REINFORCing Concrete with REBAR

George Tucker [1] **Andriy Mnih** [2] **Chris J. Maddison** [2 3] **Dieterich Lawson** [1] **Jascha Sohl-Dickstein** [1]

Abstract

Learning in models with discrete latent variables is challenging due to high variance gradient estimators. Generally, approaches have relied on control variates to reduce the variance of the REINFORCE estimator. Recent work (Jang et al., 2016; Maddison et al., 2016) has taken a different approach, introducing a continuous relaxation of discrete variables to produce low-variance, but biased, gradient estimates. In this work, we combine the two approaches through a novel control variate that produces low-variance, *unbiased* gradient estimates. Then, we introduce a novel continuous relaxation and show that the tightness of the relaxation can be adapted online, removing it as a hyperparameter. We show state-of-the-art variance reduction on benchmark generative modeling and structured prediction tasks, generally leading to faster convergence to a better final log likelihood.

1. Introduction

Models with discrete latent variables are ubiquitous in machine learning: mixture models, Markov Decision Processes in reinforcement learning (RL), generative models for structured prediction, and, recently, models with hard attention (Mnih et al., 2014) and memory networks (Zaremba & Sutskever, 2015). However, when the discrete latent variables cannot be marginalized out analytically, maximizing objectives over these models using REINFORCE-like methods (Williams, 1992) is challenging due to high-variance gradient estimates obtained from sampling. Most approaches to reducing this variance have focused on developing clever control variates (Mnih & Gregor, 2014; Titsias & Lázaro-Gredilla, 2015; Gu et al., 2015; Mnih & Rezende, 2016). Recently, Jang et al. (2016) and Maddison et al. (2016) independently introduced a novel distribution, the Gumbel-Softmax or Concrete distribution, that continuously relaxes discrete random variables. Replacing every discrete random variable in a model with a Concrete random variable results in a continuous model where the reparameterization trick is applicable (Kingma & Welling, 2013; Rezende et al., 2014). The gradients are biased with respect to the discrete model, but can be used effectively to optimize large models. The tightness of the relaxation is controlled by a temperature hyperparameter which must be tuned to balance bias versus variance.

We sought an estimator that is low variance, unbiased, and does not require tuning additional hyperparameters. To construct such an estimator, we introduce a control variate based on the difference between the REINFORCE and the reparameterization trick gradient estimators for the relaxed model. We call this the REBAR gradient estimator, because it combines REINFORCE gradients with gradients of the Concrete relaxation. Next, we show that a small modification to the Concrete relaxation connects REBAR to MuProp in the high temperature limit. Finally, because REBAR is unbiased for all temperatures, we show that the temperature can be optimized online to reduce variance further and relieve the burden of setting an additional hyperparameter.

In our experiments, we use REBAR to train sigmoid belief networks (SBNs) for generative modeling on the MNIST and Omniglot datasets and structured prediction on MNIST. Across tasks, we show that REBAR has state-of-the-art variance reduction which generally translates to faster convergence and better final log likelihoods. Although we focus on binary variables for simplicity, this work is equally applicable to categorical variables.

2. Background

For clarity, we first consider a simplified scenario and expand on it in the Appendix. Let $b \sim \text{Bernoulli}(\theta)$ be a vector of factorial binary random variables parameterized by θ. We wish to maximize $\mathbb{E}_{p(b)}[f(b,\theta)]$, where $f(b,\theta)$ is differentiable w.r.t. θ, and we suppress the dependence of $p(b)$ on θ to reduce notational clutter. This covers a wide range of discrete latent variable problems; for example, in variational inference $f(b,\theta)$ would be the stochastic variational lower bound.

[1]Google Brain, Mountain View, CA, USA [2]DeepMind, London, UK [3]University of Oxford, Oxford, UK. Correspondence to: George Tucker <gjt@google.com>.

Proceedings of the ICML 17 Workshop on Deep Structured Prediction, Sydney, Australia, PMLR 70, 2017.

Typically, this problem has been approached by gradient ascent, which requires efficiently estimating

$$\frac{d}{d\theta} \mathop{\mathbb{E}}_{p(b)} [f(b, \theta)] = \mathop{\mathbb{E}}_{p(b)} \left[\frac{\partial f(b, \theta)}{\partial \theta} + f(b, \theta) \frac{\partial}{\partial \theta} \log p(b) \right].$$

In practice, the first term can be estimated effectively with a single Monte Carlo sample, however, a naïve single sample estimator of the second term has high variance. Because the dependence of $f(b, \theta)$ on θ is straightforward to account for and to simplify exposition, we assume that $f(b, \theta) = f(b)$ does not depend on θ and concentrate on the second term.

