Statistica Sinica **33** (2023), 1507-1532
doi:https://doi.org/10.5705/ss.202020.0348

MARGINAL BAYESIAN POSTERIOR INFERENCE USING RECURRENT NEURAL NETWORKS WITH APPLICATION TO SEQUENTIAL MODELS

Thayer Fisher, Alex Luedtke, Marco Carone and Noah Simon

University of Washington

Abstract: In Bayesian data analysis, it is often important to evaluate quantiles of the posterior distribution of a parameter of interest (e.g., to form posterior intervals). In multi-dimensional problems, when non-conjugate priors are used, this is often difficult, generally requiring either an analytic or a sampling-based approximation, such as the Markov chain Monte Carlo, approximate Bayesian computation, or variational inference. We discuss a general approach that reframes this as a multi-task learning problem and uses recurrent deep neural networks (RNNs) to approximately evaluate the posterior quantiles. Because RNNs carry information along a sequence, this application is particularly useful in time series. An advantage of the risk-minimization approach is that we do not need to sample from the posterior or calculate the likelihood. Lastly, we illustrate the proposed approach using several examples.

Key words and phrases: Bayesian deep learning, machine learning, quantile estimation.

1. Introduction

We consider a common Bayesian scenario in which we have a data set, X, generated from some parametrically specified distribution $(X \,|\theta, \eta) \sim P_{\theta,\eta}$. Here, we assume that $\theta \in \mathbb{R}$ is a one-dimensional parameter of interest. For simplicity, we assume that the nuisance parameter $\eta \in \mathbb{R}^p$ is finite-dimensional, although this is not essential; in our discussion, we explain how to handle an infinite-dimensional η. We further assume that (θ, η) follows a known prior distribution g, that is, $(\theta, \eta) \sim g$. In analyzing our data set X, we would like to evaluate the posterior distribution for θ:

$$dP\left(\theta \,|\, X\right) = \frac{\int_\eta dP_{\theta,\eta}(X) g(\theta,\eta) d\eta}{\int_{\theta,\eta} dP_{\theta,\eta}(X) g(\theta,\eta) d\theta d\eta}.$$

Corresponding author: Thayer Fisher, Department of Biostatistics, University of Washington, Seattle, WA 98195-0005, USA. E-mail: thayerf@uw.edu.

The integral in the denominator is often analytically intractable. When the likelihood function, $dP_{\theta,\eta}(X)$, can be calculated efficiently (up to a normalizing constant), one can draw samples from the posterior $dP(\theta \mid X)$ using stochastic simulation techniques, such as Markov chain Monte Carlo (MCMC) and rejection sampling (Asmussen and Glynn (2007)). From these samples, any posterior summary can be computed. However, there are a number of situations in which the likelihood is difficult to calculate, but it is relatively straightforward to sample (η, θ) from g and X from $P_{\theta,\eta}$. Approximate Bayesian computation (ABC) methods have been developed for inference in this regime (e.g., Marjoram et al. (2003)). These methods rely on simulating data from a joint prior-likelihood distribution, and computing the empirical distribution of parameters drawn from data that are "close" to the observed data in some sense. However, because ABC and more classical stochastic simulation techniques often use some form of rejection sampling, they may have trouble with problems that are complex or high dimensional. Furthermore, defining "close" as it pertains to the sampled data X is nontrivial in many cases. For even a moderately large X, informative summary statistics must be constructed to reduce the rejection rate, otherwise the problem becomes intractable. If the summary statistics are not suitable for the particular setting, the approximation of the posterior will be poor (Prangle (2015)).

In this work, we frame the calculation of posterior quantiles as an optimization problem. We consider the posterior quantile functions as risk minimizers with respect to particular losses. We approximate the function that minimizes this risk by restricting our optimization to the class of functions that can be represented by recurrent neural networks (RNNs) (Rumelhart, Hinton and Williams (1988)). Because we expect the posterior quantile to "update" as more data become available, the recurrent structure is advantageous. Moreover, the recurrent framework allows us to evaluate posterior quantiles for data sets of arbitrary length. Provided that our network architecture is adequate for the specific problem, our function class is sufficiently rich, and the risk minimizer over this class is close to the true posterior quantile function. We discuss how stochastic subgradient optimization can be used to find local optima over this class. In particular, by simulating parameters/data from our prior/likelihood, our neural network can update its parameter estimates to "learn" how to evaluate posterior quantiles. We refer to this as a "statistical meta-learning procedure," because it uses simulated data to learn a Bayesian-optimal statistical rule.

Another obvious advantage of RNNs is their natural compatibility with time series data, where the contribution of each observation to a parameter of interest can depend on previous or later observations. RNNs share information across

recurrent steps, making them naturally suited to handling this type of problem. Furthermore, many Bayesian time series are difficult, owing to the intractability of the likelihood (see Ghahramani (2001) for some examples). Avoiding an approximation of the entire posterior distribution is favorable in these scenarios when our target is credible intervals, for example.

Other works have proposed using deep learning in similar contexts. The idea of using an RNN as a meta-learner was first introduced by Hochreiter, Younger and Conwell (2001). Jiang et al. (2018) use deep neural networks to approximate posterior summaries. In particular, they approximate the posterior mean of a functional in high-dimensional problems. Creel (2017) builds on this work by applying this approach to particular econometric models, using feedforward neural networks to estimate the posterior means. Chan et al. (2018) use exchangeable neural networks to model population genetic data. They train on the fly, simulating new data each time they update the weights of their neural network. However, they approximate the entire posterior using a parametric approach.

2. Reframing as an Optimization Problem

For any $t \in (0,1)$, let $Q^t(X)$ denote the tth quantile of the posterior of θ under a prior g and a likelihood $P_{\theta,\eta}$. Our goal is to learn the quantile functions $\{\Phi(t) := Q^t(\cdot) \text{ for all } t \in \mathcal{T}\}$, where $\mathcal{T} \subseteq (0,1)$ either contains finitely many elements or is equal to $(0,1)$.

The quantity $Q^t(X)$ is the solution in q to

$$P\left(\theta \leq q \mid X\right) = t,$$

where we assume for simplicity that this posterior is continuous.

Our approach hinges on the fact that Q^t can be written as the solution to an optimization problem. In particular,

$$Q^t(X) = \underset{Q}{\operatorname{argmin}}\, \mathrm{E}\left[\rho_t\left(\theta - Q(X)\right)\right], \tag{2.1}$$

where $\rho_t(u) = u\left(t - I\left\{u < 0\right\}\right)$ is the asymmetric L_1 norm (or "pinball loss," as referred to by Koenker and Hallock (2001)), and the minimization is taken over all functions that are measurable with respect to $\sigma(X)$ (the sigma-algebra generated by the data). Note that when $t = 0.5$, this loss is precisely a scaled L_1 loss.

Solving (2.1) directly is intractable, in general, over the space of all measurable functions. Instead, we restrict the problem to a rich finite-dimensional

subclass, namely, those functions that can be represented by deep RNNs, which is known to be a very rich class (Cybenko (1989); Bach (2017)). In this case, our optimization problem consists of finding

$$\begin{aligned}\beta^* &= \underset{\beta}{\operatorname{argmin}}\, \mathrm{E}\left[\rho_t\left(\theta - Q_\beta(X)\right)\right] \\ &= \underset{\beta}{\operatorname{argmin}} \int_{\theta,\eta,\mathcal{X}} \rho_t\left(\theta - Q_\beta(X)\right) dP_{\theta,\eta}(X) g(\theta,\eta) d\theta d\eta,\end{aligned} \tag{2.2}$$

where Q_β is a deep RNN that takes as input each of the n observations in X and outputs a single value, and β is a vector of the weight and bias parameters in the neural network. For this network to be an effective estimator, we may need a large number of parameters: the network needs to learn both how to combine inputs to get the output, and what values to store in memory across recurrent steps.

