REBAR: LOW-VARIANCE, UNBIASED GRADIENT ESTIMATES FOR DISCRETE LATENT VARIABLE MODELS

George Tucker[1,*], Andriy Mnih[2], Chris J. Maddison[2,3], Jascha Sohl-Dickstein[1]
[1]Google Brain, [2]DeepMind, [3]University of Oxford
{gjt, amnih, jaschasd}@google.com
cmaddis@stats.ox.ac.uk

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 is challenging due to high-variance gradient estimates. Most approaches to reduce this variance have focused on 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.

We introduce a simple control variate based on the difference between the REINFORCE gradient estimator for the relaxed model and the gradient from the reparameterization trick. This reduces variance, but does not outperform state-of-the-art methods on its own. Our key contribution is to show that it is possible to conditionally marginalize the control variate to significantly improve its effectiveness. We call this the REBAR gradient estimator, because it combines REINFORCE gradients with gradients of the Concrete relaxation.

In our experiments, we illustrate the perils of biased gradient estimators with a toy problem. Then, we use REBAR to train sigmoid belief networks (SBNs) on MNIST and show state-of-the-art variance reduction. Although we focus on binary variables for simplicity, this work is equally applicable to categorical variables.

2 BACKGROUND

For clarity, we consider a simplified scenario in the main text with full details in the Appendix. Let $b \sim \text{Bernoulli}\,(\theta)$ be a vector of independent binary random variables parameterized by θ. We wish to maximize

$$\mathop{\mathbb{E}}_{p(b)}\left[f(b,\theta)\right],$$

where $f(b,\theta)$ is differentiable, 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.

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

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

The first term can be estimated effectively with a single Monte Carlo sample, however, the second term has high variance. Paisley et al. (2012); Ranganath et al. (2014); Mnih & Gregor (2014);

*Work done as part of the Google Brain Residency Program.

Gu et al. (2015) show that carefully designed control variates can reduce the variance significantly. Because the dependence of $f(b, \theta)$ on θ is straightforward to account for, without loss of generality, we can assume that $f(b, \theta) = f(b)$ does not depend on θ.

Maddison et al. (2016) made the observation that $b = H(z)$, where H is the hard threshold function[1] and z is a Logistic random variable defined by

$$z \coloneqq g(u, \theta) \coloneqq \log(\theta/(1-\theta)) + \log(u) - \log(1-u),$$

where $u \sim \mathrm{Uniform}(0, 1)$. Notably, z is differentiably reparameterizable (Kingma & Welling, 2013; Rezende et al., 2014). The obstruction to using the reparameterization trick is the discontinuous hard threshold function. Replacing all occurrences of the hard threshold function with a continuous relaxation $H(z) \approx \sigma_\lambda(z) \coloneqq \sigma\left(\frac{z}{\lambda}\right)$ results in a reparameterizable computational graph. Thus, we can compute low-variance gradient estimates for the relaxed model that approximate the gradient for the discrete model. See Jang et al. (2016); Maddison et al. (2016) for the generalization to the categorical case.

3 REBAR

We seek a low-variance, unbiased gradient estimator. Inspired by the Concrete relaxation, our strategy will be to construct a control variate (see Appendix 6.1 for a review of control variates in this context) based on the difference between the REINFORCE gradient estimator for the relaxed model and the gradient estimator from the reparameterization trick. First, note that

$$\mathop{\mathbb{E}}_{p(b)}\left[f(b)\frac{\partial}{\partial\theta}\log p(b)\right] = \frac{\partial}{\partial\theta}\mathop{\mathbb{E}}_{p(b)}\left[f(b)\right] = \frac{\partial}{\partial\theta}\mathop{\mathbb{E}}_{p(z)}\left[f(H(z))\right] = \mathop{\mathbb{E}}_{p(z)}\left[f(H(z))\frac{\partial}{\partial\theta}\log p(z)\right]. \quad (1)$$