2.1. Variance reduction through control variates

Paisley et al. (2012); Ranganath et al. (2014); Mnih & Gregor (2014); Gu et al. (2015) show that carefully designed control variates can reduce the variance of the second term significantly. Unfortunately, even with a control variate, the second term can still have large variance.

2.2. Continuous relaxations for discrete variables

Alternatively, following Maddison et al. (2016), we can parameterize b as $b = H(z)$, where H is the element-wise hard threshold function[1] and z is a vector of factorial Logistic random variables defined element-wise by

$$z \coloneqq g(u, \theta) \coloneqq \log \frac{\theta}{1 - \theta} + \log \frac{u}{1 - u},$$

where $u \sim \text{Uniform}(0, 1)$. Notably, z is differentiably reparameterizable (Kingma & Welling, 2013; Rezende et al., 2014), but the discontinuous hard threshold function prevents us from using the reparameterization trick directly. Replacing all occurrences of the hard threshold function with a continuous relaxation $H(z) \approx \sigma_\lambda(z) \coloneqq \sigma\left(\frac{z}{\lambda}\right) = (1 + \exp(-\frac{z}{\lambda}))^{-1}$ however results in a reparameterizable computational graph[2]. $\lambda > 0$ can be thought of as a temperature that controls the tightness of the relaxation (at low temperatures, the relaxation is nearly tight). This generally results in a low-variance, but biased Monte Carlo estimator for the discrete model. In practice, λ must be tuned to balance bias versus variance.

3. REBAR

We seek a low-variance, unbiased gradient estimator. Inspired by the Concrete relaxation, our strategy will be to construct a control variate based on the difference between the REINFORCE gradient estimator for the relaxed model and the gradient estimator from the reparameterization trick.

We can decompose the objective into a reparameterizable term and a residual

$$\frac{\partial}{\partial \theta} \mathop{\mathbb{E}}_{p(b)} [f(b)] = \eta \frac{\partial}{\partial \theta} \mathop{\mathbb{E}}_{p(z)} [f(\sigma_\lambda(z))] + \frac{\partial}{\partial \theta} \mathop{\mathbb{E}}_{p(b)} \left[f(b) - \eta \mathop{\mathbb{E}}_{p(z|b)} [f(\sigma_\lambda(z))] \right]$$

where η is a learned scaling. Then, we can expand the second term similarly to REINFORCE

$$\frac{\partial}{\partial \theta} \mathop{\mathbb{E}}_{p(b)} \left[f(b) - \eta \mathop{\mathbb{E}}_{p(z|b)} [f(\sigma_\lambda(z))] \right] = \mathop{\mathbb{E}}_{p(b)} \left[\left(f(b) - \eta \mathop{\mathbb{E}}_{p(z|b)} [f(\sigma_\lambda(z))] \right) \frac{\partial}{\partial \theta} \log p(b) - \eta \frac{\partial}{\partial \theta} \mathop{\mathbb{E}}_{p(z|b)} [f(\sigma_\lambda(z))] \right].$$

Our key insight is that both $p(z)$ and $p(z|b)$ are differentiably reparameterizable (Kingma & Welling, 2013; Rezende et al., 2014), so we can estimate most terms with low variance. Putting the terms together, we arrive at

$$\mathop{\mathbb{E}}_{p(u,v)} \left[\left[f(H(z)) - \eta f(\sigma_\lambda(\tilde{z}) \right] \frac{\partial}{\partial \theta} \log p_b(H(z)) + \eta \frac{\partial}{\partial \theta} f(\sigma_\lambda(z)) - \eta \frac{\partial}{\partial \theta} f(\sigma_\lambda(\tilde{z})) \right],$$

where $u, v \sim \text{Uniform}(0, 1)$, $z \equiv g(u, \theta)$, and $\tilde{z} \equiv \tilde{g}(v, H(z), \theta)$ is the differentiable reparameterization for $z|b$ (Appendix 6.2). The REBAR estimator is the single sample Monte Carlo estimator of this expectation. To reduce computation and variance, we couple u and v using common random numbers (Owen, 2013). We estimate η by minimizing the variance of the Monte Carlo estimator with SGD.

3.1. Rethinking the relaxation

For the relaxation, $\sigma_\lambda(z) \to \frac{1}{2}$ as $\lambda \to \infty$; this is clearly a poor approximation and will lead to an ineffective control variate. Alternatively, consider the relaxation

$$H(z) \approx \sigma \left(\frac{\lambda^2 + \lambda + 1}{\lambda(\lambda + 1)} \log \frac{\theta}{1 - \theta} + \frac{1}{\lambda} \log \frac{u}{1 - u} \right).$$

As $\lambda \to \infty$, the relaxation converges to the mean, θ, and still as $\lambda \to 0$, the relaxation becomes exact.