2.1. Relation to quantile regression

The asymmetric L_1 norm described above is most commonly used in quantile regression. However, in general, a quantile regression is performed in a frequentist context, where the goal is to estimate a quantile or the distribution of a conditional outcome given a set of features. Hence, the outcomes and covariates are not fixed. In fact, the only similarity between our proposed method and a quantile regression is the use of a pinball loss. We are attempting to learn one or many posterior quantiles as functions of data simulated from the prior and the likelihood, $Q^t : \mathcal{X} \to \mathbb{R}$, as a minimizer of (2.1) over the joint distribution of (X, θ), $\int_\eta dP_{\theta,\eta}(X) g(\theta,\eta) d\eta$. We do this without ever engaging with a single, fixed, observed data set, but rather by engaging with many simulated data sets. Reframing the derivation of the posterior quantile in this way allows us to leverage deep learning for the posterior quantile approximation.

2.2. Motivation for Recurrent Structure

2.2.1. Time series and sequential models

RNNs are advantageous for posterior inference in sequential models because they pass information sequentially, by design. For illustrative purposes, consider the following second-order moving-average model described in Section 3.5:

$$X_j \sim Z_j + \theta_1 Z_{j-1} + \theta_2 Z_{j-1}.$$

The observed sequence X depends on the parameters of interest, $\{\theta_1, \theta_2\}$, only through an unobserved, latent sequence Z.

For most choices of likelihood over Z and choices of prior over θ, the likelihood $P(X \mid \theta)$ is not tractable. Furthermore, the contribution of X_j to the posterior distribution depends on the values of X_{j-1} and X_{j-2}. Because we cannot sample from the posterior directly for nontrivial choices of the likelihood and prior, and because the data are sequential, an RNN is a good choice for a posterior estimation. Later, in Section 3.5, we show that our proposed method outperforms several others in terms of credible interval estimation.

2.2.2. Canonical exponential family

To motivate the choice of an RNN over a simpler multilayer perceptron, we consider a d-parameter canonical exponential family indexed by θ with a conjugate prior on θ, namely,

$$p(x|\theta) = h(x)\exp(\theta^\top T(x) - g(\theta)),$$
$$p_{\eta_0,\gamma_0}(\theta) = k(\eta_0,\gamma_0)\exp(\eta_0(\gamma_0^\top\theta - g(\theta))),$$

for $\theta \in \mathbb{R}^d$. It follows that, if $X = \{x_1,\dots,x_n\}$, where $X_1,\dots,X_n \overset{\text{i.i.d.}}{\sim} p(\cdot|\theta)$,

$$p(\theta|X) = p_{\eta',\gamma(X)}(\theta),$$

with $\eta' = n + \eta_0$ and $\gamma(X) = \sum_{i=1}^n T(x_i)/(n+\eta_0) + (\eta_0/(n+\eta_0))\gamma_0$. By the factorization theorem, in this simple setting, there exists a $(d+1)$-parameter sufficient statistic for $\theta|X$, namely, $[\sum_{i=1}^n T(X_i), n]$. Therefore, we can write Q^t recurrently as

$$Q^t(X) = h\left[x_n, f(x_{n-1}, f(\cdots f(x_1,\beta)\cdots))\right],$$

for some fixed $\beta \in \mathbb{R}^{d+1}$, $h : \mathbb{R}^{d+2} \to \mathbb{R}$, and some $f : \mathbb{R}^{d+2} \to \mathbb{R}^{d+1}$. Here $f(x_i, \sum_{j=1}^{i-1} T(x_j), i-1) = \{\sum_{j=1}^{i} T(x_j), i\}$. We can think of f as a "memory" function that tracks the sufficient statistic across recurrent steps. In contrast, $h(\sum_{i=1}^n T(X_i), n) = F^{-1}(t)$, where F^{-1} is the quantile function of $p_{\eta',\gamma'}$. Thus, h calculates $Q^t(X)$ from the sufficient statistic. Note that increasing d is equivalent to increasing the size of the memory across the recurrent steps.

While it is not true that we can write $Q^t(X)$ in this way for a general likelihood and prior, we hope that for some sufficiently large memory size, there exist an h and f such that

$$Q^t(X) \approx h\left[x_n, f(x_{n-1}, f(\cdots f(x_1,\beta)\cdots))\right].$$

In fact, note that if the size of the memory exceeds the number of observations, the set of order statistics is sufficient.

An advantage of the recurrent structure is that it allows us to approximate a posterior quantile for a data set of arbitrary length. When the data are independent and identically distributed, this is particularly useful, because we are learning how to "update" the posterior as the number of observations grows. Using a feedforward architecture, one can only obtain posterior quantile approximations for data sets of a fixed size that are compatible with the input layer of the network.

This proposed RNN can take in a data set and return an approximation to a single posterior quantile. It is also possible to design a network that learns more than a single quantile, and potentially all quantiles. These extensions are described in what follows.

2.3. Discretized multi-task learning

We first extend our procedure to simultaneously approximate $\{Q^t\}$ for a discrete collection of t with a single network. Let $\mathcal{T}$ denote that discrete set of quantiles, and write $m_{\mathcal{T}} = |\mathcal{T}|$. We use a multi-task learning network architecture (Ruder (2017)) with shared hidden nodes and internal weights, and a set of $m_{\mathcal{T}}$ task-specific output nodes leading from the shared, final hidden layer to approximate the $m_{\mathcal{T}}$ quantiles. Let β_{in} denote the shared internal weights, and let $\beta_{out}(t)$ denote the task-specific output weights used to estimate Q^t. In this case, to find the optimal weights β^*_{in} and $\beta^*_{out}(t)$, $t \in \mathcal{T}$, we must find

$$\underset{\beta_{in},\beta_{out}(t):t\in\mathcal{T}}{\operatorname{argmin}} \sum_{t\in\mathcal{T}} w_t \, \mathrm{E}\left[\rho_t\left(\theta - Q_{\beta_{in},\beta_{out}(t)}(X)\right)\right], \tag{2.3}$$

with $w_t > 0$ denoting predefined costs. For example, one might take $w_t = 1$, for all $t \in \mathcal{T}$. This *hard sharing* multi-task learning framework that shares internal nodes across tasks seems sensible, because intermediate representations of the data that are useful for posterior calculations are likely not quantile-specific. However, one might consider modifying this to allow a "final stage," with several hidden layers specific to each quantile.

2.4. Continuous quantile learning

We further extend this to construct a single network that approximates all quantiles of the posterior simultaneously. To begin, we consider the optimization problem

$$\mathcal{Q} = \underset{Q}{\operatorname{argmin}} \int_0^1 w(t) \, \mathrm{E}\left[\rho_t\left(\theta - Q(t, X)\right)\right] dt, \tag{2.4}$$

where w is any positive integrable function on $(0, 1)$. The solution to (2.4), $\mathcal{Q}$, is a function that takes t and X as inputs, and returns the tth posterior quantile, that is, $\mathcal{Q}(t, X) = Q^t(X)$, following directly from (2.1). By approximating the problem in (2.4) using neural networks, we can build a single network to simultaneously approximate all quantiles.

Without loss of generality, we assume that $\int_0^1 w(t)dt = 1$. Then, for random T drawn from density w, we can rewrite (2.4) as

$$\mathcal{Q} = \underset{Q}{\operatorname{argmin}} \, \mathrm{E}\left[\rho_T\left(\theta - Q(T, X)\right)\right], \tag{2.5}$$

where the expectation is now over T, θ, and X.

