6th ICML Workshop on Automated Machine Learning (2019)

Improving Automated Variational Inference with Normalizing Flows

Stefan Webb INFO@STEFANWEBB.ME
University of Oxford, UK

Jonathan P. Chen, Martin Jankowiak {JPCHEN,JANKOWIAK}@UBER.COM
Uber AI Labs

Noah Goodman NDG@UBER.COM
Stanford University, Uber AI Labs

Abstract

We describe a framework for performing automatic Bayesian inference in probabilistic programs with fixed structure. Our framework takes a probabilistic program with fixed structure as input and outputs a learnt variational distribution approximating the posterior. For this purpose, we exploit recent advances in representing distributions with neural networks. We implement our approach in the Pyro probabilistic programming language, and validate it on a diverse collection of Bayesian regression models translated from Stan, showing improved inference and predictive performance relative to the existing state-of-the-art in automated inference for this class of models.

1. Introduction

Bayesian modeling is a powerful tool for scientific discovery and data exploration. In practice, however, the Bayesian workflow of modeling, inference, and criticism is often impeded by conceptual and algorithmic complexity and lack of automation. In particular, constructing effective inference algorithms often requires insight from hard-won intuitions or lengthy model-specific derivations. This slows the speed with which the model space can be searched and thus the pace of scientific research. Probabilistic programming proposes to streamline the Bayesian workflow using the power of programming languages to express models and to automate the process of inference. In this work we explore automated variational inference for graphical models using flexible variational families defined by deep neural networks.

An ideal automated inference framework would be: general (not depending on any idiosyncratic aspect of the model or data), scalable (in both the size of the dataset and the number of latent variables), and automatic (requiring little intervention from the user). Satisfying all these criteria simultaneously is a Herculean task, and automated Bayesian inference remains an open research challenge. In contrast to other work looking to automate inference in universal probabilistic programming languages with stochastic recursion/branching and first-order functions (Tolpin et al., 2015; Le et al., 2016), we restrict our attention to graphical models with fixed structure, similarly to Lunn et al. (2012) and Carpenter et al. (2017). This useful class of models includes Bayesian regression and mixed-membership models such as latent Dirichlet allocation, to name a few, and comprises the vast majority of Bayesian models used in industry and science.

Modern variational inference (Kingma and Welling, 2014; Rezende et al., 2014) is an excellent candidate for automated inference. It is applicable to a wide class of models, and is known to often converge more quickly than MCMC in practice, especially for large datasets. It has other benefits too: we get model learning for free, and can incorporate advances in deep learning, both for optimization and the design of the variational approximation. And yet, the state-of-the-art in automated variational inference for Bayesian models with fixed structure—*automatic differentiation variational inference* (ADVI) (Kucukelbir et al., 2017)—suffers from a lack of adoption in applied work, often failing to converge to an adequate approximation to the posterior distribution. We posit that a primary reason for this behavior is that the class of variational distributions used is too restricted. Consequently we investigate improving upon ADVI by increasing the representional power of the variational distribution using a recently developed class of methods known as normalizing flows (Tabak and Turner, 2013; Rezende and Mohamed, 2015). Empirically we show that this results in more effective automated variational inference, leading to tighter variational bounds and improved predictive performance.

2. Background

2.1 Modern variational inference

One of the principal challenges of Bayesian data analysis is performing efficient approximate inference. *Variational inference* (VI) (Blei et al., 2016) is a family of approximate inference methods that casts inference as an optimization problem. VI has enjoyed great popularity, since it naturally supports data subsampling while still yielding reasonable posterior approximations for a restricted class of models.