Then, as $\lambda \to \infty$, the REBAR estimator converges to $[f(H(z)) - \eta f(\theta)] \frac{\partial}{\partial \theta} \log p_b(H(z))$, which is MuProp without the linear term. We refer to this estimator as SimpleMuProp in the results.

[1] $H(z) = 1$ if $z \geq 0$ and $H(z) = 0$ if $z < 0$.

[2] The choice of continuous relaxation for f is not unique, and the effectiveness and computational cost of the method depend on this choice. For the models considered here, f is also well defined for continuous values, and empirically this performs well (Maddison et al., 2016; Jang et al., 2016).

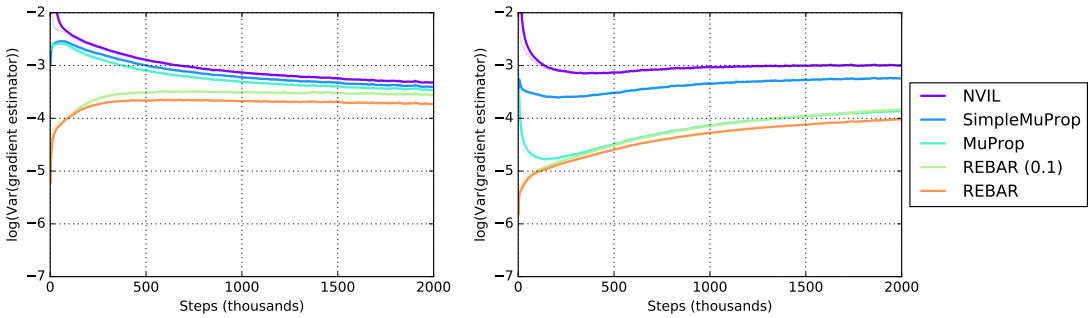

Figure 1. Log variance of the gradient estimator for the single layer nonlinear model on the MNIST generative modeling task and the structured prediction task. All of the estimators are unbiased, so their variance is directly comparable. We estimated moments from exponential moving averages (with decay=0.999; we found that the results were robust to the exact value). The temperature is shown in parenthesis where relevant.

3.2. Optimizing temperature (λ)

The REBAR gradient estimator is unbiased for *any* choice of $\lambda > 0$, so we can optimize λ to minimize the variance of the estimator without affecting its unbiasedness. This allows the tightness of the relaxation to be adapted online jointly with the optimization of the parameters and relieves the burden of choosing λ ahead of time.

4. Experiments

As our goal was variance reduction to improve optimization, we compared our method to the state-of-the-art unbiased single-sample gradient estimators, NVIL (Mnih & Gregor, 2014) and MuProp (Gu et al., 2015), and the state-of-the-art biased single-sample gradient estimator Gumbel-Softmax/Concrete (Jang et al., 2016; Maddison et al., 2016) by measuring the variance of the gradient estimators and their progress on the training objective. We follow the experimental setup established in (Maddison et al., 2016) to evaluate the methods on generative modeling and structured prediction tasks.

4.1. Learning sigmoid belief networks (SBNs)

We trained SBNs on several standard benchmark tasks. We follow the setup established in (Maddison et al., 2016). Briefly, we used the statically binarized MNIST digits from Salakhutdinov & Murray (2008) and a fixed binarization of the Omniglot character dataset. We used the standard splits into training, validation and test sets. The network used several layers of 200 stochastic binary units interleaved with deterministic nonlinearities. In our experiments, we used either a linear deterministic layer (denoted linear) or 2 layers of 200 tanh units (denoted nonlinear).

4.1.1. Generative modeling

For generative modeling, we maximized a single-sample variational lower bound on the log likelihood. We performed amortized inference (Kingma & Welling, 2013; Rezende et al., 2014) with an inference network with similar architecture in the reverse direction.

To measure the variance of the gradient estimators, we follow a single optimization trajectory and use the same random numbers for all methods. We plot the log variance of the unbiased gradient estimators in Figure 1 and Appendix Figure 2 for MNIST (and Appendix Figure 4 for Omniglot). REBAR produced the lowest variance across linear and nonlinear models for both tasks. The reduction in variance was especially large for the linear models. For the nonlinear model, REBAR (0.1) reduced variance at the beginning of training, but its performance degraded later in training. REBAR was able to adaptively change the temperature as optimization progressed and retained superior variance reduction. We also observed that SimpleMuProp was a surprisingly strong baseline that improved significantly over NVIL. It performed similarly to MuProp despite not explicitly using the gradient of f.

Generally, lower variance gradient estimates lead to faster optimization of the objective and convergence to a better final value (Table 1). For the nonlinear model, the Concrete estimator underperformed optimizing the training objective on both datasets.

