Implicit Reparameterization Gradients
Supplementary materials

A Testing of implicit gradient implementation

A simple test to verify the correctness of an implicit gradient implementation is to choose an appropriate function $f(\boldsymbol{z})$ and check that the Monte-Carlo averaging approximates the correct quantity:

$$\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z})} \left[f(\boldsymbol{z}) \right] \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_{\boldsymbol{z}} f(\boldsymbol{z_s}) \nabla_{\boldsymbol{\phi}} \boldsymbol{z}_s, \;\; \boldsymbol{z}_s \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}). \tag{10}$$

For $\mathrm{Gamma}(\alpha, 1)$, we can choose $f(z) = z$. Then, $\frac{\partial}{\partial \alpha} \mathbb{E}_{q_\alpha(z)} z = 1$, so the average stochastic gradient should be equal to 1. For $\mathrm{vonMises}(0, \kappa)$, an appropriate choice is $f(z) = \cos z$. Then, $\frac{\partial}{\partial \kappa} \mathbb{E}_{q_\kappa(z)} \cos z = \frac{\partial}{\partial \kappa} \frac{I_1(\kappa)}{I_0(\kappa)} = 1 - \frac{I_1(\kappa)}{\kappa I_0(\kappa)} - \left(\frac{I_1(\kappa)}{I_0(\kappa)} \right)^2$.

B Implementation details for the reparameterization gradient

We implement Eqn. (8) using an equivalent but more numerically stable expression:

$$\nabla_{\boldsymbol{\phi}} z = -\frac{\nabla_{\boldsymbol{\phi}} F(z|\boldsymbol{\phi})}{q_{\boldsymbol{\phi}}(z)} = -\exp(-\log q_{\boldsymbol{\phi}}(z)) \nabla_{\boldsymbol{\phi}} F(z|\boldsymbol{\phi}). \tag{11}$$

Gamma distribution. We perform forward-mode differentiation of the efficient computation method [2]. We use the implementation available in Eigen [5], which is based on the Cephes [10] library. A more advanced version of this method is available in SciPy [7]. For $z \geq 1$ and $z > \alpha$ this method uses the continued fraction expansion:

$$\gamma(z, \alpha) = 1 - \frac{\exp(-z) z^\alpha}{\Gamma(\alpha)} \cfrac{1}{z + \cfrac{1-\alpha}{1 + \cfrac{1}{z + \cfrac{2-\alpha}{1 + \cfrac{2}{z + \dots}}}}} \tag{12}$$

The expansion can be evaluated in the "direct order" using the Wallis algorithm [13]. For other values of the arguments, a series expansion is used:

$$\gamma(z, \alpha) = \frac{\exp(-z) z^\alpha}{\Gamma(\alpha + 1)} \left(1 + \sum_{k=1}^{\infty} \frac{z^k}{(\alpha+1)(\alpha+2)\dots(\alpha+r)} \right) \tag{13}$$

In both cases, all the operations are differentiable with respect to α, so forward-mode differentiation can be applied. We stop the computation as soon as the value of the derivative (not the CDF) starts changing by less than some small value. The maximum number of iterations is set to 200 for `float32` precision and 500 for `float64`. Additionally, we multiply the resulting derivative by $\exp(-\log \mathrm{Gamma}(z|\alpha, 1))$ in the same code. This improves the speed by about 30% and the accuracy by an order of magnitude, because some of the terms, including a Gamma function, cancel out.

Von Mises distribution. The CDF of a standardized von Mises distribution is given by the series

$$F(z|0, \kappa) = \int_{-\pi}^{z} \mathrm{vonMises}(t|0, \kappa) dt = \frac{z}{2\pi} + \frac{1}{\pi} \sum_{j=1}^{\infty} \frac{I_j(\kappa)}{I_0(\kappa)} \frac{\sin(j \cdot z)}{j}. \tag{14}$$

For smaller concentration parameters, $\kappa < 50$, the numerical method [6] first chooses the truncation point K for the series, and then computes the first K terms using an efficient backwards recursion. For larger κ, it computes the CDF of a Normal approximation for the von Mises. We use the implementation available in the SciPy library [7]. Again, forward-mode automatic differentiation can be used since all the operations with respect to κ are differentiable.

Table 5: The relative step size δ used for finite difference approximation of the CDF derivative.

	`float32`	`float64`
Gamma	10^{-3}	10^{-5}
Von Mises	10^{-1}	10^{-4}

C Accuracy and speed of the reparameterization gradient estimators

We start by describing how we computed the ground-truth value of the CDF derivatives and then provide the implementation details of the comparison.