Consider a generative model with joint density $p(\mathcal{D}, \mathbf{z})$ for latent variables $\mathbf{z}$ and observed data $\mathcal{D}$. VI methods assume a family of inference artifacts, $\mathcal{Q} = \{q_\psi(\mathbf{z} \mid \mathcal{D}) \mid \psi \in \Psi\}$—known variously as inference networks, probabilistic encoders, variational distributions, and guide programs—and attempt to find the member $q_{\psi*}$ closest to the true posterior $p(\mathbf{z} \mid \mathcal{D})$ relative to some divergence measure, commonly the reverse KL-divergence:

$$\psi^* = \text{argmin}_{\psi \in \Psi} \text{KL}\left\{q_\psi(\mathbf{z} \mid \mathcal{D}) || p(\mathbf{z} \mid \mathcal{D})\right\} \tag{1}$$

It is typically not possible to directly evaluate the KL divergence in (1), as it involves the intractable model evidence $p(\mathcal{D})$. However, it can be shown that minimizing the reverse KL-divergence is equivalent to maximizing the *evidence lower bound* (ELBO):

$$\begin{aligned} \psi^* &= \text{argmax}_{\psi \in \Psi} \mathcal{L}_{\text{ELBO}} \\ &= \text{argmax}_{\psi \in \Psi} \mathbb{E}_{q_\psi(\cdot|\mathcal{D})} \left[\log p(\mathcal{D}, \mathbf{z}) - \log q_\psi(\mathbf{z} \mid \mathcal{D})\right]. \end{aligned} \tag{2}$$

In classical VI (Wainwright and Jordan, 2008), (2) is typically solved by deriving closed-form updates and performing coordinate-wise ascent on ψ. This is only possible in a restricted subset of models where a certain type of conjugacy holds. Modern VI, on the other hand, leverages stochastic gradient methods to solve the optimization problem in (2), making it applicable to a much wider class of models.

The solution $q_{\psi*}$ to (2) depends on the data $\mathcal{D}$. For some models—for example those with local latent variables—it can be advantageous to parameterize $q_\psi(\cdot|\mathcal{D})$ with an explicit

dependence on $\mathcal{D}$; this is known as *amortized inference* (Gershman and Goodman, 2014). In this work, however, we consider the non-amortized scenario, and denote the variational approximation by $q_\psi(\mathbf{z})$.

Once we have solved the optimization problem in (2), samples from $q_{\psi*}$ can directly be used as approximate posterior samples; moreover, $q_{\psi*}$ can also be used as the proposal for a particle-based inference method (Paige and Wood, 2016; Le et al., 2016). Alternatively, one can simply report summary statistics obtained from the variational distribution, such as the means and quantiles of the marginals of $q_{\psi*}$.

2.2 Automatic differentiation variational inference

The current state-of-the-art in automated (non-amortized) VI for probabilistic programs with fixed structure is known as *automatic differentiation variational inference* (ADVI) (Kucukelbir et al., 2017), an implementation of which exists in the Stan probabilistic programming language (Carpenter et al., 2017). It is applicable to models with continuous latent variables.

There are two variants of ADVI. In mean-field ADVI (mf-ADVI), the approximating distribution is factorized as $q_\psi(\mathbf{z}) = \prod_{n=1}^{N} q_{\psi_n}(z_n)$, where each $q_{\psi_n}(z_n)$ is a univariate Normal distribution $\mathcal{N}(\mu_n, \sigma_n)$. Critically, mf-ADVI does not capture correlations among the latent variables. To address this, the fully connected variant of ADVI (fc-ADVI) uses a multivariate Normal distribution with a dense covariance matrix, i.e. $q_\psi(\mathbf{z}) = \mathcal{N}(\mu, \Sigma)$ with $\psi = (\mu, \Sigma)$.