4.1.2. Structured prediction on MNIST

We implemented the structured prediction task described by Raiko et al. (2014), where we modeled the bottom half of an MNIST digit conditional on the top half. We optimized the single sample lower bound on the log likelihood.

As before, we found that REBAR significantly reduced variance (Appendix Figure 6). In some configurations, MuProp excelled, especially with the single layer linear model. Again, the training objective performance generally mirrored the reduction in variance of the gradient estimator (Table 1, Appendix Figure 7).

MNIST gen.	NVIL	MuProp	REBAR (0.1)	REBAR	Concrete (0.1)
Linear 1 layer	-112.61	-111.8	-111.9	-111.5	$\mathbf{-111.34}$
Linear 2 layer	-99.44	-99.16	-99.14	$\mathbf{-98.82}$	-99.6
Nonlinear	-102.1	-101.5	-101.7	$\mathbf{-101.14}$	-103.2
Omniglot gen.					
Linear 1 layer	-117.43	-117.04	-116.93	$\mathbf{-116.86}$	-117.3
Linear 2 layer	-109.93	-109.63	$\mathbf{-109.07}$	-109.14	-110
Nonlinear	-110.4	-109.7	-109	$\mathbf{-108.84}$	-110.9
MNIST struct. pred.					
Linear 1 layer	-69.15	$\mathbf{-64.31}$	-65.75	-65.24	-65.53
Linear 2 layer	-68.88	-63.68	-65.53	$\mathbf{-61.74}$	-66.89
Nonlinear layer	-54.01	-47.58	-47.34	$\mathbf{-46.44}$	-47.09

Table 1. Mean training variational lower bound over 4 trials with different random initializations. The standard error of the mean is given in Appendix Table 2. We bolded the best performing method (up to standard error) for each task. We report trials using the best performing learning rate for each task.

5. Conclusion

Inspired by the Concrete relaxation, we introduced REBAR, a novel control variate for REINFORCE, and demonstrated that it greatly reduces the variance of the gradient estimator. We also showed that with a modification to the relaxation, REBAR and MuProp are closely related in the high temperature limit. Moreover, we showed that we can adapt the temperature online and that it further reduces variance.

It would be natural to explore the extension to the multi-sample case (e.g., VIMCO (Mnih & Rezende, 2016)), to leverage the layered structure in our models using Q-functions, and to apply this approach to reinforcement learning.

Acknowledgements

We thank Ben Poole and Eric Jang for helpful discussions and assistance replicating their results.

References

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

Gu, Shixiang, Levine, Sergey, Sutskever, Ilya, and Mnih, Andriy. Muprop: Unbiased backpropagation for stochastic neural networks. *arXiv preprint arXiv:1511.05176*, 2015.

Jang, Eric, Gu, Shixiang, and Poole, Ben. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, Diederik P and Welling, Max. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Maddison, Chris J, Mnih, Andriy, and Teh, Yee Whye. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Mnih, Andriy and Gregor, Karol. Neural variational inference and learning in belief networks. In *Proceedings of The 31st International Conference on Machine Learning*, pp. 1791–1799, 2014.

Mnih, Andriy and Rezende, Danilo. Variational inference for monte carlo objectives. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 2188–2196, 2016.

Mnih, Volodymyr, Heess, Nicolas, Graves, Alex, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pp. 2204–2212, 2014.

Owen, Art B. *Monte Carlo theory, methods and examples*. 2013.

Paisley, John, Blei, David, and Jordan, Michael. Variational bayesian inference with stochastic search. *arXiv preprint arXiv:1206.6430*, 2012.

Raiko, Tapani, Berglund, Mathias, Alain, Guillaume, and Dinh, Laurent. Techniques for learning binary stochastic feedforward neural networks. *arXiv preprint arXiv:1406.2989*, 2014.

Ranganath, Rajesh, Gerrish, Sean, and Blei, David M. Black box variational inference. In *AISTATS*, pp. 814–822, 2014.

Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of The 31st International Conference on Machine Learning*, pp. 1278–1286, 2014.

Salakhutdinov, Ruslan and Murray, Iain. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*, pp. 872–879. ACM, 2008.

Titsias, Michalis K and Lázaro-Gredilla, Miguel. Local expectation gradients for black box variational inference. In *Advances in Neural Information Processing Systems*, pp. 2638–2646, 2015.

Williams, Ronald J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Zaremba, Wojciech and Sutskever, Ilya. Reinforcement learning neural Turing machines. *arXiv preprint arXiv:1505.00521*, 362, 2015.