As noted above, we approximate the solution to (2.4) using a deep neural network by solving the optimization problem

$$\beta^* = \underset{\beta}{\operatorname{argmin}} \, \mathrm{E}\left[\rho_T\left(\theta - Q_\beta(T, X)\right)\right], \tag{2.6}$$

where Q_β is a network that takes as input a data set and the selected quantile, and returns a single output. The architecture of this network is similar to the single quantile problem described in Section 2, except that T is included as an additional input at every recurrent step.

2.5. Optimization

Let ℓ_1, ℓ_M, and ℓ_C denote the losses in criteria (2.2), (2.3), and (2.6), respectively. We can optimize all three using standard stochastic subgradient-based methods (Nedic and Bertsekas (2001)). In particular, for the loss (2.2), the subdifferential of our loss, $\nabla\ell_1(\beta)$, consists of

$$\nabla\ell_1(\beta) \ni -\mathrm{E}\left[\dot\rho_t\left(\theta - Q_\beta(X)\right)\nabla Q_\beta(X)\right],$$

where $\nabla Q_\beta(X)$ is a subgradient of our network, and $\dot\rho$, defined pointwise as $\dot\rho_t(u) = tI(u > 0) + (1 - t)I(u < 0)$, is a subgradient of the quantile loss function. There are many possible subgradients, depending on how one defines $\dot\rho_t(0)$. Stochastic subgradients can be calculated by sampling a (θ, η) pair from g and a data set X from $P_{\theta,\eta}$, and then plugging these into

$$\hat\nabla\ell_1(\beta) = \dot\rho_t\left(\theta - Q_\beta(X)\right)\nabla Q_\beta(X). \tag{2.7}$$

By linearity, stochastic subgradients can be calculated similarly for the multi-task learning criterion, ℓ_M.

For continuous quantile learning, our expectation-based formulation, ℓ_C, allows us to easily calculate stochastic subgradients. In particular, given a sampled (θ, η) pair, a data set X sampled from $P_{\theta,\eta}$, and T sampled from w, we can calculate

$$\hat{\nabla}\ell_C(\beta) = \dot{\rho}_T\left(\theta - Q_\beta(X)\right)\nabla Q_\beta(X), \tag{2.8}$$

and note that $\mathrm{E}[\hat{\nabla}\ell_C(\beta)]$ is in the sub-differential of ℓ_C at β.

The full approximation procedure for (2.2) is described in Algorithm 1. We repeatedly sample (θ, X) pairs, and then use these pairs to train our neural network. Note that, in general, steps 3 and 4 use back propagation. We can extend this algorithm using linearity for the multi-task learning criterion ℓ_M, and having the network output one prediction for each quantile of interest. For the optimization described by (2.6), we need only sample T from w before we calculate the gradient in step 3. Although we describe a simple stochastic gradient descent, extensions to more complex stochastic optimization techniques that use adaptive learning rates/momentum (e.g., Kingma and Ba (2014); Smith and Topin (2017)) are usually employed in simulations in practice.

Algorithm 1 Learn $Q^t(\cdot)$.

Require: m = batch size, k = learning rate
for $0 \leq i \leq$ total iterations **do**
1. Simulate $(\theta_1, \eta_1), \ldots (\theta_m, \eta_m) \overset{\text{i.i.d.}}{\sim} g(\theta, \eta)$
2. Simulate $X_1 \sim P_{\theta_1,\eta_1}, \ldots, X_m \sim P_{\theta_m,\eta_m}$
3. $\hat{\nabla}\ell(\beta) \leftarrow (1/m)\sum_{j=1}^{m}\dot{\rho}_t\left(\theta_j - Q_\beta(X_j)\right)\nabla Q_\beta(X_j)$
4. $\beta \leftarrow \beta - k\ \hat{\nabla}\ell(\beta)$
5. $i \leftarrow i + 1$

end for

3. Examples

In this section, we consider three examples. In the first, we simply estimate the median of i.i.d. Gaussian observations using a Gaussian prior, perform a multi-task estimation of the posterior deciles, and learn the posterior quantiles continuously. Here, the posterior quantiles have a simple closed form, so evaluating performance is straightforward. For the second example, we approximate the posterior median for the maximum component in a mixture of finite mixtures (MFM). Finally, we approximate the posterior median for the basic reproduction number in a Bayesian stochastic SIR model.

For all examples, the optimization is performed using `Adam` (Kingma and Ba (2014)), as implemented in `TensorFlow` (Abadi et al. (2016)), with subgradients

calculated using mini-batches. We evaluate the performance in all simulation settings based on the mean loss, or estimated risk, over a held-out test set. The error bars at the conclusion of training represent a 95% confidence interval of the risk of the neural network, based on the sample standard error of a held-out test set. Where appropriate, we compare our neural networks with Stan, another general-purpose posterior estimation tool, in addition to comparing them with a specialized, problem-specific tool.

3.1. Gaussian example

We consider a simple example with i.i.d. observations $x_1, \ldots, x_n$ drawn from a $N(\theta, 1)$ distribution, with θ drawn from a $N(0, \sigma^2)$ prior. Here, we would like posterior quantiles for θ given our data. In this case, it is well known that

$$\theta \,|\, x_1, \ldots, x_n \;\sim\; N\left(\frac{\bar{x}\sigma^2}{1/n + \sigma^2}, \left(n + \sigma^{-2}\right)^{-1}\right),$$

where $\bar{x} = n^{-1} \sum_i x_i$. Thus,

$$Q^t(X) = \frac{\bar{x}\sigma^2}{1/n + \sigma^2} + \frac{\Phi^{-1}(t)}{\sqrt{n + \sigma^{-2}}}.$$

Furthermore, letting $X_j = \{x_1, \ldots, x_j\}$, we have, for $j \geq 2$,

$$Q^t(X_j) = \frac{(((j-1)/j)\bar{x}_{j-1} + x_j)\sigma^2}{1/j + \sigma^2} + \frac{\Phi^{-1}(t)}{\sqrt{j + \sigma^{-2}}}.$$

Thus, in this simple case, there exists an exact recurrent update step that uses a two-dimensional sufficient statistic, $\{\bar{x}_j, j\}$. Although this case is quite basic, it is illustrative, and because we have access to the exact value of $Q^t(X)$, it is trivial to evaluate the performance of our approximated posterior quantile.

We consider approximating the posterior median. In this specific case, we can assess the quality of our network by comparing the pinball loss of our network to that of the true posterior median of a held-out test set.

We ran simulations using $n = 100$ observations, where θ has prior variance $\sigma^2 = 1/100$ (equal to the conditional variance of $\bar{x}$). We use the mini-batch `Adam` for the optimization, where each mini-batch contains 100 data sets, with a learning rate of 10^{-2}. Our network has 32 nodes per hidden layer, and four hidden layers with ReLU activation functions. We evaluate the risk on a held-out test set of 500 data sets. The performance is summarized in Figure 1.

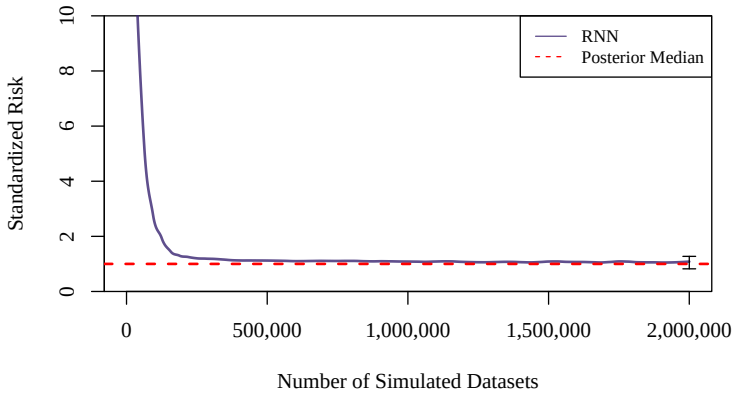

Figure 1. Risk curve for the estimation of the posterior median in a Gaussian prior and Gaussian likelihood simulation scenario. The risk of the true minimizer, the posterior median, is standardized to be equal to one (red). Here, the posterior median has a closed form. Our estimator has about 2% excess risk when compared with the Bayes estimator.