In the above we have implicitly assumed that the support of $\mathbf{z}$ is $\mathbb{R}^N$. When this is not the case, ADVI employs an invertible transformation f to transform samples $\mathbf{z}' \sim q_\psi(\cdot)$ via $\mathbf{z} = f(\mathbf{z}')$, where f is chosen so that the support of each z_n is the same as that specified by the model. For instance, if $\text{supp}(z_n) = \mathbb{R}^+$ in the model, then one could choose $z_n = f_n(z'_n) = \exp(z'_n)$. In this case the log density of $q_\psi(\mathbf{z})$ is computed as

$$\log q_\psi(\mathbf{z}) = \log \mathcal{N}(\mathbf{z}'|\mu, \text{diag}(\sigma)) - \sum_{n=1}^{N} \log \left| \frac{\partial f_n}{\partial z'_n} \right|$$

3. Method

3.1 Extending ADVI with normalizing flows

One of the shortcomings of ADVI is that the transformation f is fixed. As noted by the authors, the solution obtained from ADVI often depends on the specific choice for f, which is not unique. For example, $f_n(z'_n) = \text{softplus}(z'_n)$ would be another valid choice when $\text{supp}(z_n) = \mathbb{R}^+$. Moreover, the ADVI variational approximation q_ψ can represent only a narrow class of distributions, with each marginal being the fixed transformation of a Gaussian distribution.

We propose improving on both of these aspects of ADVI by augmenting f with a learnable invertible transformation, known in the literature as a *normalizing flow* (NF) (Tabak and Turner, 2013; Rezende and Mohamed, 2015). Specifically, we employ a *neural autoregressive flow* (NAF) (Huang et al., 2018), which is provably a universal approximator for continuous

densities, and has been shown to easily recover multimodal distributions.[1] We refer to the resulting framework as *normalizing flow automatic variational inference* (NF-AVI).

We apply the fixed ADVI transformation f_n to the output of a single layer of NAF to ensure that z_n has the same support as the model, resulting in the element-wise transformation,

$$z_n = f_n(g_n(z'_n)) = f_n\left(s^{-1}\left(\mathbf{w}_n^T s\left(\mathbf{a}_n z'_n + \mathbf{b}_n\right)\right)\right) \qquad \text{with} \qquad s(x) = (1 + e^{-x})^{-1}$$

where $\mathbf{w}_n, \mathbf{a}_n, \mathbf{b}_n \in \mathbb{R}^D$, $0 < w_{n,i} < 1$, $\sum_i w_{n,i} = 1$, $a_{n,i} > 0$, and D denotes the number of hidden units. The pseudo-parameters $\mathbf{w}_n, \mathbf{a}_n, \mathbf{b}_n$ are the outputs of a neural network that only depends on $z'_{\prec n}$, and all such parameters can be calculated in parallel with an autoregressive neural network such as MADE (Germain et al., 2015), which acts like a hypernetwork (Ha et al., 2016).

Typically, the underlying base distribution that is input to a NF is chosen to be a unit Normal distribution, i.e. $\mathbf{z}' \sim \mathcal{N}(0, 1)$. However, we find that is beneficial to use a base distribution given by $\mathbf{z}' \sim \mathcal{N}(\mu, \text{diag}(\sigma))$, where μ and σ are learned. In this way, our method can be seen as learning how the variational distribution deviates from the mean-field ansatz.

NAF is a type of autoregressive flow, i.e. the transformed z_n is only a function of $z'_{\prec n} = (z'_1, z'_2, \ldots, z'_n)$ and the learnable parameters. For autoregressive flows, the Jacobian is triangular, and hence the log det Jacobian needed to compute the transformed density can be easily calculated:

$$\log q_\psi(\mathbf{z}) = \log \mathcal{N}(\mathbf{z}'|\mu, \text{diag}(\sigma)) - \sum_{n=1}^{N} \log \left|\frac{\partial f_n}{\partial g_n}\right| - \sum_{n=1}^{N} \log \left|\frac{\partial g_n}{\partial z'_n}\right|$$

To be clear, the learnable parameters for our autoregressive NF variational approximation are $\psi = (\mu, \sigma, \theta)$, where θ are the parameters of the autoregressive neural network underlying the flow.

3.2 Pyro and automated inference

Early probabilistic programming languages (PPLs) generally necessitated automated inference algorithms to be built into the system. In contrast, more recent languages permit automated inference algorithms to be developed in the language itself, using the abstractions provided by the language. We developed our automated inference algorithms in the Pyro (Bingham et al., 2018) PPL, which is built upon PyTorch and thus benefits from GPU accelerated math and a mature neural network toolkit.

