculation.* It is reasonably fast to compute $\nabla \log q(\xi;\phi)$. (Throughout the article ∇ represents the gradient operator with respect to ϕ.)

Universality means that spline flows can approximate many distributions well. Sampling and gradient calculation are required in our algorithm. Alternative types of flow can allow faster gradient calculation, but are often less expressive.

We note that Gaussian mixture models are an alternative to flows. However, their cost of sampling and density evaluation grows rapidly with $\dim \xi$, as it involve a $O([\dim \xi]^3)$ matrix inversion cost.

3. Parameter Augmentation

Here we outline an approach to likelihood-free inference which our algorithm will use. The idea is to infer not just the parameters θ, but also all the internal stochastic behavior of the simulator. To do so, we introduce a random variable x. This represents all outputs of an underlying random number generator used by the simulator. We suppose all the sampled random variables required during simulation can be expressed as transformations of θ and x. So the simulator is now a deterministic function $y(\xi)$, where ξ represents the *augmented parameters* (θ, x).

Throughout this article we work with simulators where x can be expressed as a fixed length random vector $x \sim \mathcal{N}(0, I)$. We denote its density as $\pi(x)$. Future work on more complex simulators could consider alternative choices of x. First, x could be allowed to be dependent on θ. Second, x could be an infinite sequence of independent $\mathcal{N}(0, 1)$ random variables. This is helpful if the number of random draws required by the simulator is unknown in advance. For instance the simulator could perform a loop for a number of iterations which depends on θ in a complex fashion.

3.1. Advantages and Challenges

An advantage of parameter augmentation is avoiding the ABC curse of dimensionality. We first give an intuitive explanation. Sampling ξ from the posterior $p(\xi|y_0)$, or a close approximation, can be viewed as *controlling the simulator* through x so that $y(\xi)$ is a close match to y_0. This avoids the problem of close matches being rare when only θ is controlled, as discussed in Section 2.1. Secondly, we give a more direct explanation. Suppose we can produce a good approximation $q(\xi)$ of the augmented posterior[2] $p(\xi|y)$, which is suitable as an importance sampling proposal. This allows straightforward inference using standard importance sampling.

A difficulty of parameter augmentation is that the posterior for ξ often has a challenging form. A first challenge is that ξ usually has high dimension. MacKay (2003) argues that importance sampling is not suitable for high-dimensional targets due to the difficulty of producing suitable proposals. Recent advances in approximating distributions has made progress on this challenge. For instance Müller et al. (2019) demonstrates that normalizing flows can learn good importance sampling proposals for difficult target distributions, including high-dimensional cases. A theoretical justification for this is the universality property discussed in Section 2.3. This is the motivation for using flows in this article.

A second challenge is that parameter augmentation may result in extremely strong posterior dependence. Indeed the posterior is often a singular distribution whose mass lies on a lower dimensional manifold (Graham and Storkey 2017). Monte Carlo inference for such posteriors is challenging due to the difficulty of producing proposals exactly on an unknown manifold. We make progress by performing inference on a sequence of increasingly accurate approximate posteriors, which are all nonsingular.

3.2. Approximate Posteriors

We define approximate posterior densities for ξ, controlled by a *bandwidth* value $\epsilon > 0$, as $p_\epsilon(\xi) = \tilde{p}_\epsilon(\xi)/Z_\epsilon$ where

$$\tilde{p}_\epsilon(\xi) = \pi(\xi) \exp\left[-\frac{1}{2\epsilon^2}||y(\xi) - y_0||^2\right] \qquad (6)$$

$$Z_\epsilon = \int \tilde{p}_\epsilon(\xi) d\xi.$$

Here $\pi(\xi) = \pi(\theta)\pi(x)$ and $||y(\xi) - y_0||$ is the Euclidean distance between the simulated and observed data. Densities of the form (6) are often used in ABC and correspond to the posterior for data observed with independent $\mathcal{N}(0, \epsilon^2)$ additive errors (Wilkinson 2013).

Other similar approximate posteriors can be defined. For instance, the most common ABC approximation takes $\tilde{p}_\epsilon(\xi) = \pi(\xi)\mathbb{1}[||y(\xi) - y_0|| \leq \epsilon]$, where $\mathbb{1}$ denotes an indicator function. We prefer the approximation (6) in this article since it has the same support as $\pi(\xi)$. This makes the occurrence of zero importance weights in our algorithm less likely (or impossible if $\pi(\xi)$ has full support). Such zero weights can cause numerical problems to our algorithm.

It will be useful later to extend the unnormalized density (6) to $\epsilon = 0$ by taking

$$\tilde{p}_0(\xi) = \pi(\xi)\mathbb{1}[y(\xi) = y_0]. \qquad (7)$$

A valid density p_0 results when $Z_0 > 0$. This is typically the case when the data y is discrete, but not when it is continuous. When $Z_0 = 0$, the distribution for $\epsilon \to 0$ is singular with respect to $\pi(\xi)$.

4. Objective and Gradient

Our algorithm approximates p_ϵ using a family of densities $q(\xi;\phi)$, typically a normalizing flow. This section introduces an objective function to judge how well q approximates p_ϵ. It then discusses how to estimate the gradient of this objective with respect to ϕ. Section 5 presents our algorithm, which uses these gradients to update ϕ while also reducing ϵ.

[2]Or of an approximation $p_\epsilon(\xi)$ as defined below in (6).

4.1. Objective

Given p_ϵ, we aim to minimize the inclusive Kullback-Leibler (KL) divergence,

$$\mathrm{KL}(p_\epsilon || q) = \mathrm{E}_{\xi \sim p_\epsilon}[\log p_\epsilon(\xi) - \log q(\xi;\phi)]. \tag{8}$$

This is equivalent to maximizing a scaled negative *cross-entropy*, which is our objective,

$$\mathcal{J}_\epsilon(\phi) = Z_\epsilon \, \mathrm{E}_{\xi \sim p_\epsilon}[\log q(\xi;\phi)].$$

(Scaling by Z_ϵ avoids this intractable constant appearing in our gradient estimates below.)

The inclusive KL divergence penalizes ϕ values which produce small $q(\xi;\phi)$ when $p_\epsilon(\xi)$ is large. Hence, the optimal ϕ tends to make $q(\xi;\phi)$ nonnegligible where $p_\epsilon(\xi)$ is nonnegligible, known as the *zero-avoiding* property. This is an intuitively attractive feature for importance sampling proposal distributions. Indeed recent theoretical work shows that, under some conditions, the sample size required in importance sampling scales exponentially with the inclusive KL divergence (Chatterjee and Diaconis 2018).