6. Appendix

6.1. Review of control variates

Suppose we want to estimate $\mathbb{E}_x[f(x)]$ for an arbitrary function f. The variance of the naive Monte Carlo estimator $\mathbb{E}_x[f(x)] \approx \frac{1}{k}\sum_i f(x^i)$, with $x^1, ..., x^n \sim p(x)$, can be reduced by introducing a control variate $g(x)$. In particular,

$$\mathbb{E}[f(x)] \approx \left(\frac{1}{k}\sum_i f(x^i) - \eta g(x^i)\right) + \eta\,\mathbb{E}[g(x)]]$$

is an unbiased estimator for any value of η. We can choose η to minimize the variance of the estimator and it is straightforward to show that the optimal one is

$$\eta = \frac{\mathrm{Cov}(f,g)}{\mathrm{Var}(g)},$$

and it reduces the variance of the estimator by $(1-\rho(f,g)^2)$. So, if we can find a g that is correlated with f, we can reduce the variance of the estimator. If we cannot compute $\mathbb{E}[g]$, we can use a low-variance estimator $\hat{g}$. This is the approach we take. Of course, we could define $\tilde{g} = g - \hat{g}$, which has zero mean, as the control variate, however, this obscures the interpretability of the control variate.

6.2. Reparameterization for $z|b$

Now, we describe how to reparameterize $p(z|b)$. We can sample from $z|b$ by noting the point u' where $g(u', \theta) = 0$ and sampling $v \sim \mathrm{Uniform}(0, 1)$ and then scaling it to the interval $[u', 1]$ if $b = 1$ or $[0, u']$ otherwise. Then we can apply g to this value. Define this composed function as $\tilde{g}(v, b, \theta)$. Note that $\tilde{g}$ is also differentiable.

6.3. Multilayer stochastic networks

Suppose we have multiple layers of stochastic units (i.e., $b = \{b_1, b_2, \ldots, b_n\}$) where, $p(b)$ factorizes as

$$p(b_{1:n}) = p(b_1)p(b_2|b_1)\cdots p(b_n|b_{n-1}),$$

and similarly for the underlying Logistic random variables $p(z_{1:n})$ recalling that $b_i = H(z_i)$. We can define a relaxed distribution over $z_{1:n}$ where we replace the hard threshold function $H(z)$ with a continuous relaxation $\sigma_\lambda(z)$. We refer to the relaxed distribution as $q(z_{1:n})$.

We can take advantage of the structure of p, by using the fact that the high variance REINFORCE term of the gradient also decomposes

$$\mathop{\mathbb{E}}_{p(b)}\left[f(b)\frac{\partial}{\partial\theta}\log p(b)\right] = \sum_i \mathop{\mathbb{E}}_{p(b)}\left[f(b)\frac{\partial}{\partial\theta}\log p(b_i|b_{i-1})\right].$$

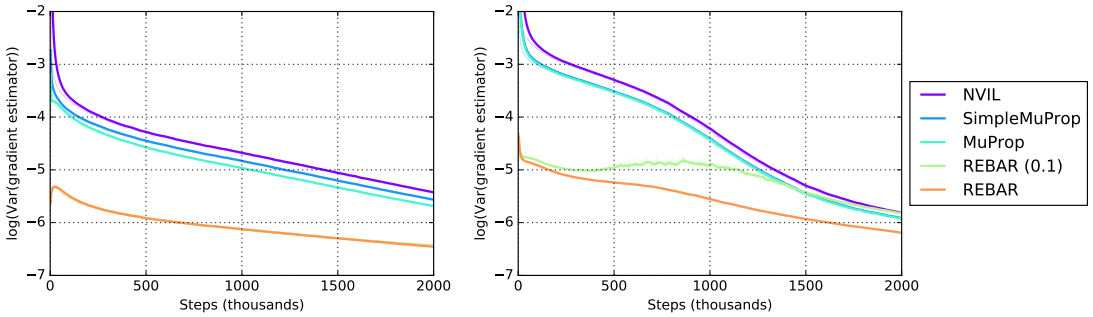

Figure 2. Log variance of the gradient estimator for the one layer linear model (left) and two layer linear model (right) on the MNIST generative modeling task.

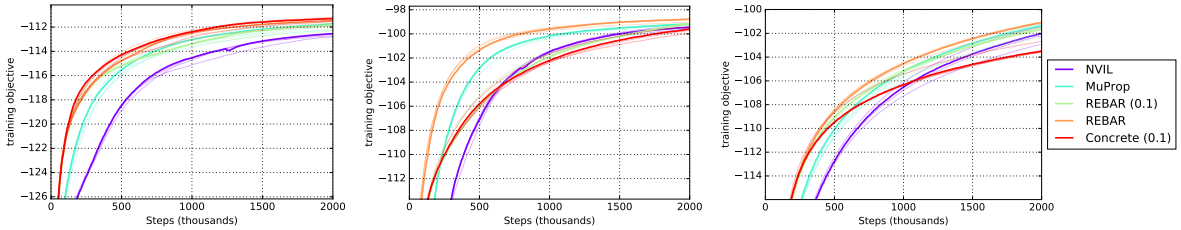

Figure 3. Training variational lower bound for the single layer linear model (left), the two layer linear model (middle), and the single layer nonlinear model (right) on the MNIST generative modeling task. We plot 4 trials over different random initializations for each method with the median trial highlighted.