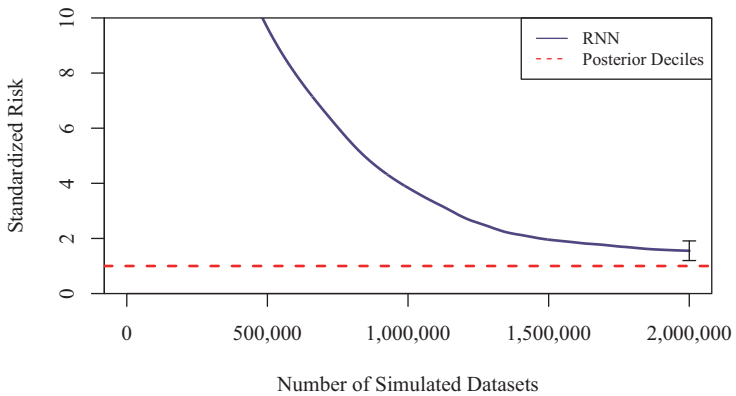

Figure 2. Risk curve for the estimation of the posterior deciles in a Gaussian prior and Gaussian likelihood simulation scenario. The risk of the true minimizer, the posterior deciles, is in set to one (red). Here, the posterior deciles have a closed form. Our estimator has about 30% excess risk when compared with the Bayes estimator.

We also ran simulations using an identical prior and likelihood, but this time attempting to simultaneously learn all nine posterior deciles, with a uniform weight on the loss over the nine tasks, as described in (2.3). The performance is summarized in Figure 2. We compare the results with the risk of the true posterior deciles, and obtain a risk that is close to the true risk of the posterior deciles. We evaluate the performance on a test set of 500 held-out data sets.

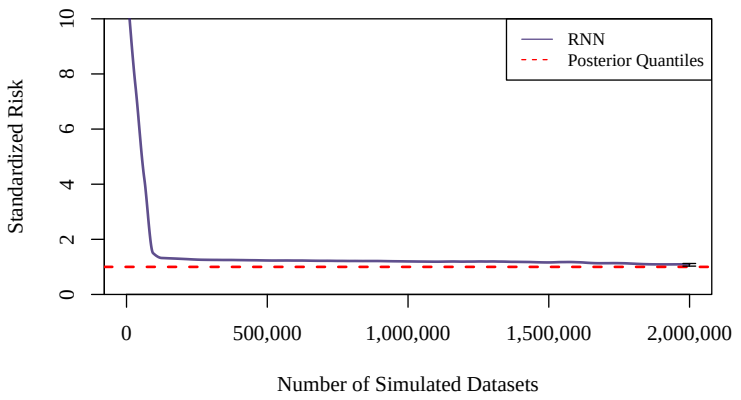

Figure 3. Risk curve for the estimation of the posterior quantiles in a Gaussian prior and Gaussian likelihood simulation scenario. The quantile estimated for each of the 1500 simulated data sets is sampled uniformly at random from the unit interval. The risk of the true minimizer, the posterior quantiles, is set to one (red). Here, the posterior quantiles have a closed form. Our estimator has about 5.6% excess risk when compared with the Bayes estimator.

Finally, we ran simulations using the same prior and likelihood as above, but where the target of the estimation was a random quantile sampled uniformly on the unit interval, using the loss described in (2.6). We included the target quantile as an input to our neural network at each recurrent step. We evaluated the performance on a test set of 500 held-out data sets and 500 held-out random quantiles. In Figure 3, we compare the average loss over this test set with that of the corresponding posterior quantiles, which have a closed form.

3.2. MFM example

Here, we consider a slightly more complex case in which i.i.d. observations $x_1, \ldots, x_n$ are drawn from a $(1/k)\sum_{i=1}^{k} N(\theta_i, 0.01)$ distribution, with θ drawn from a $N(0, 0.25)$ prior, and k drawn from a Pois(4) prior, shifted to have a minimum of one. The choice of variance for the prior and the likelihood ensure some separation between the components.

Our estimation target is the posterior median of $\theta_{(k)}$, the maximum component. Note that this parameter is sensitive to the number of components, with the distribution of $\theta_{(k)}|X, k$ depending heavily on k. This makes procedures that try to determine the number of clusters, such as the BIC, inaccurate.

We ran simulations using $n = 250$ observations. We used mini-batch `Adam` for optimization, where each mini-batch contains 150 data sets, with a learning

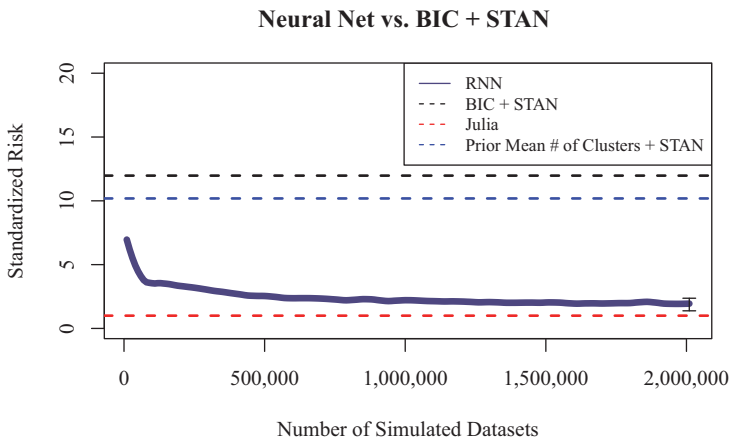

Figure 4. Risk curve for the estimation of the posterior median in a Gaussian prior and Gaussian mixture likelihood simulation scenario, with a prior on the number of mixture components. We compare the results with those of Stan, using two strategies to select the number of components, and with those of a specialized Julia package, BayesianMixtures. Our estimator has 87% excess risk when compared with BayesianMixtures, though it is much improved over our Stan implementations. A better architecture and more training may lead to better results.

rate of 10^{-4}. We used a network with 32 nodes per hidden layer, and four hidden layers.

The reason for the larger mini-batch in this simulation setting is the increased variability in the gradient due to the variable number of components. A larger mini-batch size makes the stochastic optimization more stable. The performance is summarized in Figure 4. We compare the results with those of a Julia package specialized for obtaining approximate posteriors for this model class (Miller and Harrison (2018)), and with those of Stan (Carpenter et al. (2016)), using two strategies to estimate the number of components. The first strategy uses the BIC to estimate the number of clusters, and the second uses the prior mean number of clusters. Stan proceeds with the number of clusters fixed. Because our parameter of interest is sensitive to the number of clusters, the Stan posterior estimation framework performs poorly. Our neural network approaches the risk of the approximate posterior median given by a specialized package within about 15,000 simulated data sets. We evaluated the performance on a held-out test set of 500 datasets.