The chief abstraction in Pyro is the *effectful state-handler* (Kammar et al., 2013; Plotkin and Pretnar, 2009), which we use to implement automated inference. The model is executed once, using an effectful state handler to record the names, dimensions, and supports of the latent variables. From this information, the dimensionality of the auxiliary $\mathbf{z}'$ for ADVI and NF-AVI can be established, plus the necessary transformation f that is needed to ensure $f(\mathbf{z}')$ has the correct support. In this way, a probabilistic program for the variational approximation is automatically generated from the model.

1. Multiple NF transformations can be composed to produce richer flows. Although in the case of NAF, the transformation is sufficiently complex that a single layer often suffices. We use a single NAF in all our experiments.

	mf-ADVI	fc-ADVI	NF-AVI
(1)	**1150.0 (3.340 × 10^{-1})**	1179.0 (4.765)	1153.0 (1.074)
(1) chr	1149.0 (2.082 × 10^{-1})	1149.0 (2.021 × 10^{-1})	**1148.0 (2.321 × 10^{-1})**
(2)	−416.1 (7.630 × 10^{-1})	−416.2 (6.078 × 10^{-1})	**−418.6 (2.331 × 10^{-1})**
(3)	611.7 (2.669)	2384.0 (3.942 × 10^{1})	**562.0 (1.753 × 10^{1})**
(4)	5965.0 (4.794 × 10^{2})	5834.0 (5.674 × 10^{2})	**1886.0 (8.503 × 10^{1})**
(5)	1834.0 (2.399 × 10^{2})	3633.0 (1.108 × 10^{2})	**1518.0 (9.517)**
(5) chr	1539.0 (3.854 × 10^{1})	4064.0 (2.941 × 10^{2})	**1436.0 (9.098 × 10^{-1})**
(6)	4926.0 (2.465 × 10^{2})	5004.0 (1.722 × 10^{2})	**1536.0 (1.752 × 10^{1})**
(6) chr	6406.0 (8.503 × 10^{2})	14830.0 (2.023 × 10^{3})	**1460.0 (8.528)**
(7)	**1398.0 (1.234 × 10^{1})**	**1394.0 (1.200 × 10^{1})**	1421.0 (1.268 × 10^{1})
(8)	1645.0 (9.216)	1562.0 (1.608 × 10^{1})	**1477.0 (5.332)**
(9)	618.0 (5.682)	612.5 (2.386)	**588.4 (5.815 × 10^{-1})**
(10)	1160.0 (7.877 × 10^{-1})	1159.0 (7.641 × 10^{-1})	**1157.0 (2.366 × 10^{-2})**
(11)	1089.0 (7.986 × 10^{-1})	1116.0 (1.159)	**1083.0 (1.156 × 10^{-1})**
(12)	1996.0 (6.268 × 10^{-1})	1996.0 (5.764 × 10^{-1})	**1969.0 (4.155 × 10^{-2})**
(13)	2222.0 (5.341)	2233.0 (5.952)	**2073.0 (8.818)**

Table 1: An estimate of the negative ELBO at 1000 epochs (lower is better), averaging over 10 runs and indicating 1 standard error. The best mean for each model is highlighted in bold. Entries that do not statistically significantly differ from the best entry at the 1 standard error level are also given in bold. All numbers are to four significant figures. "chr" abbreviates the Choo–Hoffman reparameterization.