Our work could be adapted to use the χ^2 divergence (Dieng et al. 2017; Müller et al. 2019), which also has theoretical links to the sample size needed by IS (Agapiou et al. 2017).

4.2. Basic Gradient Estimate

Assuming standard regularity conditions (Mohamed et al. 2020, Section 4.3.1), the objective has gradient

$$\nabla \mathcal{J}_\epsilon(\phi) = Z_\epsilon \, \mathrm{E}_{\xi \sim p_\epsilon}[\nabla \log q(\xi;\phi)]. \tag{9}$$

Using (2), an importance sampling form is

$$\nabla \mathcal{J}_\epsilon(\phi) = \mathrm{E}_{\xi \sim \lambda}\left[\frac{\tilde{p}_\epsilon(\xi)}{\lambda(\xi)} \nabla \log q(\xi;\phi)\right],$$

where $\lambda(\xi)$ is a proposal density satisfying (1). For now we assume λ is not a function of ϕ. An unbiased Monte Carlo gradient estimate of (9) is

$$g_1 = \frac{1}{N} \sum_{i=1}^{N} w_i \nabla \log q(\xi^{(i)};\phi), \tag{10}$$

where $\xi^{(i)} \sim \lambda(\xi)$ are independent samples and $w_i = \tilde{p}_\epsilon(\xi^{(i)})/\lambda(\xi^{(i)})$ (importance sampling weights). We calculate $\nabla \log q(\xi^{(i)};\phi)$ by backpropagation (see, e.g., Baydin et al. 2018).

Choice of Proposal Density. In our main algorithm, defined below in Section 5, when gradient estimates are required we have an existing estimate of ϕ (i.e., at the start of the outer for loop in Algorithm 1 we have access to ϕ_{t-1}.) Therefore, it is appealing to take $\lambda(\xi) = q(\xi;\phi)$. Since we use choices of q with full support, (1) is satisfied.

This λ is a function of ϕ, so the $\xi^{(i)}$ terms may have some dependence on ϕ. Interpreting ∇ as full differentiation could cause terms involving $\nabla \xi^{(i)}$ to appear in g_1, preventing it being an unbiased estimate of $\nabla \mathcal{J}_\epsilon(\phi)$. To avoid this henceforth we take ∇ as the gradient operator based on *partial* differentiation with respect to ϕ. (Or equivalently we take $\lambda(\xi) = q(\xi;\perp\phi)$, where $\perp$ is the "stop-gradient" operator in the notation of Foerster et al. (2018). This matches the implementation in code, as $\perp$ corresponds to the torch command `detach`.

4.3. Improved Gradient Estimates

Here we discuss reducing the variance and cost of g_1.

Truncating Weights. To avoid high variance gradient estimates we apply *truncated importance sampling* (Ionides 2008). This truncates the weights at a maximum value ω, giving truncated importance weights $\tilde{w}_i = \min(w_i, \omega)$. The resulting gradient estimate is

$$g_2 = \frac{1}{N} \sum_{i=1}^{N} \tilde{w}_i \nabla \log q(\xi^{(i)};\phi).$$

This typically has lower variance than g_1, but has some bias. See the supplement (Section A) for more details and discussion, including how we choose ω automatically, and a heuristic argument why truncation bias is not problematic.

One can also think of g_2 as replacing $\tilde{p}_\epsilon(\xi^{(i)})$ with $\min[\tilde{p}_\epsilon(\xi), \omega\lambda(\xi)]$. Thus, we attempt to optimize q in a local neighborhood of λ.

A more sophisticated alternative to truncation is Pareto smoothing the largest importance weights (Yao et al. 2018). We do not use this as it is more expensive to implement than truncation, but it would be interesting to investigate in future work.

Resampling. Calculating g_2 requires evaluating $\nabla \log q(\xi^{(i)};\phi)$ for $1 \le i \le N$. Each of these has a computational cost, but often many receive small weights and so contribute little to g_2.

To reduce this cost we can discard many low weight samples, by using *importance resampling* (Smith and Gelfand 1992) as follows. We sample $n \ll N$ times, with replacement, from the $\xi^{(i)}$s with probabilities $\tilde{s}_i = \tilde{w}_i/S$ where $S = \sum_{i=1}^{N} \tilde{w}_i$. Denote the resulting samples as $\tilde{\xi}^{(j)}$. Then an unbiased estimate of g_2 is

$$g_3 = \frac{S}{nN} \sum_{j=1}^{n} \nabla \log q(\tilde{\xi}^{(j)};\phi). \tag{11}$$

5. Algorithm

Algorithm 1 gives our main algorithm, and this section discusses various details of it. The algorithm outputs an ϵ value and a density $q(\xi;\phi)$ trained to be an importance sampling proposal for $p_\epsilon(\xi)$. We recommend using q in a final stage of importance sampling with a large sample size. While the density q is itself an approximation of p_ϵ (and therefore, of the exact posterior), in our experience it can have artifacts: see discussion at the end of Section 6.

5.1. Algorithm Overview

A summary of the algorithm follows. Steps 4–6 perform importance sampling with target p_ϵ. Steps 7–11 use the output to train ϕ, aiming for the resulting density to approximate the importance sampling target. As the algorithm progresses, step 5 reduces ϵ, aiming for the importance sampling ESS to equal M (a tuning choice). The goal is to make the importance sampling target closer to the posterior, slowly enough to avoid high variance gradient estimates.