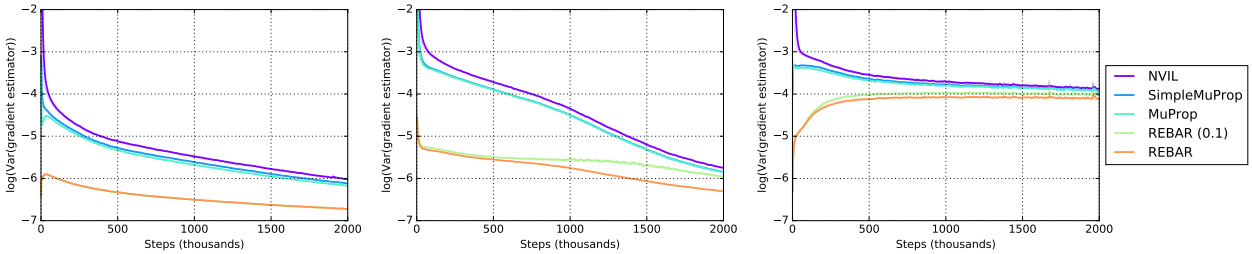

Figure 4. Log variance of the gradient estimator for the single layer linear model (left), the two layer linear model (middle), and the single layer nonlinear model (right) on the Omniglot generative modeling task.

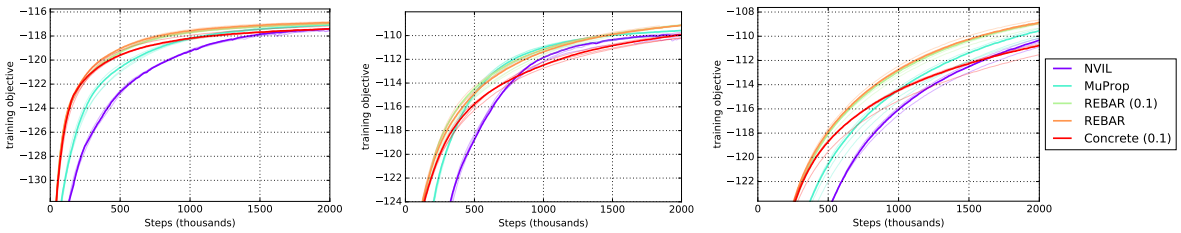

Figure 5. Training variational lower bound for the single layer linear model (left), the two layer linear model (middle), and the single layer nonlinear model (right) on the Omniglot generative modeling task. We plot 4 trials over different random initializations for each method with the median trial highlighted.

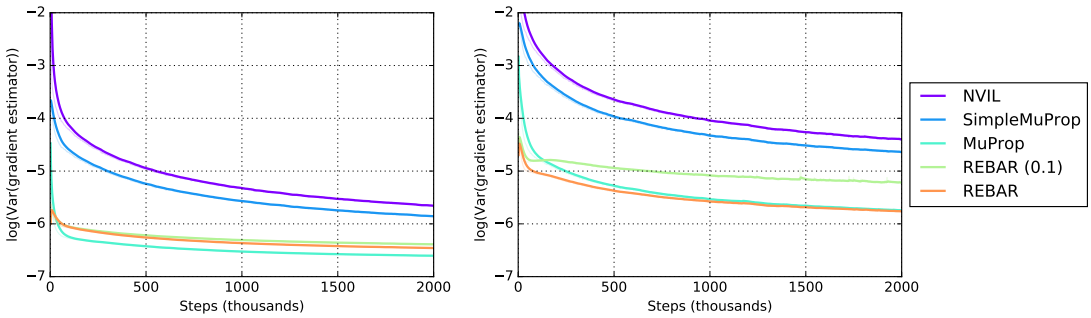

Figure 6. Log variance of the gradient estimator for the single layer linear model (left) and two layer linear model (right) on the structured prediction task.