Gamma distribution. The derivative of the CDF of a Gamma distribution can be obtained in terms of the hypergeometric function ${}_2F_2$ [16]:

$$\frac{\partial \gamma(z,\alpha)}{\partial \alpha} = \gamma(z,\alpha)(\log z - \psi(\alpha)) + {}_2F_2(\alpha,\alpha;\alpha+1,\alpha+1;-z)\frac{z^\alpha}{\alpha\Gamma(\alpha+1)}, \tag{15}$$

where $\psi(\alpha) = (\log\Gamma(\alpha))'$ is the digamma function and ${}_2F_2(\alpha,\alpha;\alpha+1,\alpha+1;-z) = \sum_{k=0}^{\infty}\frac{\alpha^2}{(\alpha+k)^2}\frac{(-z)^k}{k!}$. The function ${}_2F_2$ is implemented in `mpmath` package [11], allowing to evaluate this expression to arbitrary precision for comparison purposes. We compute it with the default settings that result in `float64` precision.

Von Mises distribution. Differentiating the series (14) with respect to κ using the identity $\frac{\partial}{\partial\kappa}I_j(\kappa) = \frac{j}{\kappa}I_j(\kappa) + I_{j+1}(\kappa)$ yields

$$\frac{\partial F(z|0,\kappa)}{\partial\kappa} = \frac{1}{\pi}\sum_{j=1}^{\infty}\left(\frac{\frac{j}{\kappa}I_j(\kappa)+I_{j+1}(\kappa)}{I_0(\kappa)} - \frac{I_j(\kappa)I_1(\kappa)}{(I_0(\kappa))^2}\right)\frac{\sin(j\cdot z)}{j}. \tag{16}$$

We compute this expression using the SciPy implementation of the Bessel functions by truncating the series at the 100^{th} term.

Details of the comparison. We first choose a grid of the parameters. For the Gamma distribution, we consider $\alpha \in \{1\times10^{-2}, 1\times10^{-1}, 1, 1\times10^{1}, 1\times10^{2}, 1\times10^{3}\}$, and for von Mises we consider $\kappa \in \{1\times10^{-2}, 1\times10^{-1}, 1, 1\times10^{1}\}$. Then, we sample 1000 random variables from the distribution for each value of the parameter. We report the relative step sizes $\delta > 0$ determined by grid search that we use for the finite difference approximation in Table 5. The timings are measured on a single core of an Intel Xeon CPU.

D Experimental details

RSVI details. We use the proposal distributions suggested in [12]. For $\mathrm{Gamma}(\alpha, 1)$, we employ the proposal distribution from Marsaglia and Tsang [9] which is explicitly reparameterizable using the standard Normal. For $\mathrm{vonMises}(0,\kappa)$, we use the wrapped Cauchy proposal distribution [1] that is explicitly reparameterizable using the Uniform distribution.

Gradient of the cross-entropy. The gradient variance is computed as

$$\mathbb{E}_{q_\phi(\boldsymbol{z})}\left(\frac{\partial}{\partial\phi}[-\log p(\boldsymbol{z})] - c\right)^2, \tag{17}$$

where the expectation is estimated using 1000 samples and $c = \frac{\partial}{\partial\phi}\mathbb{E}_{q_\phi(\boldsymbol{z})}[-\log p(\boldsymbol{z})]$ is the analytical gradient of the cross-entropy. The timings are measured on a single core of an Intel Xeon CPU.

Variance of the gradient during training. For LDA and VAE models, we estimate the variance of the gradient by reusing the exponential moving averages of the first and second moments computed by the Adam optimizer [8]. Specifically, denoting by $\boldsymbol{m}$ and $\boldsymbol{v}$ the estimates of the first and second moments of the gradient respectively, the variance estimate is $(\boldsymbol{v} - \boldsymbol{m}^2)$. We average this estimate over all the parameters. Note that we compute the variance of the gradient with respect to the model parameters: weights, biases and prior parameters.

Table 6: Hyperparameters used for the LDA experiments.

Dataset	20 Newsgroups	RCV1	RCV1
Model	LN-LDA, LDA	LN-LDA	LDA
Learning rate	3×10^{-4}	1×10^{-3}	1×10^{-3}
Layers in the inference network	3	1	2
Units per layer	300	200	250
Initial value of $\boldsymbol{\alpha}$	0.7	0.5	0.95
Burn-in epochs for $\boldsymbol{\alpha}$	350	7	5

Parameters of distributions. The positive-valued parameters (scale of Normal; all the parameters of Gamma, Beta and Dirichlet; concentration parameter of von Mises) are computed as a softplus of an unconstrained value. For all of these parameters except for the Normal scale, we additionally perform clipping to the $[10^{-3}, 10^3]$ range. This makes the analytical KL-divergence numerically stable in `float32` precision. A useful sanity check of numerical stability is that the KL-divergence is always non-negative.