Algorithm 1 Distilled importance sampling (DIS)

```
1: Input: importance sampling size $N$, target ESS $M$, batch size $n$, number of batches $B$.
2: Initialize $\phi_0$ (followed by pretraining if necessary) and let $\epsilon_0 = \infty$.
3: for $t = 1, 2, \dots$ do
4:     Sample $(\xi_i)_{1 \leq i \leq N}$ from $q(\xi; \phi_{t-1})$.
5:     Select a new value $\epsilon_t \leq \epsilon_{t-1}$ (see Section 5.3 for details).
6:     Calculate $w_i = \tilde{p}_\epsilon(\xi^{(i)})/q(\xi^{(i)}; \phi_{t-1})$ weights and truncate to $\tilde{w}_i$s (see the supplement, Section A, for details).
7:     for $j = 1, 2, \dots, B$ do
8:         Resample $(\tilde{\xi}^{(j)})_{1 \leq j \leq n}$ from $(\xi^{(i)})_{1 \leq i \leq N}$ using normalized $\tilde{w}_i$s as probabilities, with replacement.
9:         Calculate gradient estimate $g$ using (11).
10:        Calculate $\phi_{t,j}$ from $\phi_{t,j-1}$ and $g$ using a step of an optimization algorithm such as Adam (where $\phi_{t,0} = \phi_{t-1}$.)
11:    end for
12:    Let $\phi_t = \phi_{t,B}$.
13:    if $\epsilon = 0$ or runtime above prespecified limit then
14:        return $\epsilon, q(\xi; \phi_t)$ (typically to be used as importance sampling proposal)
15:    end if
16: end for
```

To use the importance sampling output efficiently, we sample from it (with replacement) several times to create training batches, and use each batch for one update of ϕ. We fix training batch size to $n = 100$, and number of batches B to $\lceil M/n \rceil$. So approximately M importance sampling outputs are used for training. Limiting the number of outputs used aims to avoid overfitting from too much reuse of the same training data.

An update of ϕ is done by a step of an optimization algorithm based on gradient estimates, aiming to increase $\mathcal{J}_\epsilon$. The gradient estimate is calculated from the current batch of training data, and is calculated using (11). We use the Adam optimization algorithm (Kingma and Ba 2015), and found it worked well, but many alternatives could also be used (see the review of Ruder 2016 for instance).

Steps 13–14 terminate the algorithm after a fixed runtime or once $\epsilon = 0$ is reached (when this is possible, i.e., for discrete data.) Alternatively, the termination decision could be based on approximate inference diagnostics, such as those of Yao et al. (2018) or Huggins et al. (2020).

We investigate the remaining tuning choices empirically in Section 7. For now note that N must be reasonably large since our method to update ϵ_t relies on making an accurate ESS estimate, as detailed in Section 5.3. A further summary and discussion of tuning choices appears in the supplement (Section E).

5.2. *Pretraining*

Our initial target is the prior $\pi(\xi)$, since we take $\epsilon_0 = \infty$. The initial q should be similar to this. Otherwise the first gradient estimates produced by importance sampling are likely to have high variance. To achieve this we often make use of *pretraining*. We assume the prior can easily be sampled from. Then pretraining iterates the following steps:

1. Sample $(\xi^{(i)})_{1 \leq i \leq n}$ from $\pi(\xi)$.
2. Update ϕ using gradient $\frac{1}{n} \sum_{i=1}^n \nabla \log q(\xi^{(i)}; \phi)$.

This maximizes the negative cross-entropy $\mathrm{E}_{\xi \sim \pi}[\log q(\xi; \phi)]$. We use $n = 100$, and terminate once $q(\xi; \phi)$ achieves a ESS of 75 when targeting the prior in importance sampling.

5.3. *Selecting ϵ_t*

We select ϵ_t using ESS, as in Del Moral, Doucet, and Jasra (2012). Given $(\xi_i)_{1 \leq i \leq N}$ sampled from $q(\xi; \phi_{t-1})$, the ESS value for target $p_\epsilon(\xi)$ is

$$N_{\mathrm{ESS}}(\epsilon) = \left[\sum_{i=1}^N w(\xi_i, \epsilon) \right]^2 \Bigg/ \sum_{i=1}^N w(\xi_i, \epsilon)^2$$

where $w(\xi, \epsilon) = \tilde{p}_\epsilon(\xi)/q(\xi; \phi_{t-1})$.

(Also, to avoid numerical errors, we define the ESS to be zero if all weights are zero.)

In Step 5 of Algorithm 1 we first check whether $N_{\mathrm{ESS}}(\epsilon_{t-1}) < M$, the target ESS value. If so we set $\epsilon_t = \epsilon_{t-1}$. Otherwise we set ϵ_t to an estimate of the minimal ϵ such that $N_{\mathrm{ESS}}(\epsilon) \geq M$, computed by a bisection algorithm[3]. We perform at least 50 iterations of bisection, and then terminate once $N_{\mathrm{ESS}}(\epsilon_t) \in [M - 0.01, M + 0.01]$.

5.4. *Reaching $\epsilon_t = 0$: Exact Inference*

For continuous data, the set of ξ such that $y(\xi) = y_0$ typically has probability zero[4] under any choice of $q(\theta; \phi)$. Hence, DIS cannot reach $\epsilon = 0$ since the unnormalized density $\tilde{p}_0$ from (7) will almost surely result in all weights being zero. Instead, like ABC, DIS produces increasingly good posterior approximations as $\epsilon \to 0$.

However, for discrete data, it is plausible to reach $\epsilon_t = 0$. When this happens, the algorithm produces an approximation of the augmented posterior $p_0(\xi)$ which is a suitable proposal density for exact importance sampling.

5.5. *Choice of q*

For all our examples we use an autoregressive neural spline flow for $q(\xi; \phi)$ with five bins for each variable, and bounding box $[-10, 10]$. The spline details are output by an autoregressive residual neural network (Nash and Durkan 2019) with 3 blocks, 20 hidden features and ReLU activation. More comments on the choice of q are in Section E of the supplement.

6. Illustration: Sinusoidal Simulator

As a simple illustration, consider the simulator $y(\theta, x) = -\sin\theta + x$ with independent priors $\theta \sim U(-\pi, \pi)$, $x \sim$

[3]We must sometimes bisect intervals of the form $[a, \infty]$. To do so we take the midpoint to be $a + 100$.

[4]For instance because this set has Lebesgue measure zero and q has Lebesgue dominating measure.