3.3. Stochastic SIR example

Here, we consider a stochastic SIR model. In this setting, we observe a disease epidemic, and would like to estimate the posterior median for the basic reproduction number, R_0. A stochastic SIR model is a continuous time stochastic process that models how a disease interacts with a population of size N. At each time t, there are a number of susceptible individuals, $S(t)$, a number of infected individuals, $I(t)$, and a number of recovered individuals, $R(t)$. The disease is modeled according to the following differential equations:

$$\begin{aligned}\frac{\partial S(t)}{\partial t} &= -\frac{\beta S(t)I(t)}{N} + \sqrt{\frac{\beta S(t)I(t)}{N}}\omega_1(\partial t)\\ \frac{\partial I(t)}{\partial t} &= \frac{\beta S(t)I(t)}{N} - \gamma I(t)\\ &\quad - \sqrt{\frac{\beta S(t)I(t)}{N}}\omega_1(\partial t) + \sqrt{\gamma I(t)}\omega_2(\partial t),\end{aligned}$$

where β is the infection rate, γ is the recovery rate, and ω_1 and ω_2 are standard Wiener processes (Allen (2017)). The basic reproduction number, R_0, is the number of expected new infections an individual generates over the course of their disease in a fully susceptible population, β/γ. This parameter, and its posterior quantiles, are of interest in this problem (Clancy and O'Neill (2008)), because $R_0 > 1$ means the disease is likely to become an epidemic.

We simulate from $t = 0$ to $t = 100$ using finite differences with $\partial t = 0.01$. However, we observe only integer values of t, for a total of 101 observations. In this setting, we observe both $S(t)$ and $I(t)$, though it is simple to reduce the observed data to only $R(t)$, for example. We specify the following priors for β and γ:

$$\begin{aligned}\beta &\sim \Gamma(9, 0.05),\\ \gamma &\sim \Gamma(3, 0.05).\end{aligned}$$

We selected these priors based on Clancy and O'Neill (2008), tuning the values so that the sample paths express a variety of disease dynamics. We used mini-batch `Adam` for optimization, where each mini-batch contains 100 data sets, with a learning rate of 10^{-4}. We used a network with 32 nodes per hidden layer, and four hidden layers.

The performance is summarized in Figure 5. We compare the results with those of Stan, where we have attempted to solve this problem using finite differences. Our neural network approaches the risk of Stan within about 30,000

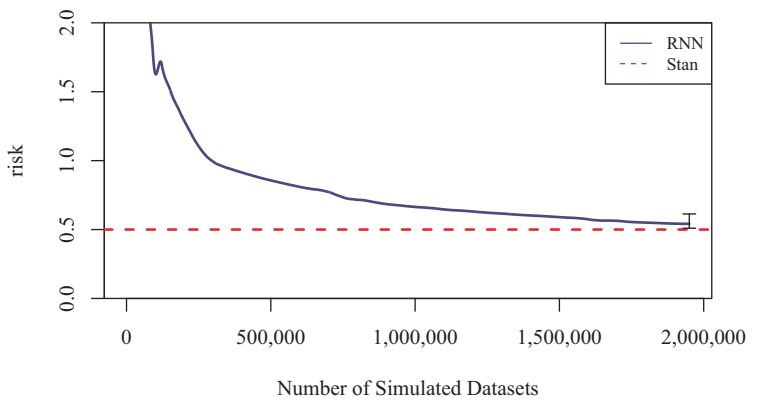

Figure 5. Risk curve for estimation of the posterior median of the basic reproduction rate in the stochastic SIR simulation setting. We compare the results with those of Stan, and both methods observe a grid of 101 time points. Our estimator achieves a risk that is about 3% greater than that of the Stan implementation.

simulated data sets, though this problem is slightly more difficult for us. We evaluated the performance on a held-out test set of 2,000 data sets.

To determine whether our method improves over Stan in this setting, we consider the same data-generating mechanism, but instead of observing $S(t)$ and $I(t)$ for all integer values of t, we observe only $t \in \{0, 20, 40, 60, 80, 100\}$. In the case of our Stan implementation, we can only use finite differences over the six observations, because we do not marginalize out the drift and the stochastic component over a finer, unobserved grid. Because our observed grid is sparse, this leads to a very poor approximation of the true data-generating mechanism. However, for our RNN, we can simulate over a grid of arbitrary granularity, but only train on the six observed time points. The results are shown in Figure 6. We see that, owing to the misspecification of Stan, our method has a lower risk after a small number of observed data sets. This is an example of how stochastic simulation techniques can fail when calculating the likelihood exactly is computationally intractable.

3.4. HMM example

In this section, we consider a hidden Markov model (HMM) example. In this setting, we observe a sequence of values, $y_1, \ldots, y_{100}$. These values are emissions from hidden states $x_1, \ldots, x_{100}$, which are not observed. There are three possible states at each x_i, $\{s_1, s_2, s_3\}$. We selected the following prior and likelihood for these simulations:

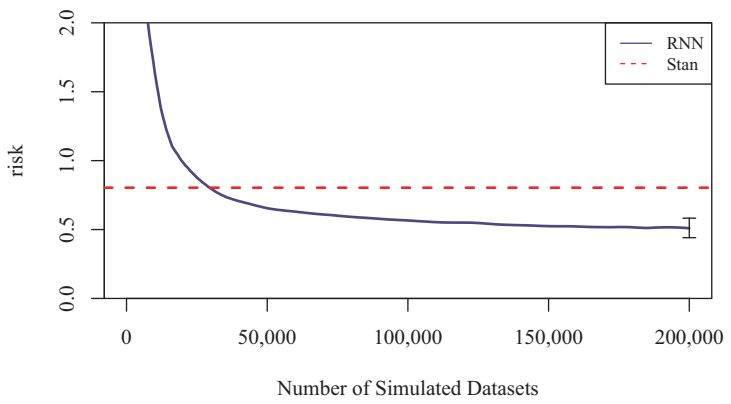

Figure 6. Risk curve for the estimation of the posterior median of the basic reproduction rate in the stochastic SIR simulation setting. We compare the results with those of Stan, and both methods observe only six time points. Our estimator had about a 44% reduction in risk over that of the Stan implementation.

$$\begin{aligned} \theta &\sim N(0,1), \\ Z_1, Z_2, Z_3 &\sim N(\theta, 1), \\ X_0 &\sim U(\{s_1, s_2, s_3\}), \\ y_i | X_i = s_j &\sim N(Z_j, 1), \\ P(X_{i+1} = s_j | X_i = s_k) &= \frac{e^{|z_j - z_k|}}{\sum_{j=1}^{3} e^{|z_j - z_k|}}, \end{aligned}$$

Therefore, the hidden states represent unobserved means from which observed Gaussian random variables are drawn. The transition probability between states is proportional to the exponentiated ℓ_1-distance between those states. Our goal is to conduct an inference on the posterior distribution of θ, the center of these unobserved means, $\theta | y_1, \ldots, y_{100}$. This model is a simplified version of the type of continuous-emission HMMs used in speech recognition (Ananthi and Dhanalakshmi (2015)).

We estimated 90% credible intervals on a held-out test set of 2,000 sequences. We compared the results with those of a standard importance sampling ABC with Gaussian kernel weights. Because Y is too high-dimensional to be computationally tractable, we instead use the deciles of the Y sequence as a summary statistic. We generated 1,000 importance samples per observed sequence, and estimated the 0.05 and 0.95 posterior quantiles from this sample with a bandwidth of one.

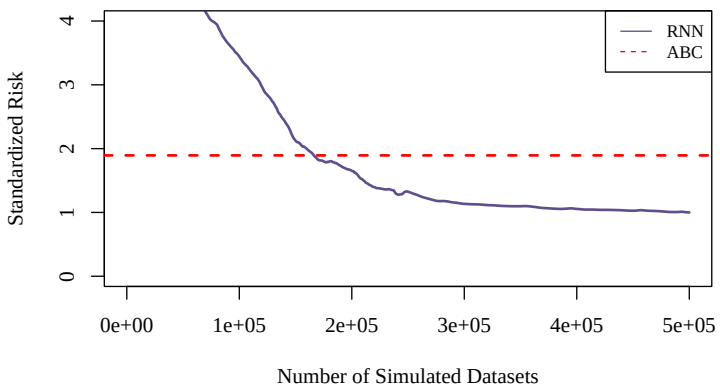

Figure 7. Standardized risk of the estimated 90% credible interval for the center of the unobserved hidden state means.