The samples from von Mises distribution, and likewise the location parameter μ, can be equivalently represented as an angle $z \in [-\pi, \pi)$, or as a point on a circle (x, y). We find that learning in the second case is much easier. Thus, we compute the location parameter as $\mu = \texttt{atan2}(x, y)$, where x and y are unconstrained values, and transform the samples from the distribution: $z \to (\cos z, \sin z)$.

Latent Dirichlet Allocation. The 20 Newsgroups models are trained for 500 epochs, while the RCV1 ones for 10 epochs. We set the number of topics to 50. The inference network is a ReLU multilayer perceptron with the same number of units per layer. The optimization method is Adam [8] with $\beta_1 = 0.9$ and the batch size is 32. The model parameters are initialized using the Xavier initializer [4]: the truncated Normal distribution with zero mean and the variance of $\frac{2}{\texttt{fan\_in}+\texttt{fan\_out}}$. The prior parameters are fixed at the initial value for a number of epochs (burn-in period) and then trained jointly with other parameters. For the LN-LDA model [15], we tune the prior parameters of the underlying Dirichlet distribution for which the Laplace approximation is performed; we also checked that training the Normal prior parameters without any constraints does not improve the perplexity. We find that the architectural modifications suggested in [15], such as using dropout and batch normalization, do not lead to improved values of the perplexity, so we do not use them (they report the perplexity of 1059 for 20 Newsgroups dataset, while we obtain 875).

We find the key hyperparameters by Bayesian optimization of the validation set perplexity. The validation set consists of 10% random training documents for 20 Newsgroups and 1% random training documents for RCV1. A separate search is performed for each dataset (20 Newsgroups, RCV1) and model (LN-LDA, LDA (implicit)) combination, for a total of four runs. For 20 Newsgroups the obtained hyperparameters are very similar for both models, so we use the same values. The hyperparameter values are shown in Table 6.

We use GenSim library [14] implementation of stochastic variational inference (SVI). We train for the same number of epochs and with the same number of topics as above. We perform a grid search for the key hyperparameters. For 20 Newsgroups, we use `chunk_size=1000` and `decay=0.5`, while for RCV1 we set `chunk_size=2000` and `decay=0.5`. In both cases, we set `alpha="auto"`, meaning that the prior hyperparameters $\boldsymbol{\alpha}$ are learned. The remaining options were set to the default values.

We performed control experiments on the 20 Newsgroups dataset showing that (i) using a fixed prior distribution increases the perplexity by 60 points; (ii) computing the KL using sampling instead of an analytical expression increases the perplexity by 80 points. The latter result highlights the importance of using variational posteriors that allow for analytical KL estimation when dealing with "sparse" distributions that have density asymptotes.

Variational autoencoder. We base our experimental setup on the one from Davidson et al. [3]: a fully-connected ReLU network with two layers of 256 and 128 units as the encoder, a two-layer fully-connected ReLU network with 128 and 256 units as the decoder, minibatch size of 64, Adam optimizer [8], and annealing the KL term from 0 to 1 over the first 10^5 minibatches. The only differences are (i) we do not perform early stopping and always train for 2 million minibatches; (ii) we train each model with the learning rates of 10^{-3} and 10^{-4} and choose the best-performing one.

Table 7: Comparison of implicit reparameterization gradients and RSVI for the generative modeling task on MNIST dataset. Test negative log-likelihood (lower is better) mean $\pm$ standard deviation over 5 runs.