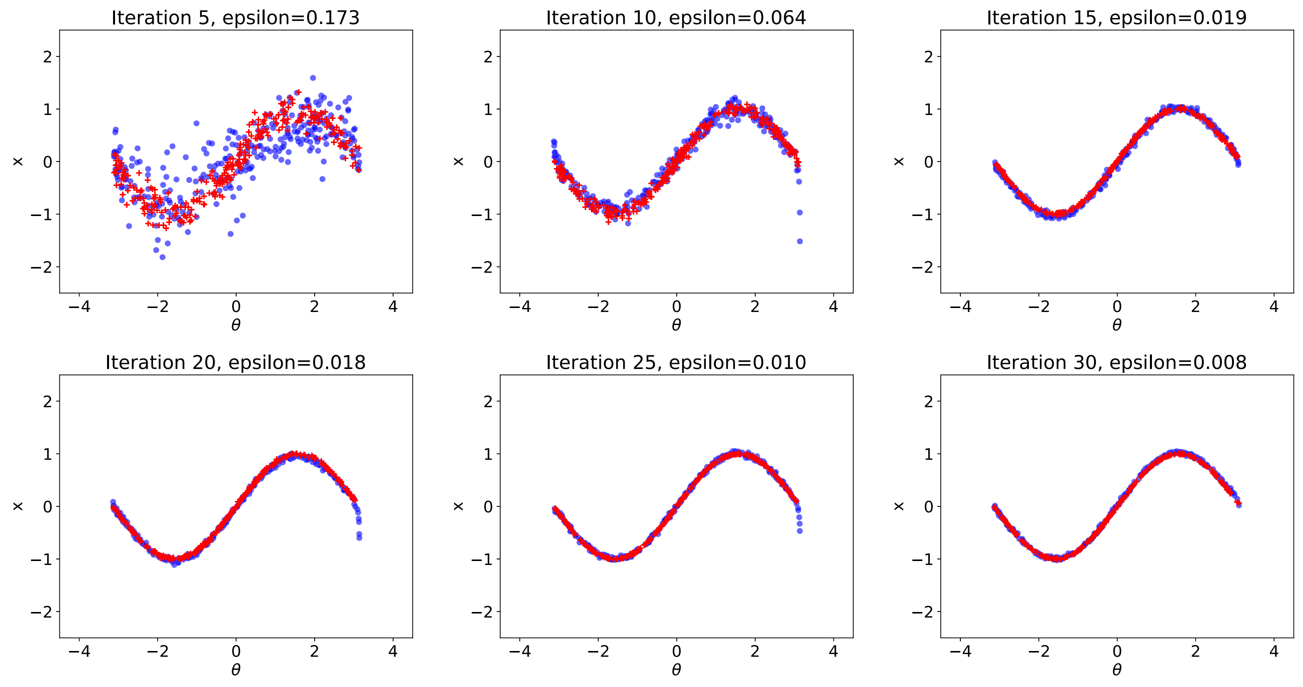


Figure 1. Sinusoidal example output. In each frame, dots are 300 samples from the current density q. Crosses are 300 samples targeting $p_\epsilon(\theta)$. These are based on the importance sampling stage of Algorithm 1 (steps 3–6) using q as a proposal, with importance resampling (see Section 4.3) used to get unweighted samples.

$\mathcal{N}(0,1)$. As observations we take $y_0 = 0$. Thus, the exact posterior is supported on the line $x = \sin\theta$.

It is helpful to reparametrize θ to a distribution with unbounded support. So we introduce $\vartheta \sim \mathcal{N}(0,1)$ and let $\theta = \pi[2\Phi(\vartheta) - 1]$ where Φ is the $\mathcal{N}(0,1)$ cumulative distribution function. We then infer $\xi = (\vartheta, x)$.

We use Algorithm 1 with $N = 4000$ training samples and a target ESS of $M = 2000$. These values give a clear visual illustration: we investigate efficient tuning choices later. We pretrain so that $q(\xi;\phi_0)$ approximates the prior.

Figure 1 shows our results. The normalizing flow quickly adapts to meet the importance sampling results, and after 30 iterations we reach $\epsilon = 0.008$, taking roughly 0.1 min. Visually, the samples lie close to the exact posterior support described above. In some panels, q has artifacts—an unwanted "tail" at its right hand end— which are removed by importance sampling. This illustrates our recommendation (see start of Section 5) to use q as an importance proposal, rather than as a posterior estimate.

7. Example: M/G/1 Queue

7.1. Model

Consider a M/G/1 queuing model of a single queue of customers. Times between arrivals at the back of the queue are $\text{Exp}(\theta_1)$. On reaching the front of the queue, a customer's service time is $U(\theta_2, \theta_3)$. All these random variables are independent. We consider a setting where only *inter-departure times* are observed: times between departures from the queue.

We sample a synthetic dataset of 20 observations from parameter values $\theta_1 = 0.1, \theta_2 = 4, \theta_3 = 5$. We attempt to infer these parameters under the prior $\theta_1 \sim U(0, 1/3), \theta_2 \sim U(0, 10), \theta_3 - \theta_2 \sim U(0, 10)$ (all independent).

7.2. Existing Methods

The M/G/1 model with inter-departure observations is a common benchmark for likelihood-free inference, which can often provide fast approximate inference. See Papamakarios, Sterratt, and Murray (2019) for a comparison of methods for an M/G/1 example. A potential disadvantage of existing likelihood-free methods is that they typically require using low dimensional summary statistics to be efficient, and this can lose some information from the data. As a likelihood-free inference baseline, we investigate a ABC-PMC algorithm. This is a modification of existing ABC-PMC methods to use the unnormalized target (6). See Section C of the supplement for full details.

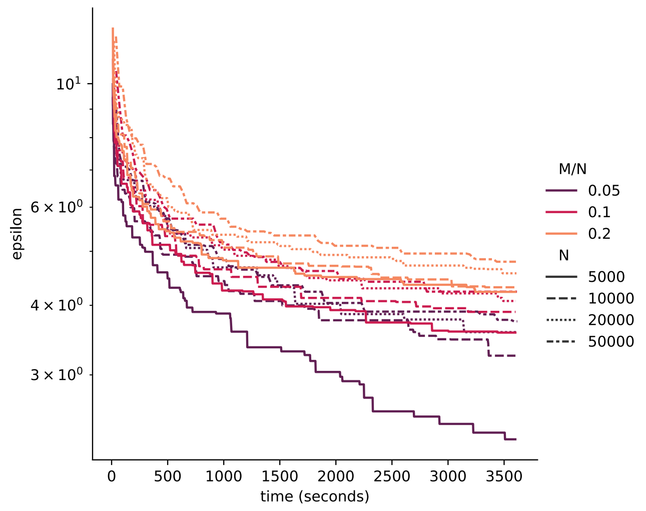
We also include results from a sophisticated MCMC scheme (Shestopaloff and Neal 2014) for this model. This provides near-exact samples from the posterior, which is a useful gold standard comparison for the approximate inference methods. A limitation of this MCMC scheme is the difficulty in adapting it to other observation regimes—for example, observing the queue length at various times (Pickands and Stine 1997)—which would be relatively easy for likelihood-free methods.