The similar form of the REINFORCE gradient estimator for the relaxed model

$$\frac{\partial}{\partial\theta}\mathop{\mathbb{E}}_{p(z)}\left[f(\sigma_\lambda(z))\right] = \mathop{\mathbb{E}}_{p(z)}\left[f(\sigma_\lambda(z))\frac{\partial}{\partial\theta}\log p(z)\right] \quad (2)$$

suggests it will be an effective control variate. Unfortunately, the Monte Carlo gradient estimator derived from the left hand side of Eq. 1 has much lower variance than the Monte Carlo gradient estimator derived from the right hand side. This is because the left hand side can be seen as analytically performing a conditional marginalization, which is noisily approximated by Monte Carlo samples on the right hand side (see Appendix 6.2 for details). Our key insight is that a similar manipulation can be done for the control variate (Eq. 2),

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

where the first term on the right-hand side can be efficiently estimated with the reparameterization trick (see Appendix 6.2 for the derivation).

Putting the terms together, we arrive at

$$\frac{\partial}{\partial\theta}\mathop{\mathbb{E}}_{p(b)}\left[f(b)\right] = \mathop{\mathbb{E}}_{p(u,v)}\Bigg[\left[f(H(z)) - \eta f(\sigma_\lambda(\tilde{z}))\right]\frac{\partial}{\partial\theta}\log p_b(H(z)) + \eta\frac{\partial}{\partial\theta}f(\sigma_\lambda(z)) - \eta\frac{\partial}{\partial\theta}f(\sigma_\lambda(\tilde{z}))\Bigg], \quad (3)$$

where $u, v \sim \mathrm{Uniform}(0, 1)$, $z \equiv g(u, \theta)$, $\tilde{z} \equiv g'(v, H(z), \theta)$ is the differentiable reparameterization for $z|b$ (Appendix 6.2), and η is a scaling on the control variate. The REBAR estimator is the single sample Monte Carlo estimator of this expectation. To reduce computation, we can tie u and v together using common random numbers (Appendix 6.4). We estimate η by tracking empirical moments with exponential moving averages and minimizing the variance of the Monte Carlo estimator (see (Paisley et al., 2012; Ranganath et al., 2014) and Appendix 6.1 for details). Similarly, λ could be optimized to minimize the variance of the estimator. We leave this to future work.

Finally, we note that the derivation of this control variate is independent of f, thus the REBAR control variate can be generalized by replacing f with a learnable, differentiable Q-function. This suggests that the REBAR control variate is applicable to RL, where it would allow a "pseudo" action-dependent baseline. In this case, the pseudo-action would be the relaxation of the discrete output from a policy network.

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

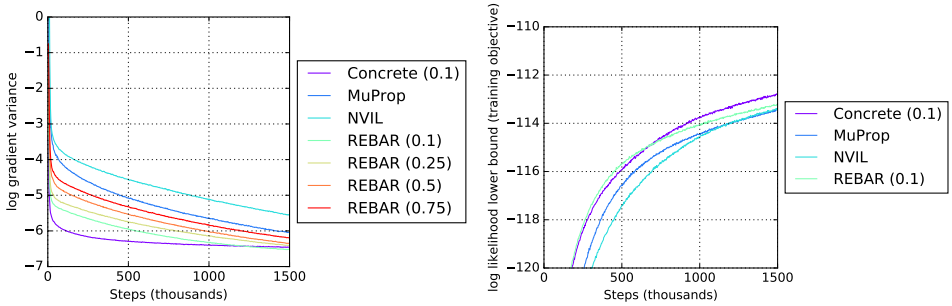

Figure 1: Log gradient estimator variance (left) and training variational lower bound on a single layer SBN model. λ shown in parenthesis when applicable. We evaluated the REBAR and Concrete estimators over a range of λ. We show the choice resulting in the lowest variance for REBAR and the best performance for Concrete ($\lambda = 0.1$ in both cases). More detailed curves in Appendix Figure 3. All of the estimators, except the Concrete estimator, are unbiased, so the variance of their gradient estimates is directly comparable. We compute the mean parameter gradient estimate variance by estimating moments from exponential moving averages (decay=0.999).