The results are shown in Figure 7. In this simulation scenario, it is difficult to choose sensible summary statistics. The recurrent network has a marginal coverage of 88.95% and an average interval width of 2.11. The ABC with summary statistics has a marginal coverage of 50.35% and an average interval width of 0.92. Recall that the prior quantiles have an interval width of 3.29 and marginal coverage of 90%. Thus, ABC with these summary statistics is more conservative than the prior, whereas the RNN leverages information to provide narrower intervals. The RNN is ideal in this scenario because of the difficulty of the problem and the sequential nature of the observed data.

3.5. Order-two moving average example

Finally, we consider an order-two moving-average model. Suppose we observe a sequence X, with

$$X_j \sim Z_j + \theta_1 Z_{j-1} + \theta_2 Z_{j-1}, \tag{3.1}$$

where Z_j are latent Gaussian random variables. We would like to obtain a 90% credible interval on θ_2. Note that if Z_j follow a non-Gaussian distribution, the likelihood $P(X \mid \theta)$ is intractable. However, here, we can draw comparisons with the exact posterior distribution.

The likelihood and prior are identical to those of (Jiang et al. (2018)). The Z_i are standard Gaussian, and θ_1 and θ_2 are uniform over a specific triangular region such that they are identifiable.

We compare our method with the semi-automatic ABC procedure described in (Fearnhead and Prangle (2011)), transforming the input data into a vector

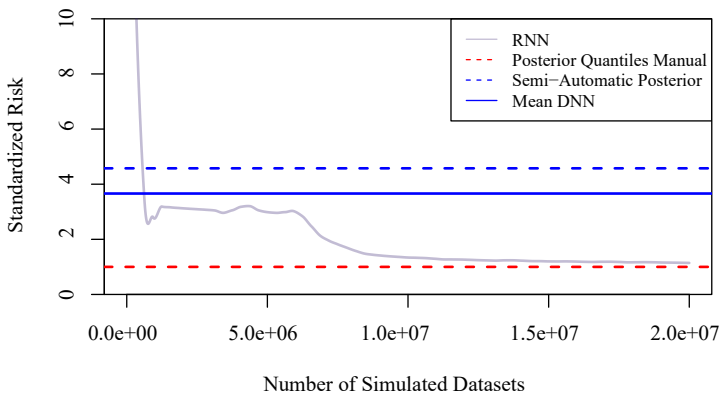

Figure 8. Standardized risk of the estimated 90% credible interval for the second-order coefficient in a moving average model.

of one-gap products, X_jX_{j+2}. We chose this transformation because the sample mean of X_jX_{j+2} is a consistent estimator of θ_2. Using a polynomial basis expansion of this transformed X, we then use a linear regression to estimate the posterior mean, and use that estimate as a summary statistic. Fearnhead and Prangle call the method "semi-auto ABC," because although the procedure can be automatic, the choice of the transformation in this problem is not. We also compare our results with those of a method proposed by (Jiang et al. (2018)), who use a deep neural network to estimate the posterior mean. Then, they use that estimate as a summary statistic for ABC. In both settings, we use rejection sampling to obtain the approximate posterior. The rejection scheme and the number of samples drawn are calibrated to have approximately the same runtime as our RNN method. The method were tested on a held-out set of 1,000 sequences.

The loss results can be seen in Figure 8. Because both comparison methods use an estimate of the posterior mean as a summary statistic, it is not surprising that the perform poorly in the tails. Our method directly targets the 90% credible interval, and has approximately correct coverage, while maintaining relatively narrow interval widths.

3.6. Summary of coverage results

For all of the simulation settings above, we estimated 90% posterior credible intervals. A comparison of our coverage results with those of various other estimators can be seen in Table 1. Our RNN estimators outperform or have comparable performance to STAN in every simulation setting. We maintain approximately

Table 1. Coverage of 90% credible intervals using various posterior estimators in several simulation settings. Estimators exhibiting poor marginal coverage are highlighted in red.

Gaussian			
Method	Coverage	Interval Width	Loss
RNN	0.906	0.241	0.0149
Exact Posterior	0.8958	0.233	0.0146
Prior Quantiles	0.903	0.323	0.0203
Mixture of Finite Mixtures			
Method	Coverage	Interval Width	Loss
RNN	0.942	0.339	0.019
Stan (BIC)	0.130	0.283	0.377
BayesianMixtures	0.921	0.147	0.009
Prior Quantiles	0.900	1.225	0.079
Stochastic SIR			
Method	Coverage	Interval Width	Loss
RNN	0.879	4.12	0.321
Stan	0.878	2.773	0.198
Prior Quantiles	0.900	10.17	0.935
Sparse Stochastic SIR			
Method	Coverage	Interval Width	Loss
RNN	0.9105	3.64	0.362
Stan	0.586	2.55	0.926
Prior Quantiles	0.900	10.17	0.935
Hidden Markov Model			
Method	Coverage	Interval Width	Loss
RNN	0.890	2.11	0.137
ABC	0.504	0.92	0.260
Prior Quantiles	0.900	3.29	0.203
Order Two Moving Average			
Method	Coverage	Interval Width	Loss
Exact Posterior	0.90	0.417	0.029
RNN	0.865	0.527	0.035
Semi-Auto ABC	0.440	0.476	0.135
Post-Mean DNN	0.466	0.400	0.108

correct marginal coverage, while having much narrower intervals than those of the prior.

4. Discussion

We have proposed a method for using deep RNNs to approximate posterior quantiles of a univariate parameter of interest from a possibly multivariate

Bayesian problem. To fit this neural net, it is only necessary to sample from the prior and likelihood. Posterior samples never need to be drawn, and, as long as we can sample from $P_{\theta,\eta}$, the likelihood itself never needs to be calculated. We have proposed three types of networks: a simple network for estimating a single, prespecificed, conditional quantile; a multi-task network for estimating a finite set of prespecified quantiles; and a slightly more complex network for estimating the entire conditional quantile function. We show that, in increasingly complex settings, an RNN can approximate single posterior quantiles approximately as accurately as specific state-of-the-art methods can that attempt to sample from the posterior directly.

To simplify the exposition, in Section 1, we assumed that the nuisance parameter, η, belongs to a finite-dimensional space. In fact, we do not use this assumption. We require only that one can sample parameters from the prior, and subsequently sample data from the corresponding likelihood. Therefore, in nonparametric Bayes problems, where an infinite-dimensional parameter can be sampled from a prior and the data can be sampled from the likelihood, the proposed method can be immediately used to estimate a univariate summary of this infinite-dimensional parameter.

Compared with other generalized tools for posterior approximation, such as Stan, our method is advantageous when the likelihood is not easily calculable. In the second stochastic SIR model, the results of which are shown in Figure 6, we achieve a lower loss than that of a natural implementation in Stan. We reach a lower loss because our Stan implementation must necessarily misspecify the likelihood in order to get posterior estimates.