7.3. DIS Implementation

We introduce ϑ and x, vectors of length 3 and $2\dim(y)$, and take ξ as the collection (ϑ, x). Our simulator transforms these inputs to $\theta(\vartheta)$ and $y(\xi)$, with the property that $\xi \sim \mathcal{N}(0, \mathrm{I})$ outputs a sample from the prior and the model. See the supplement (Section B) for details. We pretrain q to approximate this distribution.

7.4. Results

Figure 2 (left) compares different choices of N (number of importance samples) and M (target ESS). It shows that ϵ reduces more quickly for smaller M/N and smaller N, with M/N having



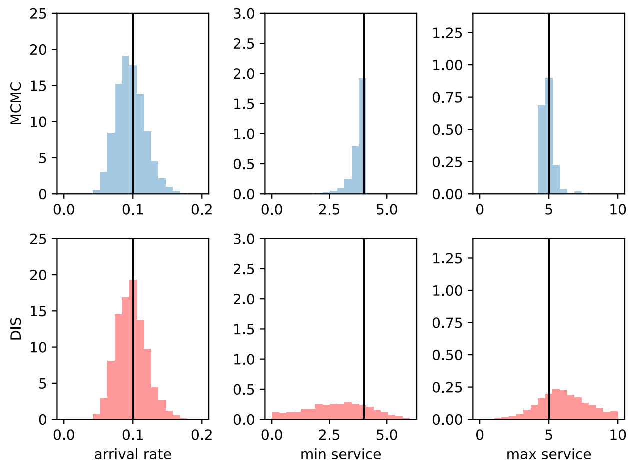


Figure 2. M/G/1 results. Left: The ϵ value reached by DIS on the M/G/1 example against computation time, for various choices of N (number of importance samples) and M/N (ratio of target effective sampling size to N). Right: Marginal posterior histograms for M/G/1 example. The DIS output shown is for $\epsilon = 2.30$, which took 60 min to reach. To produce DIS histograms we ran a final stage of importance sampling with 750,000 samples, and then used importance resampling (see Section 4.3) to get unweighted samples.

most influence. Note that small N also has the advantage of lower memory requirements.

This tuning study suggests taking N and M/N as small as possible, while ensuring M is at least a few hundred to avoid high variance in the importance sampling estimates. We use this guidance here and in the next section's example. In both these examples, the cost of a single simulator run is low. For more expensive models efficient tuning choices may differ.

Figure 2 (right) is based on the DIS results with $N = 5000$ and $M = 250$, following by a final importance sampling step with 750,000 samples[5], which takes a further 7.2 min. The results are a reasonably close match to near-exact MCMC output using the algorithm of Shestopaloff and Neal (2014). The DIS results for θ_2 and θ_3 are more diffuse than the MCMC results. Also the θ_2 DIS results lacks the sharp truncation of the right tail present in the MCMC results. (Truncation occurs at the minimum observed inter-departure time of 4.01, as the minimum service time cannot be greater than this.)

We also compare DIS to our ABC-PMC baseline, under a similar runtime. For full details of methods and results see the supplement (Section C). Table 1 summarizes the results. ABC-PMC without summary statistics targets (6), just as DIS does, but cannot reach as low an ϵ value in the same time, so it produces inaccurate estimates for all parameters. To improve ABC-PMC performance, we consider replacing the data with summaries: quartiles of the inter-departure times, as used in Papamakarios, Sterratt, and Murray (2019). This improves results for θ_1 and θ_2, but still produces severe bias for θ_3.

Table 1. Estimated posterior means and 95% quantile intervals for the M/G/1 example using various methods: MCMC (near-exact), ABC without summary statistics for $\epsilon = 6.32$, ABC with quartile summaries for $\epsilon = 0.16$, DIS for $\epsilon = 2.30$.

	θ_1 (arrival rate)	θ_2 (min service time)	θ_3 (max service time)
MCMC	0.098 (0.060, 0.142)	3.72 (2.43, 4.00)	5.0 (4.4, 6.9)
ABC (no summaries)	0.147 (0.070, 0.303)	6.72 (2.15, 9.47)	19.0 (12.4, 26.1)
ABC (summaries)	0.094 (0.056, 0.128)	3.82 (2.92, 4.26)	9.0 (7.9, 11.3)
DIS	0.098 (0.060, 0.145)	2.85 (0.20, 5.43)	6.4 (2.7, 11.1)

The ϵ values resulted from running ABC and DIS methods for roughly equal times.

8. Example: Susceptible-Infective Process on a Random Network

This section illustrates how DIS can be used to infer discrete variables. The key idea is to represent them as transformations of continuous latent variables. Such a mapping must be noninjective, and this is supported by the DIS algorithm without a need for any modifications.

We demonstrate this in the context of a spreading process model of an infectious disease on a random network, specifically a susceptible-infective (SI) compartmental model on a Erdős-Rényi network. See Dutta, Mira, and Onnela (2018) for a discussion of similar models, although not the exact one we consider.

8.1. Model

Our model represents a population of m individuals as a network. Each individual corresponds to a node. The individuals are labeled $0, 1, \dots, m-1$. An edge indicates contact between the corresponding individuals. The network is assumed to be an instance of the Erdős-Rényi random network (Erdős and Rényi 1959), so each pair of nodes form an edge with probability θ_1.

The spreading process of the infectious disease is modeled as a discrete-time process. We consider a compartmental model where at any time each individual is in one of three states: susceptible, infective or immune. Initially there is a single infective node, individual 0. Susceptible neighbors of an infective individual at time t are *exposed* and become infective at time $t+1$ with probability θ_2. Exposed individuals who are not infected instead become immune. (All infection and edge formation events are assumed independent.)

Parameters θ_1 and θ_2 have independent $U(0,1)$ prior distributions. We assume that the observations are a binary vector indicating infective status for each node/time combination.

[5]The large number of samples is to meet a reviewer request to estimate the tails in Figure 2 accurately.

8.2. Likelihood-Based Inference

The likelihood function of the model can be calculated by summing contributions from all possible networks, as detailed in the supplement (Section D). This is computationally feasible for small m, enabling near-exact posterior inference. Below, we use likelihood-based importance sampling as a benchmark for an example with $m = 5$. However, likelihood calculations become infeasibly expensive for larger m, as the number of networks to sum is $2^{m(m-1)/2}$. For instance below we also investigate an example with $m = 10$, giving $2^{45} \approx 3 \times 10^{13}$ total networks.