4. Experiments

To evaluate our proposed automated VI framework, we run our method on 16 Bayesian regression models taken from Gelman and Hill (2006) that are part of the Stan model examples repository,[2] translated to Pyro (Chen et al., 2018). They were chosen to represent a diversity of datasets and model structures (single-level and multi-level regression, standard and logistic regression, standard parameterization and Choo-Hoffman parameterization, etc.).[3]

We compare the two variants of ADVI to NF-AVI, the details of which are given in the appendix. We also tried a variational approximation based on inverse autoregressive flows (IAF) (Kingma et al., 2016) but found them to frequently suffer optimization issues, and so omit these results. We report the best ELBO per model-guide pair at 1000 epochs in Table 1, and the mean-squared-error (MSE) in Table 2. Additionally, we report the MSE from samples obtained by the No U-Turn Sampler (NUTS; Hoffman and Gelman (2014)), a state-of-the-art MCMC method that automatically tunes the parameters of Hamiltonian Monte Carlo, a popular inference algorithm for Bayesian regression models.

We observe that NF-AVI obtains the lowest negative ELBO for nearly all models, and when it does not the differences are minimal. For several models such as (4) and (6) the gap between NF-AVI and ADVI is large. Also, we see that the lower ELBO translates into better predictive performance relative to ADVI across nearly all models. NV-AVI is competitive with NUTS, obtaining a lower MSE for several models. The worse predictive performance of fc-ADVI relative to mf-ADVI for most models was unexpected and warrants further investigation.

2. `https://github.com/stan-dev/example-models/`
3. Code to reproduce the experiments is provided at `https://github.com/stefanwebb/autoguides`.

	mf-ADVI	fc-ADVI	NUTS	NF-AVI
(1)	$1.275\ (1.986 \times 10^{-3})$	$1.343\ (2.398 \times 10^{-2})$	$1.275\ (1.607 \times 10^{-4})$	$\mathbf{1.27\ (5.371 \times 10^{-3})}$
(1) chr	$1.278\ (6.888 \times 10^{-4})$	$1.281\ (7.109 \times 10^{-4})$	$\mathbf{1.276\ (2.929 \times 10^{-4})}$	$1.287\ (5.344 \times 10^{-3})$
(2)	$0.00924\ (3.469 \times 10^{-5})$	$0.009363\ (3.523 \times 10^{-5})$	$\mathbf{0.009103\ (6.783 \times 10^{-6})}$	$0.009149\ (2.898 \times 10^{-5})$
(3)	$0.2222\ (3.739 \times 10^{-3})$	$0.22\ (2.201 \times 10^{-3})$	$\mathbf{0.2194\ (1.607 \times 10^{-4})}$	$0.2239\ (1.031 \times 10^{-3})$
(4)	$165.7\ (1.030 \times 10^{1})$	140.1 (9.181)	$\mathbf{1.554\ (6.439 \times 10^{-4})}$	$3.713\ (9.689 \times 10^{-1})$
(5)	$2.31\ (3.516 \times 10^{-1})$	$156.7\ (3.537 \times 10^{1})$	$\mathbf{1.531\ (1.064 \times 10^{-3})}$	$1.663\ (1.590 \times 10^{-2})$
(5) chr	$1.964\ (3.027 \times 10^{-1})$	$1290.0\ (6.145 \times 10^{2})$	$\mathbf{1.525\ (3.043 \times 10^{-4})}$	$1.546\ (5.418 \times 10^{-3})$
(6)	158.0 (5.368)	$214.2\ (1.097 \times 10^{1})$	$\mathbf{1.637\ (4.502 \times 10^{-4})}$	$2.041\ (6.273 \times 10^{-2})$
(6) chr	79.44 (8.998)	$259.2\ (3.071 \times 10^{1})$	$\mathbf{1.644\ (7.725 \times 10^{-4})}$	$1.737\ (2.324 \times 10^{-2})$
(7)	$\mathbf{0.4505\ (2.835 \times 10^{-4})}$	$\mathbf{0.4503\ (2.854 \times 10^{-4})}$	$0.4582\ (8.933 \times 10^{-5})$	$0.4524\ (7.554 \times 10^{-4})$
(8)	$0.4414\ (5.216 \times 10^{-5})$	$\mathbf{0.4408\ (1.812 \times 10^{-4})}$	$0.4537\ (1.610 \times 10^{-4})$	$0.4454\ (4.109 \times 10^{-4})$
(9)	$1.314\ (1.430 \times 10^{-2})$	$1.308\ (4.336 \times 10^{-3})$	$1.337\ (1.036 \times 10^{-3})$	$\mathbf{1.138\ (3.022 \times 10^{-3})}$
(10)	$1.374\ (3.657 \times 10^{-3})$	$1.373\ (3.640 \times 10^{-3})$	$\mathbf{1.355\ (5.205 \times 10^{-4})}$	$1.356\ (2.408 \times 10^{-3})$
(11)	$1.157\ (2.289 \times 10^{-3})$	$1.19\ (3.751 \times 10^{-3})$	$1.153\ (8.104 \times 10^{-4})$	$\mathbf{1.145\ (2.053 \times 10^{-3})}$
(12)	$0.4578\ (7.537 \times 10^{-5})$	$0.4578\ (7.853 \times 10^{-5})$	$\mathbf{0.4538\ (2.893 \times 10^{-5})}$	$0.458\ (9.012 \times 10^{-5})$
(13)	$0.4822\ (4.713 \times 10^{-4})$	$0.4824\ (4.619 \times 10^{-4})$	$\mathbf{0.4818\ (4.111 \times 10^{-5})}$	$0.4819\ (7.233 \times 10^{-4})$