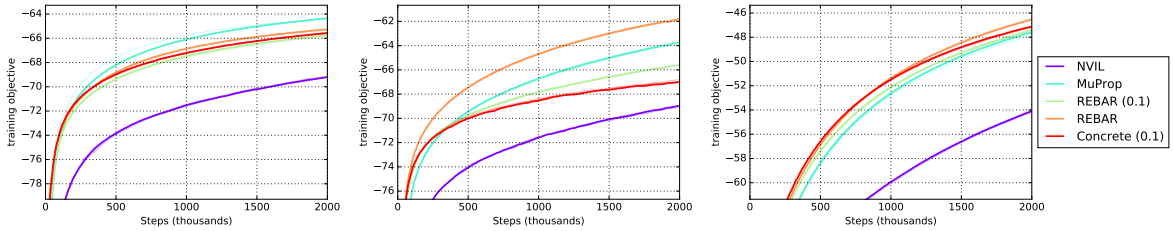

Figure 7. Training variational lower bound for the single layer linear model (left), the two layer linear model (middle), and the single layer nonlinear model (right) on the structured prediction task. We plot 4 trials over different random initializations for each method with the median trial highlighted.

MNIST gen.	NVIL	MuProp	REBAR (0.1)	REBAR	Concrete (0.1)
Linear 1 layer	-112.61 ± 0.05	-111.8 ± 0.1	-111.9 ± 0.2	$\mathbf{-111.5 \pm 0.1}$	$\mathbf{-111.34 \pm 0.06}$
Linear 2 layer	-99.44 ± 0.03	-99.16 ± 0.03	-99.14 ± 0.03	$\mathbf{-98.82 \pm 0.06}$	-99.6 ± 0.1
Nonlinear	-102.1 ± 0.3	-101.5 ± 0.2	-101.7 ± 0.2	$\mathbf{-101.14 \pm 0.09}$	-103.2 ± 0.2
Omniglot gen.					
Linear 1 layer	-117.43 ± 0.04	-117.04 ± 0.02	-116.93 ± 0.03	$\mathbf{-116.86 \pm 0.04}$	-117.3 ± 0.06
Linear 2 layer	-109.93 ± 0.1	-109.63 ± 0.08	$\mathbf{-109.07 \pm 0.02}$	-109.14 ± 0.03	-110 ± 0.08
Nonlinear	-110.4 ± 0.1	-109.7 ± 0.2	-109 ± 0.08	$\mathbf{-108.84 \pm 0.07}$	-110.9 ± 0.2
MNIST struct. pred.					
Linear 1 layer	-69.15 ± 0.02	$\mathbf{-64.31 \pm 0.01}$	-65.75 ± 0.02	-65.244 ± 0.009	-65.53 ± 0.01
Linear 2 layer	-68.88 ± 0.04	-63.68 ± 0.02	-65.525 ± 0.004	$\mathbf{-61.74 \pm 0.02}$	-66.89 ± 0.04
Nonlinear layer	-54.01 ± 0.03	-47.58 ± 0.04	-47.34 ± 0.02	$\mathbf{-46.44 \pm 0.03}$	-47.09 ± 0.02

Table 2. Mean training variational lower bound over 4 trials with different random initializations and standard error of the mean. We report trials using the best performing learning rate for each task.

Focusing on the i^{th} term, we have

$$\mathop{\mathbb{E}}_{p(b)}\left[f(b)\frac{\partial}{\partial\theta}\log p(b_i|b_{i-1})\right] = \mathop{\mathbb{E}}_{p(b_{1:i-1})}\left[\mathop{\mathbb{E}}_{p(b_i|b_{i-1})}\left[\mathop{\mathbb{E}}_{p(b_{i+1:n}|b_i)}\left[f(b)\right]\frac{\partial}{\partial\theta}\log p(b_i|b_{i-1})\right]\right],$$

which suggests the following control variate

$$\mathop{\mathbb{E}}_{p(z_i|b_i,b_{i-1})}\left[\mathop{\mathbb{E}}_{q(z_{i+1:n}|z_i)}\left[f(b_{1:i-1},\sigma_\lambda(z_{i:n}))\right]\right]\frac{\partial}{\partial\theta}\log p(b_i|b_{i-1})$$

for the middle expectation. Similarly to the single layer case, we can debias the control variate with terms that are reparameterizable. Note that due to the switch between sampling from p and sampling from q, this approach requires n passes through the network (one pass per layer).

6.4. Test log-likelihood

Although our primary focus was optimization, for completeness, we include results on the test set in Appendix Table 3 computed with a 100-sample lower bound (Burda et al., 2015). Improvements on the training variational lower bound do not directly translate into improved test log likelihood. Previous work (Maddison et al., 2016) showed that regularizing the inference network alone was sufficient