It is interesting to compare the proposed method to the ABC methodology in terms of different forms of the smoothing methods used in classical nonparametric problems. Both the ABC methodology and our meta-learning procedure aim to approximate a quantile using many draws from the prior and likelihood. In the ABC methodology, the function mapping from the observed data to a quantile of the posterior distribution is approximated by a local smoother that uses only information from simulation replicates near the originally observed data. This is analogous to kernel smoothing methods in nonparametric regression settings, where a regression function estimate at a single predictor value x relies only on observations with predictors near x. For bounded kernels, a perturbation of distant observed predictor values has no impact on the regression fit at a distant point. On the other hand, our neural network approach uses all simulated data points to learn a rich neural network approximation to the function mapping from the observed data to a posterior quantile. This is analogous to series estimators

from nonparametric statistics, where linear models are fitted using a collection of basis functions applied to the observed predictors. Here, a perturbation of distant observed predictor values can have a nontrivial impact on the regression fit at a given point.

Our proposed method is also quite similar to other Bayesian deep learning methodologies, such as the exchangeble network proposed in Chan et al. (2018). However, our networks exhibit good performance in both exchangeable and non-exchangeable cases. One critical advantage of our recurrent networks is that we are able to obtain predictions for sequences or data sets of arbitrary length. This is not possible with an exchangeble network. However, as noted in Vaswani et al. (2017), the memory in a recurrent network is a bottleneck in a number of contexts. Therefore, as the dependence grows between data points that are far apart in our sequences, the size of the memory in our recurrent nodes must grow as well. However, the fixed memory size of the recurrent nodes could also be beneficial in an online learning setting. As new data are acquired, our RNN estimator quickly calculates new posterior estimates, without the need for retraining. Further investigation of potential applications to online inference should be conducted.

As with other posterior sampling and approximation methodologies, we anticipate that our method will suffer from the curse of dimensionality. For MCMC methods, the curse of dimensionality typically occurs when the parameters are high dimensional. In contrast, we do not expect that having a high-dimensional nuisance parameter η will negatively impact the performance of our method, because we only estimate the posterior of a univariate parameter θ. However, our method is susceptible to the curse of dimensionality in the data structure, because it implicitly smooths across possible data realizations. One approach to improve the performance of our method in the presence of high-dimensional data would be to increase the richness of our network. Because we can simulate an unlimited number of prior-likelihood draws for our stochastic gradient steps, using a very rich network will not lead to overfitting, though the optimization scheme may require more draws to converge. Therefore, when the data are high dimensional, we expect that our method will benefit greatly from recent progress in neural network software that enables users to leverage massive computational power to quickly optimize deep networks.

Acknowledgments

Noah Simon and Marco Carone were supported by NIH grant R01HL137808.

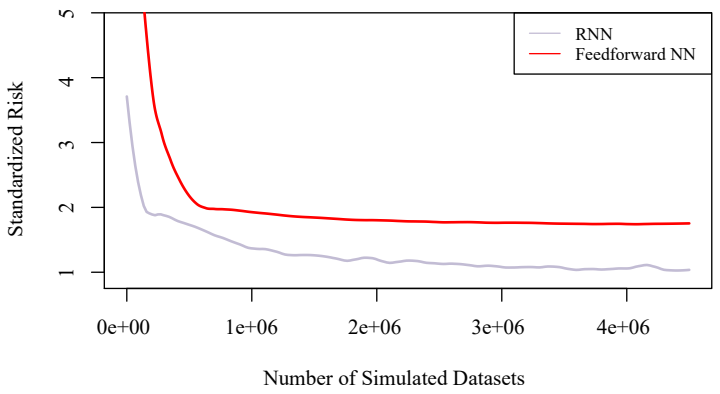

Figure 9. Standardized risk of a feedforward neural network vs. a recurrent neural network. Targets of estimation are the 0.05 and 0.95 posterior quantiles of the maximum component in a mixture of finite mixtures with a prior on the number of components. Both networks have 3 hidden layers and 10 nodes per hidden layer.

Appendix

A.1. Feedforward comparison

As an illustrative example of how an RNN outperforms a Feedforward network in posterior quantile estimation, we performed simulations which make a direct comparison. In Figure 9, we show that a Feedforward network with an identical internal structure (same number of hidden nodes per layer and number of layers) is outperformed by a recurrent counterpart in estimating a 90% credible interval for the maximum component of a mixture of finite mixtures. Furthermore, the majority of the cases where our method is best-suited involve series data where the likelihood is not easily calculable, making the RNN an even more preferable option.

A.2. Rate of convergence

According to the Bernstein-von Mises theorem, any posterior quantile should concentrate around the true parameter at a rate of $O(1/\sqrt{n})$. For the Gaussian conjugate prior simulation setting, we trained several networks with varying sample sizes to predict the 0.05 quantile. Averaging the distance from these estimates to the true values over a large test set give us the approximate convergence rate. In Figure 10, we see a log-log plot of the 0.05 quantile distance from the true value vs. sample size. The slope of the imposed line of best fit is -0.48. Therefore, we see that the 0.05 quantile concentrates around the true parameter at approx-

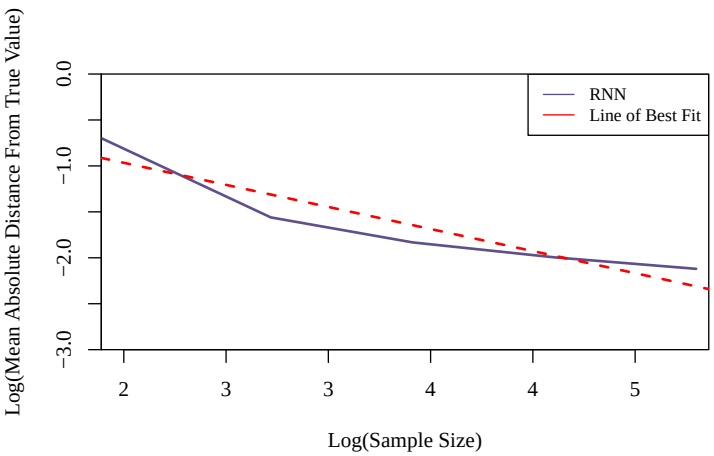

Figure 10. Log(Mean absolute difference) between estimated 0.05 quantile and true value. The line of best fit is superimposed.

imately a rate of $O(1/\sqrt{n})$, though we admit this relationship is not perfectly linear. While ideally we would also show this convergence in more complicated simulation settings, doing so requires training a separate neural network for each value of n, which is computationally expensive for more complex examples. Note also that as n increases, the optimal architecture and training regime changes. However, all of the networks use to produce this convergence plot were identical in their architecture and training regimen. This is one reason why the convergence rate is not exact.

A.3. Distribution function analysis

In the continuous estimation setting, where the quantile to be estimated is an imput to the RNN, it is possible to obtain an estimate for the entire distribution function by inputting a grid of quantiles. In the Gaussian conjugate prior scenario, it is also possible to compare this estimated distribution function with the true posterior distribution, since it has closed-form. One example of this can be seen in Figure 11. While the distribution function is easier to estimate than the density function, one could imagine binning predicted quantile values to achieve a density estimate.

A.4. Conditional coverage analysis

Risk is not an ideal measure of performance, but it is the best option in some cases. Other measures worth considering, such as marginal coverage and interval width, are reported in Table 1. However, conditional coverage is still of

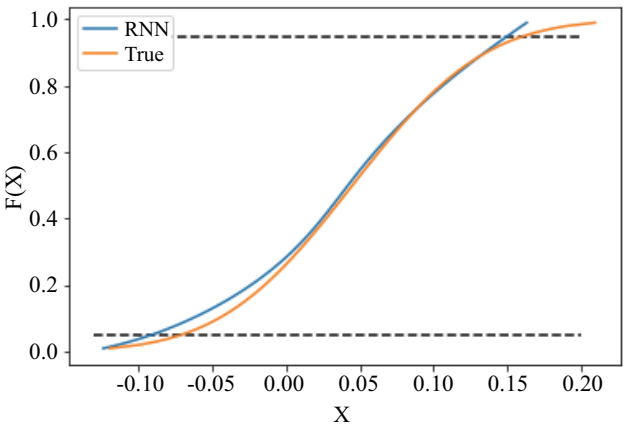

Figure 11. True posterior distribution functions vs. RNN estimated distribution function. Horizontal lines represent the 0.05 and 0.95 quantiles.