Table 2: An estimate of the MSE across models at 1000 epochs (lower is better).

5. Discussion

We have demonstrated on a diverse collection of Bayesian regression models that variational inference can be made more reliable by increasing the representational capacity of the variational approximation with normalizing flows. Surprisingly, we found the predictive performance using samples directly drawn from the variational approximation to be competitive with state-of-the-art MCMC methods, at least for the computational budget and set of models used in our experiments.

A future version of this work will investigate the cause of the poor relative performance of ADVI, and compare learning across a greater number of regression models. We will also extend the framework to models with discrete variables and large datasets that necessitate data subsampling.

The choice of optimization hyperparameters crucially effects the performance of learning, and yet is chosen manually in our current setup. We intend to investigate more robust optimization methods in future work, e.g. trust-region methods as in (Regier et al., 2017).

In Webb et al. (2018), minimally faithful graphical model structures were found to result in improved VI relative to taking a mean-field or fully-connected approach. Currently, the variational approximations we use have a fully-connected structure, and thus ignore significant structural information in the model. We will explore designing the structure of the variational approximation using a novel extension of the NaMI algorithm (Webb et al., 2018) for producing compact minimally faithful inverses for *plated* graphical models. Also, using an inversion algorithm would allow us to introduce amortization into the guide program in a sensible way, which is important when there are latent variables local to each data point.

Acknowledgments

We would like to thank Neeraj Pradhan and Eli Bingham for helpful discussions. SDW acknowledges funding from Uber AI Labs to complete this work during an internship.

References

Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 2018.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *arXiv preprint arXiv:1601.00670v2 [stat.CO]*, 2016.

Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76(1), 2017.

Jonathan P. Chen, Rohit Singh, Eli Bingham, and Noah Goodman. Transpiling stan models to pyro. In *The International Conference on Probabilistic Programming*, 2018.

Andrew Gelman and Jennifer Hill. *Data analysis using regression and multilevel/hierarchical models*. Cambridge university press, 2006.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: masked autoencoder for distribution estimation. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 881–889, 2015.

Samuel J Gershman and Noah D Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the 36th Annual Conference of the Cognitive Science Society*, 2014.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *The Journal of Machine Learning Research*, 15(1): 1593–1623, 2014.

Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. *arXiv preprint arXiv:1804.00779*, 2018.

Ohad Kammar, Sam Lindley, and Nicolas Oury. Handlers in action. In *ACM SIGPLAN Notices*, volume 48, pages 145–158. ACM, 2013.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Diederik P Kingma, Tim Salimans, and Max Welling. Improving variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4736–4744, 2016.

Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. Automatic differentiation variational inference. *The Journal of Machine Learning Research*, 18(1):430–474, 2017.

Tuan Anh Le, Atilim Gunes Baydin, and Frank Wood. Inference Compilation and Universal Probabilistic Programming. *AISTATS 2017*, 2016. URL `http://arxiv.org/abs/1610.09900`.

David Lunn, Chris Jackson, Nicky Best, David Spiegelhalter, and Andrew Thomas. *The BUGS book: A practical introduction to Bayesian analysis*. Chapman and Hall/CRC, 2012.

Brooks Paige and Frank Wood. Inference networks for sequential monte carlo in graphical models. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, 2016.

Gordon Plotkin and Matija Pretnar. Handlers of algebraic effects. In *European Symposium on Programming*, pages 80–94. Springer, 2009.

Jeffrey Regier, Michael I Jordan, and Jon McAuliffe. Fast black-box variational inference through stochastic trust-region optimization. In *Advances in Neural Information Processing Systems*, pages 2402–2411, 2017.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1530–1538, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of The 31st International Conference on Machine Learning*, 2014.

EG Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

David Tolpin, Jan-Willem van de Meent, and Frank Wood. Probabilistic programming in anglican. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 308–311. Springer, 2015.

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

Stefan Webb, Adam Golinski, Rob Zinkov, Siddharth Narayanaswamy, Tom Rainforth, Yee Whye Teh, and Frank Wood. Faithful inversion of generative models for effective amortized inference. In *Advances in Neural Information Processing Systems*, pages 3070–3080, 2018.

Appendix A. Experimental setup

Our NAF variational approximation utilized a single layer of flow, with $D = 32$ hidden units. The MADE network was chosen to have a single hidden layer and 10 times as many hidden units as the number of latent variables.

The standard ELBO objective was used for training, estimated with 100 latent samples. Optimization was performed with the Adam optimizer for 1000 epochs, after performing a grid search for the best learning rate over $\{0.1, 0.05, 0.025, 0.01, 0.005, 0.001, 0.0001, 0.00001\}$ as judged by that resulting in the lowest MSE averaged over 10 runs at 1000 epochs.

As a measure of predictive performance, we estimated the mean-squared-error as the expectation of the squared residual of the prediction from our Bayesian regression model using the formula,

$$\mathrm{MSE}(q, \mathcal{D}) = \mathbb{E}_{\mathbf{z}\sim q_\psi(\cdot)}\left[\mathbb{E}_{(\mathbf{x}',y')\sim\mathcal{D}}\left[\mathbb{E}_{y\sim p(\cdot|\mathbf{z},\mathbf{x}')}\left[(y-y')^2\right]\right]\right],$$

where $\mathbf{x}$ denotes the covariate (i.e. input) variable and y denotes the response (i.e. output) variable. We report a MC estimate of the MSE, taking 1000 samples of $\mathbf{z}$ from q_ψ, and 10 samples of y for each $(z, \mathbf{x}', y')$ in Table 2.

NUTS used 10 chains with a burn-in period of 1000, taking 100 samples of $\mathbf{z}$ from each chain to compute the posterior.

Appendix B. Models

The following models were used from Gelman and Hill (2006):

	model	Ch	N	M
(1)	`anova radon nopred`	22	919	88
(2)	`congress`	7	343	4
(3)	`earnings1`	7	1379	4
(4)	`earnings2`	7	1192	4
(5)	`earnings latin square`	13	1059	51
(6)	`earnings vary si`	13	1059	13
(7)	`election88`	14	2015	55
(8)	`election88`	19	2015	95
(9)	`hiv`	20	369	173
(10)	`radon complete pool`	12	919	3
(11)	`radon group`	12	919	92
(12)	`wells dae inter c`	5	3020	7
(13)	`wells dist`	5	3020	2

"Ch" refers to the book chapter where the model appears, N is the size of the data, and M the dimension of the latent space.