8.3. DIS Implementation

Latent Variables and Simulator. Similarly to Section 7, we introduce vectors of latent variables $\vartheta, x_{\text{edge}}, x_{\text{infect}}$. Here $\dim(\vartheta) = 2, \dim(x_{\text{edge}}) = \binom{m}{2}, \dim(x_{\text{infect}}) = m$. Now ξ is the collection $(\vartheta, x_{\text{edge}}, x_{\text{infect}})$.

Our simulator transforms these inputs to $\theta(\vartheta)$ and $y(\xi)$. As in Section 7, the simulator has the property that $\xi \sim \mathcal{N}(0, I)$ produces samples from the prior and model. We pretrain q to approximate this distribution. Full details of the simulator transformation are in the supplement (Section D). In brief, x_{edge} has an entry for each possible edge. An edge occurs when the corresponding entry falls below a threshold controlled by ϑ_1. Similarly x_{infect} has an entry for each individual. When that individual is exposed, then infection occurs if the corresponding entry falls below a threshold controlled by ϑ_2. (Only one random variable is needed per individual as they can only be exposed once. Afterwards they are either infective or immune.)

To summarize, as well as continuous parameters θ we also wish to infer discrete variables: presence of edges and infection status. As noted at the start of Section 8, the discrete variables are inferred by representing them as a mapping of continuous latent variables ξ.

8.4. Results

We first consider a small network with $m = 5$ nodes, observed for 5 time steps. This has $\dim(\xi) = 17$. Following Section 7 we use $N = 5000$ and $M = 250$ in DIS. As discussed in Section 3.2, it is plausible here for DIS to reach $\epsilon = 0$, since the data are discrete. Our experiment reaches $\epsilon = 0$ after 1.4 min. Then we perform importance sampling using this DIS proposal with 100,000 samples, which takes a further 1.2 min. Figure 3 shows posterior histograms for θ match those for likelihood-based importance sampling, which produces 100,000 samples in only 4 sec.

Next we consider a larger network with $m = 10$ nodes, observed for 10 time steps. Now $\dim(\xi) = 57$. As described in Section 8.2, exact likelihood calculation is infeasible here. However, DIS with $N = 5000$, $M = 250$ can perform exact inference, reaching $\epsilon = 0$ after 393 min. Figure 4 (left) shows the resulting posterior histograms for θ. Then we perform importance sampling using this DIS proposal with 20,000 samples, which takes a further 1.7 min.

DIS also outputs posterior distributions of the latent variables x_{edge} and x_{infect}, controlling the network structure and whether individuals become immune on exposure. These are summarized in Figure 4 (right). In the observed data, only individuals 1 and 7 do not become infected. The figure shows these individuals have a low posterior probability of becoming infected on exposure. Individual 0 has a moderate probability of infection on exposure. This is because they are already infective initially, so there is little information in the data on what would happen to them on exposure. The remaining individuals have a high posterior probability of becoming infected. In the data, individuals 3,6,8 are infected in the first time period. Consequently the edges from these to individual 0 have the highest posterior probabilities, as this is the only possible route of infection. Edges representing plausible routes to individuals infected later have lower probabilities, presumably as there are more possibilities for how this happens. The lowest probability edges are those which would result in individuals being infected before this is observed to happen in the data , for example, the edge $(0, 2)$.

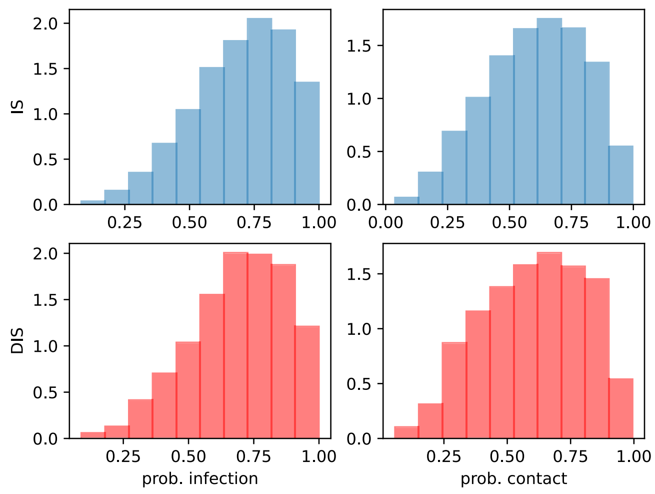


Figure 3. Output for SI example with $m = 5$, showing marginal posterior histograms for parameters θ_1 and θ_2 from IS and DIS output. To produce DIS histograms we ran a final stage of importance sampling with 100,000 samples and then used importance resampling (see Section 4.3) to get unweighted samples.

9. Conclusion

We present *distilled importance sampling* for likelihood-free inference, and show its application in several examples. An M/G/1 queuing example demonstrates that the method produces approximate results which are much more accurate than ABC, without needing the use of summary statistics. An epidemic model example shows that the method can also produce inference for discrete data. Indeed it demonstrates that for discrete data, it is possible for DIS to target the exact posterior. We also investigate tuning choices, and propose some default choices. The supplement (Section E) has a summary of tuning recommendations, and discussion of some alternative possibilities.

The remainder of the section discusses possible extensions of our methodology, as well as limitations.

Likelihood-Based Inference. This article concentrates on likelihood-free inference. DIS can also be applied to some likelihood-based inference problems. However, exploratory work shows that here it perform less well than the competing

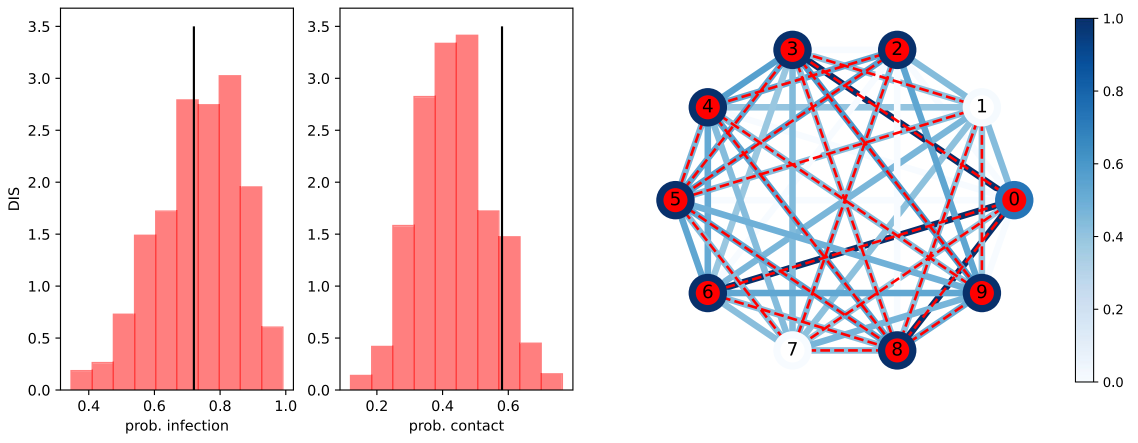


Figure 4. DIS output for SI example with $m = 10$. To produce the plots we ran a final stage of importance sampling with 20,000 samples, and then used importance resampling (see Section 4.3) to get unweighted samples. Left: Marginal posterior histograms for parameters θ_1 and θ_2. Vertical lines show the true values. Right: Posterior distribution of the network structure. Colors (in electronic version) / shades (in printed version) indicate the posterior probability of the existence of an edge and that a node will be infected upon exposure. Nodes represented by shaded circles are infective at some point, and dashed edges are those existing in the true network.