4 Experiments

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).

As a preliminary exploration, we tested the methods on a simple toy problem. Suppose we wish to minimize $\mathbb{E}_{p(b)}[(b-t)^2]$, where $t \in (0, 1)$ is a continuous target value. Appendix Figure 2 shows the perils of biased gradient estimators. All of the unbiased estimators correctly converge to the optimal loss, whereas the biased estimator fails to.

Next, we train generative SBNs to model MNIST digits, a standard benchmark task. We follow the setup established in (Maddison et al., 2016). Briefly, we used the statically binarized MNIST digits from Salakhutdinov. The generative network uses several layers of stochastic binary units interleaved with deterministic nonlinearities. In our experiments, we used a single layer of stochastic binary units and either a single linear deterministic layer or 2 layers of 200 tanh units. We maximize the variational lower bound on the log likelihood and perform amortized inference with an inference network with similar architecture. In particular, denoting the image by x and the hidden layer random activations by $h \sim q(h|x, \theta)$, we have

$$\log p(x|\theta) \geq \mathop{\mathbb{E}}_{q(h|x,\theta)} \left[\log p(x, h|\theta) - \log q(h|x, \theta)\right],$$

which has the required form for REBAR. REBAR produces much lower variance gradient estimates than the other unbiased methods (Figure 1), which translates into faster optimization of the objective and convergence to a better final value. For the linear case, the Concrete estimator outperforms the unbiased estimators. However, for the nonlinear case, the Concrete estimator severely underperforms optimizing the training objective (Figure 4). These results do not directly translate into improved test log likelihood lower bounds, likely due to overfitting in the inference network (q). We will explore this issue in future work.

5 Conclusion

Inspired by the Concrete relaxation, we introduced a novel control variate for REINFORCE and preliminary experiments show that it greatly reduces the variance of the gradient estimator. It would be natural to explore the extension to the multisample 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.

REFERENCES

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

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

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

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

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

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

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

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

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

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*, pp. 1278–1286, 2014.

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

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

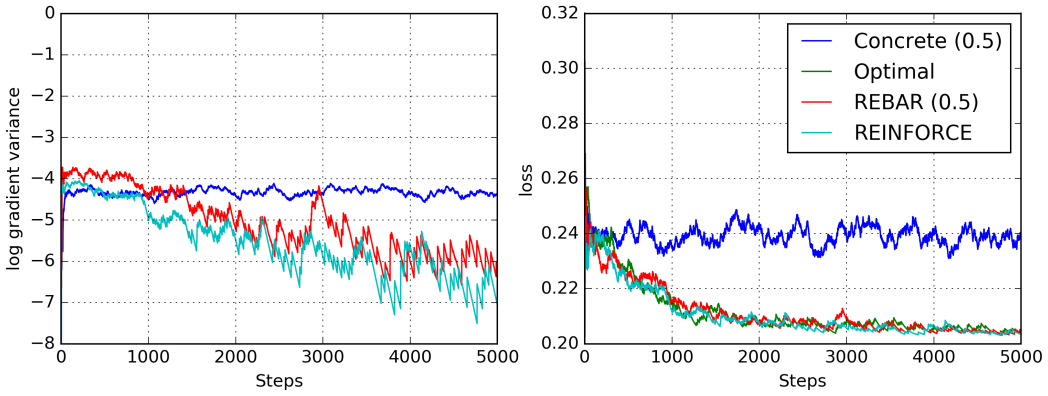

Figure 2: Log gradient variance and loss for the toy problem. $t = 0.45$. Only the unbiased estimators converge to the correct answer.