MNIST gen.	NVIL	MuProp	REBAR (0.1)	REBAR	Concrete (0.1)
Linear 1 layer	-108.4 ± 0.2	-108 ± 0.2	-107.7 ± 0.1	-107.48 ± 0.09	$\mathbf{-106.76 \pm 0.09}$
Linear 2 layer	-96.39 ± 0.08	-96.11 ± 0.08	$\mathbf{-95.6 \pm 0.04}$	$\mathbf{-95.67 \pm 0.03}$	$\mathbf{-95.64 \pm 0.06}$
Nonlinear	-100 ± 0.1	-100.5 ± 0.1	-100.6 ± 0.1	-100.59 ± 0.07	$\mathbf{-99.64 \pm 0.07}$
Omniglot gen.					
Linear 1 layer	$\mathbf{-117.54 \pm 0.06}$	-117.7 ± 0.04	-117.67 ± 0.07	-117.72 ± 0.03	$\mathbf{-117.54 \pm 0.04}$
Linear 2 layer	-111.46 ± 0.04	-111.24 ± 0.06	$\mathbf{-110.83 \pm 0.04}$	$\mathbf{-110.87 \pm 0.02}$	-111.29 ± 0.03
Nonlinear	$\mathbf{-116.75} \pm 0.05$	-117.62 ± 0.07	-117.89 ± 0.04	-118.2 ± 0.1	$\mathbf{-116.73 \pm 0.1}$
MNIST struct. pred.					
Linear 1 layer	-66.11 ± 0.02	-65.63 ± 0.01	-65.59 ± 0.01	-65.58 ± 0.03	$\mathbf{-65.31 \pm 0.02}$
Linear 2 layer	-63.22 ± 0.02	-62.08 ± 0.05	-62.07 ± 0.04	$\mathbf{-61.68 \pm 0.02}$	-61.919 ± 0.01
Nonlinear	-61.22 ± 0.04	-61.43 ± 0.03	-61.27 ± 0.02	-61.27 ± 0.02	$\mathbf{-61.05 \pm 0.04}$

Table 3. Mean test 100-sample variational lower bound on the log likelihood (Burda et al., 2015) over 4 random initializations with standard error of the mean. We report the best performing learning rate for each task based on the validation set.

to prevent overfitting. This led us to hypothesize that the overfitting results was primarily due to overfitting in the inference network (q). To test this, we trained a separate inference network on the validation and test sets, taking care not to affect the model parameters. This reduced overfitting, but did not completely resolve the issue, suggesting that the generative and inference networks jointly overfit.

6.5. Implementation details

We used Adam (Kingma & Ba, 2014) with a constant learning rate from $\{3\times10^{-5}, 1\times10^{-4}, 3\times10^{-4}, 10^{-3}, 3\times10^{-3}\}$ for the linear models and from $\{3 \times 10^{-5}, 10^{-4}\}$ for the nonlinear models and decays $\beta_1 = 0.9, \beta_2 = 0.99999$. Higher learning rates caused training to diverge. We used minibatches of 24 elements and optimized for 2 million steps. We centered the input to the inference network with the training data statistics. As in (Maddison et al., 2016), all binary variables took values in $\{-1, 1\}$. We initialized the bias of the output layer to the training data statistics as in (Burda et al., 2015). All of the unbiased estimators used input-dependent baselines as described in (Mnih & Gregor, 2014). We used a 10 times faster learning rate for the parameters of the baselines and control variate scalings.

Preliminary evaluations of the REBAR and Concrete estimators over a range of λ found $\lambda = 0.1$ to perform well across tasks and configurations.

6.5.1. MuProp

We found that the linear term in the MuProp baseline was detrimental for later layers, so we learned an additional scaling factor to modulate the linear terms. This reduced the variance of the MuProp learning signal beyond the algorithm described in (Gu et al., 2015).

6.5.2. REBAR

We learned separate control variate scalings (η) for each parameter group (e.g., the weights in the first layer, the biases in first layer, etc.).

When computing the REBAR estimator, we leverage common random numbers to sample from b, z, and $z|b$. Recall that $z = g(u, \theta)$ where u is a uniform random variable, $b = H(z)$, and $z|b = \tilde{g}(v, b, \theta)$. So, if we sample u uniformly and then generate z and b, z will be distributed according to $z|b$. We can also sample from $z|b$ by noting the point u' where $g(u', \theta) = 0$ and sampling v uniformly and then scaling it to the interval $[u', 1]$ if $b = 1$ or $[0, u']$ otherwise. So a choice of u will determine a corresponding choice of v which produces the same z. We propose using this pair (u, v) as the random numbers in the reparameterization trick.

6.5.3. Concrete

A subtle point addressed in (Maddison et al., 2016) is that the objective optimized by the method we called Concrete and in Jang et al. (2016) is not a stochastic lower bound on the marginal log likelihood. The results reported in (Maddison et al., 2016; Jang et al., 2016) were similar and REBAR is most similar to (Jang et al., 2016), so we chose to compare against it. However, although the Concrete method does not optimize a lower bound, we emphasize that we evaluated a proper stochastic lower bound for all plots and numbers reported (including on the training set).