methods listed at the start of Section 1.1. See the supplement (Section F) for further discussion.

More Efficient Training. Every iteration of Algorithm 1 generates new model simulations, and uses them for a few iterations of training. This results in a computationally feasible algorithm for our examples, as the simulators are reasonably fast. However, for more expensive simulators, it would be desirable to use simulations more efficiently in the training process. For instance, training could continue with the same training data until convergence—although this risks overtraining. Alternatively, training data from all previous iterations could be reused for training—although ensuring the correct target distribution may be difficult. We plan to explore these approaches in future work.

Amortized Flows. Throughout the article, $q(\xi;\phi)$ is a generic normalizing flow producing the vector ξ. We refer to this as a *black-box* approach, since it involves no knowledge of how x is used by the simulator. This is less practical for large $\dim(\xi)$, as an increasingly large amount of simulations are required to learn such a high dimensional distribution.

An alternative is leveraging *amortization* in the flow, using knowledge about the simulator. Suppose a simulator requests a random variable. An amortized flow would determine the distribution to sample from using knowledge of: previously returned random variables (as for a black-box flow); the current simulator state; the line of code making the request. The complexity of the flow now depends on the number of sampling statements in the code, rather than the total number of samples made. This can be a big reduction in complexity when the simulator makes many iterations over a few sample statements. Similar ideas appear in *inference networks* (Le, Baydin, and Wood 2017).

Overriding Random Number Generators. For the examples in this article, we wrote custom simulator code which allowed x values sampled from q to be supplied as input. For large simulator codebases, it would be ideal to work with the original code, without needing to develop a custom version. Following Baydin et al. (2019), a possible approach is to override the random number generator that the code uses to instead provide x values proposed by q.

Use with CDE. As mentioned in Sections 1 and 2.1, one approach to likelihood-free inference is conditional density estimation (CDE). Unlike DIS, this allows amortized inference: after training it can be used for inference of many different observed datasets. Future work could combine the methods: use CDE to produce an initial approximate posterior estimate, then refine further using DIS. This makes use of complementary advantages of the methods: CDE is amortized, while DIS relies less on summary statistics.

Acknowledgments

The authors gratefully acknowledge: Alex Shestopaloff for providing MCMC code for the M/G/1 model; Sammy Ragy for spotting several typos; Andrew Golightly, Chris Williams and anonymous reviewers for helpful suggestions. This work was supported by the EPSRC under grant EPSRC EP/V049127/1 (Dennis Prangle); and Italian Ministry of University and Research (MIUR) under grant 58523_DIPECC (Cecilia Viscardi). The authors have no potential conflict of interest to report.

Supplementary Material

Distilling Importance Sampling: Supplementary Material: This document contains additional information and figures. (PDF)

ORCID

Cecilia Viscardi https://orcid.org/0000-0002-2791-7025

References

Agapiou, S., Papaspiliopoulos, O., Sanz-Alonso, D., and Stuart, A. M. (2017), "Importance Sampling: Intrinsic Dimension and Computational Cost," *Statistical Science*, 32, 405–431. [5]

Arbel, M., Matthews, A., and Doucet, A. (2021), "Annealed Flow Transport Monte Carlo," in *International Conference on Machine Learning*, pp. 318–330. [2]

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2018), "Automatic Differentiation in Machine Learning: A Survey," *Journal of Machine Learning Research*, 18, 1–43. [5]

Baydin, A. G., Shao, L., Bhimji, W., Heinrich, L., Naderiparizi, S., Munk, A., Liu, J., Gram-Hansen, B., Louppe, G., Meadows, L., Torr, P., Lee, V., Prabhat, Cranmer, K., and Wood, F. (2019), "Efficient Probabilistic Inference in the Quest for Physics Beyond the Standard Model," in *Advances in Neural Information Processing Systems* (Vol. 32). [10]

Bornschein, J., and Bengio, Y. (2014), "Reweighted Wake-Sleep," arXiv preprint arXiv:1406.2751. [2]

Chatterjee, S., and Diaconis, P. (2018), "The Sample Size Required in Importance Sampling," *The Annals of Applied Probability*, 28, 1099–1135. [5]

Cornuet, J.-M., Marin, J.-M., Mira, A., and Robert, C. P. (2012), "Adaptive Multiple Importance Sampling," *Scandinavian Journal of Statistics*, 39, 798–812. [2]

Cotter, S. L., Kevrekidis, I. G., and Russell, P. (2020), "Transport Map Accelerated Adaptive Importance Sampling, and Application to Inverse Problems Arising from Multiscale Stochastic Reaction Networks," *SIAM/ASA Journal on Uncertainty Quantification*, 8, 1383–1413. [2]

Del Moral, P., Doucet, A., and Jasra, A. (2012), "An Adaptive Sequential Monte Carlo Method for Approximate Bayesian Computation," *Statistics and Computing*, 22, 1009–1020. [6]

Dieng, A. B., Tran, D., Ranganath, R., Paisley, J., and Blei, D. (2017), "Variational Inference via χ Upper Bound Minimization," in *Advances in Neural Information Processing Systems* (Vol. 30). [5]

Duan, L. L. (2021), "Transport Monte Carlo: High-Accuracy Posterior Approximation via Random Transport," *Journal of the American Statistical Association*, DOI: 0.1080/01621459.2021.2003201. [2]

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019), "Neural Spline Flows," in *Advances in Neural Information Processing Systems* (Vol. 32). [3]

Dutta, R., Mira, A., and Onnela, J.-P. (2018), "Bayesian Inference of Spreading Processes on Networks," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474, 20180129. [8]