Prior	Variational posterior	Training method	$D = 2$	$D = 5$	$D = 10$	$D = 20$	$D = 40$
$\mathcal{N}(0, 1)$	$\mathcal{N}(\mu, \sigma^2)$	Explicit	131.1 ± 0.6	107.9 ± 0.4	92.5 ± 0.2	88.1 ± 0.2	88.1 ± 0.0
$\text{Gamma}(0.3, 0.3)$	$\text{Gamma}(\alpha, \beta)$	Implicit	132.4 ± 0.3	108.0 ± 0.3	94.0 ± 0.3	90.3 ± 0.2	90.6 ± 0.2
		RSVI $B = 20$	132.3 ± 0.2	108.5 ± 0.3	94.3 ± 0.9	90.1 ± 0.1	90.6 ± 0.1
$\text{Gamma}(10, 10)$	$\text{Gamma}(\alpha, \beta)$	Implicit	135.0 ± 0.2	107.0 ± 0.2	92.3 ± 0.2	88.3 ± 0.2	88.3 ± 0.1
		RSVI $B = 20$	131.6 ± 0.3	107.1 ± 0.1	92.2 ± 0.1	88.2 ± 0.1	88.2 ± 0.1
$\text{Uniform}(0, 1)$	$\text{Beta}(\alpha, \beta)$	Implicit	128.3 ± 0.2	107.4 ± 0.2	94.1 ± 0.1	88.9 ± 0.1	88.6 ± 0.1
		RSVI $B = 20$	128.9 ± 0.8	107.3 ± 0.1	94.3 ± 0.1	88.8 ± 0.1	88.5 ± 0.1
$\text{Beta}(10, 10)$	$\text{Beta}(\alpha, \beta)$	Implicit	131.1 ± 0.4	106.7 ± 0.1	92.1 ± 0.2	87.8 ± 0.1	87.7 ± 0.1
		RSVI $B = 20$	131.7 ± 0.4	106.9 ± 0.1	92.2 ± 0.1	87.7 ± 0.1	87.6 ± 0.1
$\text{Uniform}(-\pi, \pi)$	$\text{vonMises}(\mu, \kappa)$	Implicit	127.6 ± 0.4	107.5 ± 0.4	94.4 ± 0.5	90.9 ± 0.1	91.5 ± 0.4
		RSVI	129.1 ± 0.4	107.6 ± 0.3	96.0 ± 0.5	92.8 ± 0.2	92.8 ± 0.2
$\text{vonMises}(0, 10)$	$\text{vonMises}(\mu, \kappa)$	Implicit	130.7 ± 0.8	107.5 ± 0.5	92.3 ± 0.2	87.8 ± 0.2	87.9 ± 0.3
		RSVI	130.4 ± 0.7	107.8 ± 0.5	93.0 ± 0.1	88.7 ± 0.2	88.7 ± 0.1

The model parameters are initialized from the truncated Normal distribution with zero mean and the variance of $\frac{1}{\texttt{fan\_in}}$. We estimate the log-likelihood using importance sampling with 500 samples.

We present the comparison between implicit gradients and RSVI gradients in Table 7. We find that while they perform similarly for Gamma and Beta distributions, for the von Mises distribution implicit gradients obtain better results, since there is no analogue of the shape augmentation parameter B for this distribution.

References

[1] D. Best and N. I. Fisher. "Efficient simulation of the von Mises distribution". In: *Applied Statistics* (1979), pp. 152–157.

[2] G. P. Bhattacharjee. "Algorithm AS 32: The Incomplete Gamma Integral". In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 19.3 (1970), pp. 285–287. ISSN: 00359254, 14679876.

[3] T. R. Davidson, L. Falorsi, N. De Cao, T. Kipf, and J. M. Tomczak. "Hyperspherical Variational Auto-Encoders". In: *Conference on Uncertainty in Artificial Intelligence* (2018).

[4] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *International Conference on Artificial Intelligence and Statistics*. 2010, pp. 249–256.

[5] G. Guennebaud, B. Jacob, et al. *Eigen v3*. 2010. URL: http://eigen.tuxfamily.org.

[6] G. W. Hill. "Algorithm 518: Incomplete Bessel Function I_0. The Von Mises Distribution". In: *ACM Transactions on Mathematical Software (TOMS)* 3.3 (1977), pp. 279–284.

[7] E. Jones, T. Oliphant, P. Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001. URL: http://www.scipy.org/.

[8] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations* (2015).

[9] G. Marsaglia and W. W. Tsang. "A simple method for generating gamma variables". In: *ACM Transactions on Mathematical Software (TOMS)* 26.3 (2000), pp. 363–372.

[10] S. Moshier. *Cephes math library*. 2000. URL: http://www.moshier.net.

[11] F. Johansson et al. *mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 1.0.0)*. http://mpmath.org/. 2017.

[12] C. Naesseth, F. Ruiz, S. Linderman, and D. Blei. "Reparameterization gradients through acceptance-rejection sampling algorithms". In: *International Conference on Artificial Intelligence and Statistics* (2017), pp. 489–498.

[13] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.

[14] R. Řehůřek and P. Sojka. “Software Framework for Topic Modelling with Large Corpora”. English. In: *LREC 2010 Workshop on New Challenges for NLP Frameworks* (May 2010), pp. 45–50.

[15] A. Srivastava and C. Sutton. “Autoencoding variational inference for topic models”. In: *International Conference on Learning Representations* (2017).

[16] The Wolfram Functions Site. *Derivative of GammaRegularized with respect to a*. 2001. URL: http://functions.wolfram.com/06.08.20.0001.01.