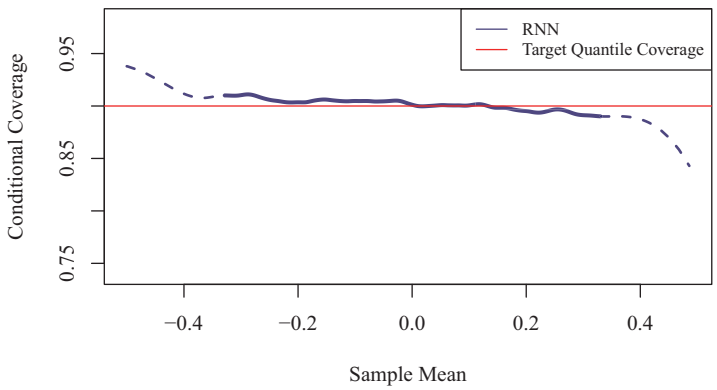

Figure 12. Conditional coverage of RNN posterior quantile estimates, stratified by sample mean. Dotted lines represent values in the lowest or highest 1% of our test set.

interest. In the Gaussian conjugate prior setting, where we have direct access to the posterior distribution, we can stratify by the sufficient statistic, the sample mean, to obtain a measure of conditional coverage, seen in Figure 12. We see that, apart from extreme values of the sufficient statistic, we obtain approximately correct conditional coverage as well.

A.5. Permutation invariance

In the first two simulation settings presented, the posterior distribution of our parameter of interest is invariant to permutations of the input data. In the

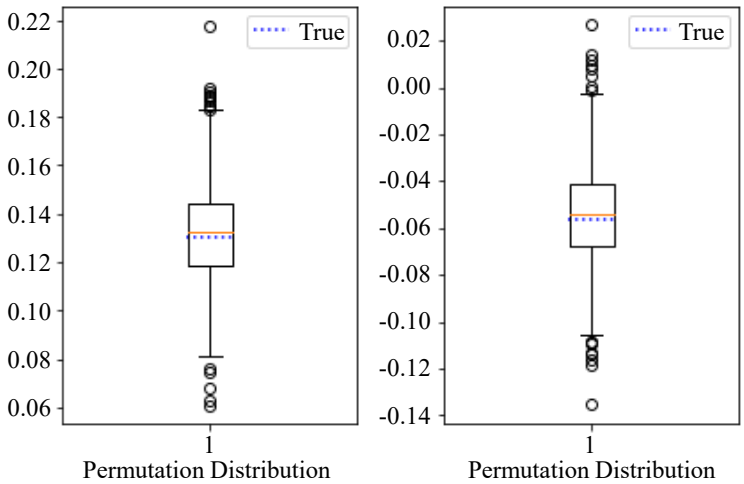

Figure 13. Boxplot of predicted values under permutation with a superimposed true posterior value for two example datasets.

first of these examples, the Gaussian conjugate prior scenario, we analyzed the performance of our model by averaging predictions from permutations of the input data. Two examples of this can be seen in Figure 13.

Another potential solution to exchangeability is sorting the input data. When we tested coverage in Table 1, we sorted the input data for the Mixture of Finite Mixture example, which improved performance. In this specific setting, because we are interested in the posterior distribution of the maximum component, sorting is an intuitive solution.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J. et al. (2016). TensorFlow: A system for large-scale machine learning. In *OSDI*, 265–283.

Allen, L. J. (2017). A primer on stochastic epidemic models: Formulation, numerical simulation, and analysis. *Infectious Disease Modelling* **2**, 128–142.

Ananthi, S. and Dhanalakshmi, P. (2015). SVM and HMM modeling techniques for speech recognition using LPCC and MFCC features. 519–526.

Asmussen, S. and Glynn, P. W. (2007). *Stochastic Simulation: Algorithms and Analysis*. Springer, New York.

Bach, F. (2017). Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research* **18**, 1–53.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M. et al. (2016). Stan: A probabilistic programming language. *Journal of Statistical Software* **20**, 1–37.

Chan, J., Perrone, V., Spence, J. P., Jenkins, P. A., Mathieson, S. and Song, Y. S. (2018). A likelihood-free inference framework for population genetic data using exchangeable neural networks.

Clancy, D. and O'Neill, P. D. (2008). Bayesian estimation of the basic reproduction number in stochastic epidemic models. *Bayesian Analysis* **3**, 737–757.

Creel, M. (2017). Neural nets for indirect inference. *Econometrics and Statistics* **2**, 36–49.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* **2**, 303–314.

Fearnhead, P. and Prangle, D. (2011). Constructing summary statistics for approximate Bayesian computation: Semi-automatic ABC. *arXiv:1004.1112*.

Ghahramani, Z. (2001). An introduction to hidden Markov models and Bayesian networks. In *Hidden Markov Models: Applications in Computer Vision* (Edited by H. Bunke and T. Caelli), 9–41. World Scientific, Singapore.

Hochreiter, S., Younger, A. S. and Conwell, P. R. (2001). Learning to learn using gradient descent. In *Artificial Neural Networks - ICANN 2001* (Edited by G. Dorffner, H. Bischof and K. Hornik), 87–94. Springer, Berlin, Heidelberg.

Jiang, B., Wu, T.-Y., Zheng, C. and Wong, W. H. (2018). Learning summary statistic for approximate Bayesian computation via deep neural network. *Statistica Sinica* **27**, 1595–1618.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv:1412.6980*.

Koenker, R. and Hallock, K. F. (2001). Quantile regression. *Journal of Economic Perspectives* **15**, 143–156.

Marjoram, P., Molitor, J., Plagnol, V. and Tavaré, S. (2003). Markov chain Monte Carlo without likelihoods. In *Proceedings of the National Academy of Sciences* **100**, 15324–15328.

Miller, J. W. and Harrison, M. T. (2018). Mixture models with a prior on the number of components. *Journal of the American Statistical Association* **113**, 340–356.

Nedic, A. and Bertsekas, D. P. (2001). Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization* **12**, 109–138.

Prangle, D. (2015). Summary statistics in approximate Bayesian computation. *arXiv:1512.05633*.

Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *arXiv:1706.05098*.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1988). Learning internal representations by error propagation. In *Readings in Cognitive Science* (Edited by A. Collins and E. E. Smith), 399-421. Morgan Kaufmann, Burlington.

Smith, L. N. and Topin, N. (2017). Super-convergence: Very fast training of neural networks using large learning rates. *arXiv:1708.07120*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N. et al. (2017). Attention is all you need. In *Proceedings of Advances in Neural Information Processing Systems* **30**. Curran Associates, Inc., New York.

Thayer Fisher

Department of Biostatistics, University of Washington, Seattle, WA 98195-0005, USA.

E-mail: thayerf@uw.edu

Alex Luedtke
Department of Biostatistics, University of Washington, Seattle, WA 98195-0005, USA.
E-mail: aluedtke@uw.edu
Marco Carone
Department of Biostatistics, University of Washington, Seattle, WA 98195-0005, USA.
E-mail: mcarone@uw.edu
Noah Simon
Department of Biostatistics, University of Washington, Seattle, WA 98195-0005, USA.
E-mail: nrsimon@uw.edu

(Received September 2020; accepted February 2022)