Elvira, V., Martino, L., and Robert, C. P. (2022), "Rethinking the Effective Sample Size," *International Statistical Review*, 90, 525–550. [3]

Erdős, P., and Rényi, A. (1959), "On Random Graphs," *Publicationes Mathematicate*, 6, 290–297. [8]

Foerster, J., Farquhar, G., Al-Shedivat, M., Rocktäschel, T., Xing, E., and Whiteson, S. (2018), "Dice: The Infinitely Differentiable Monte Carlo Estimator," in *International Conference on Machine Learning,* pp. 1529–1538. [5]

Germain, M., Gregor, K., Murray, I., and Larochelle, H. (2015), "MADE: Masked Autoencoder for Distribution Estimation," in *International Conference on Machine Learning*, pp. 881–889. [3]

Graham, M. M., and Storkey, A. J. (2017), "Asymptotically Exact Inference in Differentiable Generative Models," *Electronic Journal of Statistics*, 11, 5105–5164. [2,4]

Grazian, C., and Fan, Y. (2019), "A Review of Approximate Bayesian Computation Methods via Density Estimation: Inference for Simulator-Models," *Wiley Interdisciplinary Reviews: Computational Statistics*, 12, e1486. [1,3]

Huggins, J. H., Kasprzak, M., Campbell, T., and Broderick, T. (2020), "Practical Posterior Error Bounds from Variational Objectives," in *Artificial Intelligence and Statistics*, pp. 1792–1802. [6]

Ikonomov, B., and Gutmann, M. U. (2020), "Robust Optimisation Monte Carlo," in *International Conference on Artificial Intelligence and Statistics*, pp. 2819–2829. [2]

Ionides, E. L. (2008), "Truncated Importance Sampling," *Journal of Computational and Graphical Statistics* 17, 295–311. [5]

Jerfel, G., Wang, S., Fannjiang, C., Heller, K. A., Ma, Y., and Jordan, M. I. (2021), "Variational Refinement for Importance Sampling Using the Forward Kullback-Leibler Divergence," in *Uncertainty in Artificial Intelligence*, pp. 1819–1829. [2]

Kingma, D. P., and Ba, J. (2015), "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations*. [6]

Larrañaga, P., and Lozano, J. A. (2002), *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, New York, NY: Springer. [2]

Le, T. A., Baydin, A. G., and Wood, F. (2017), "Inference Compilation and Universal Probabilistic Programming," in *Artificial Intelligence and Statistics*, pp. 1338–1348. [10]

Li, Y., Turner, R. E., and Liu, Q. (2017), "Approximate Inference with Amortised MCMC," arXiv preprint arXiv:1702.08343. [2]

Liu, J. S. (1996), "Metropolized Independent Sampling with Comparisons to Rejection Sampling and Importance Sampling," *Statistics and Computing*, 6, 113–119. [3]

MacKay, D. J. C. (2003), *Information Theory, Inference and Learning Algorithms*, Cambridge, UK: Cambridge University Press. [4]

Marin, J.-M., Pudlo, P., Robert, C. P., and Ryder, R. J. (2012), "Approximate Bayesian Computational Methods," *Statistics and Computing*, 22, 1167–1180. [1,3]

Meeds, T., and Welling, M. (2015), "Optimization Monte Carlo: Efficient and Embarrassingly Parallel Likelihood-Free Inference," *Advances in Neural Information Processing Systems* (Vol. 28). [2]

Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. (2020), "Monte Carlo Gradient Estimation in Machine Learning," *Journal of Machine Learning Research*, 21, 1–62. [5]

Müller, T., Mcwilliams, B., Rousselle, F., Gross, M., and Novák, J. (2019), "Neural Importance Sampling," *ACM Transactions on Graphics (TOG)*, 38, 1–19. [2,4,5]

Naesseth, C. A., Lindsten, F., and Blei, D. (2021), "Markovian Score Climbing: Variational Inference with $KL(p||q)$," in *Advances in Neural Information Processing Systems*, Vol. 33, pp. 15499–15510. [2]

Nash, C., and Durkan, C. (2019), "Autoregressive Energy Machines," in *International Conference on Machine Learning*, pp. 1735–1744. [3,6]

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021), "Normalizing Flows for Probabilistic Modeling and Inference," *Journal of Machine Learning Research*, 22, 1–64. [1,3]

Papamakarios, G., Sterratt, D., and Murray, I. (2019), "Sequential Neural Likelihood: Fast Likelihood-Free Inference with Autoregressive Flows," in *Artificial Intelligence and Statistics*, pp. 837–848. [7,8]

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019), "Pytorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, Vol. 32. [2]

Pickands III, J., and Stine, R. A. (1997), "Estimation for an M/G/∞ Queue with Incomplete Information," *Biometrika*, 84, 295–308. [7]

Prangle, D., Everitt, R. G., and Kypraios, T. (2018), "A Rare Event Approach to High-Dimensional Approximate Bayesian Computation," *Statistics and Computing*, 28, 819–834. [2,3]

Robert, C. P., and Casella, G. (2013), *Monte Carlo Statistical Methods*, New York, NY: Springer. [3]

Rubinstein, R. (1999), "The Cross-Entropy Method for Combinatorial and Continuous Optimization," *Methodology and Computing in Applied Probability*, 1, 127–190. [2]

Rubinstein, R. Y., and Kroese, D. P. (2016), *Simulation and the Monte Carlo Method*, Hoboken, NJ: John Wiley & Sons. [3]

Ruder, S. (2016), "An Overview of Gradient Descent Optimization Algorithms," arXiv preprint arXiv:1609.04747. [6]

Shestopaloff, A. Y., and Neal, R. M. (2014), "On Bayesian Inference for the M/G/1 Queue with Efficient MCMC Sampling," arXiv preprint arXiv:1401.5548. [2,7,8]

Smith, A. F. M., and Gelfand, A. E. (1992), "Bayesian Statistics without Tears: A Sampling–Resampling Perspective," *The American Statistician*, 46, 84–88. [5]

Wilkinson, R. D. (2013), "Approximate Bayesian Computation (ABC) Gives Exact Results UNDER the Assumption of Model Error," *Statistical Applications in Genetics and Molecular Biology*, 12, 129–141. [4]

Yao, Y., Vehtari, A., Simpson, D., and Gelman, A. (2018), "Yes, but Did It Work?: Evaluating Variational Inference," in *International Conference on Machine Learning*, pp. 5581–5590. [5,6]