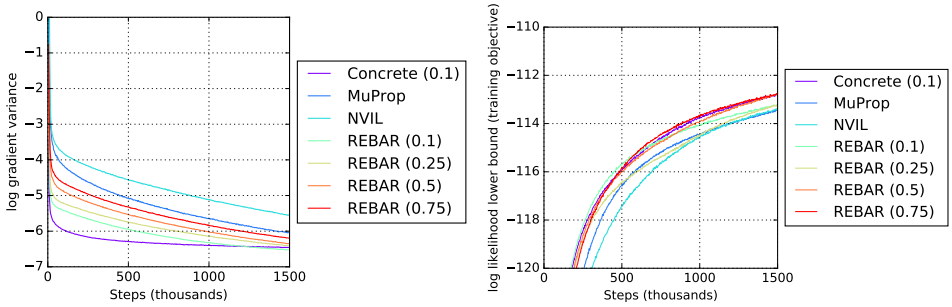

Figure 3: Log gradient estimator variance (left) and training variational lower bound on a single layer SBN model. λ shown in parenthesis when applicable. We evaluated the REBAR and Concrete estimators over a range of λ. Only the best training curve for the Concrete estimator is shown. All of the estimators, except the Concrete estimator, are unbiased, so the variance of their gradient estimates is directly comparable. We compute the mean parameter gradient estimate variance by estimating moments from exponential moving averages (decay=0.999).

ACKNOWLEDGEMENTS

We thank Dieterich Lawson for many insightful discussions. We also thank Ben Poole and Eric Jang for helpful discussions and assistance replicating their results.

6 APPENDIX

6.1 CONTROL VARIATES

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

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

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

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

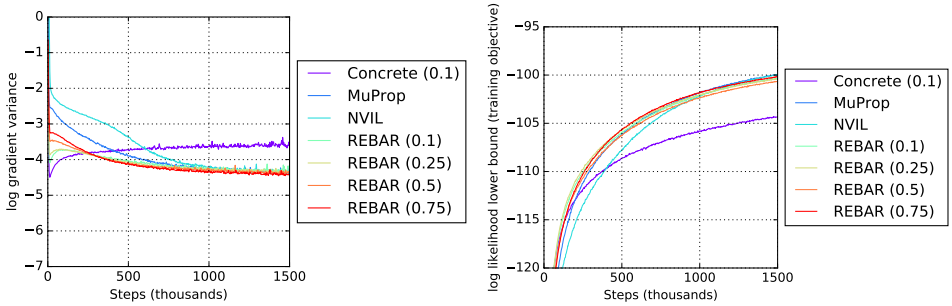

Figure 4: Log gradient estimator variance (left) and training variational lower bound on a single layer nonlinear SBN model. λ shown in parenthesis when applicable. We evaluated the REBAR and Concrete estimators over a range of λ. Only the best training curve for the Concrete estimator is shown. All of the estimators, except the Concrete estimator, are unbiased, so the variance of their gradient estimates is directly comparable. We compute the mean parameter gradient estimate variance by estimating moments from exponential moving averages (decay=0.999).

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 $E[g]$, 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.

In our experiments, we estimate the moments required to compute η using exponential moving averages with decay 0.999.

6.2 Conditional marginalization for the control variate

In this section, we explain why the Monte Carlo gradient estimator derived from the left hand side of Eq. 1 has much lower variance than the Monte Carlo gradient estimator derived from the right hand side. This is because the left hand side can be seen as analytically performing a conditional marginalization

$$\begin{aligned}
\mathop{\mathbb{E}}_{p(z)}\left[f(H(z))\frac{\partial}{\partial\theta}\log p(z)\right] &= \mathop{\mathbb{E}}_{p(b)}\left[\mathop{\mathbb{E}}_{p(z|b)}\left[f(H(z))\frac{\partial}{\partial\theta}\log p(z)\right]\right] = \mathop{\mathbb{E}}_{p(b)}\left[f(b)\mathop{\mathbb{E}}_{p(z|b)}\left[\frac{\partial}{\partial\theta}\log p(z)\right]\right] \\
&= \mathop{\mathbb{E}}_{p(b)}\left[f(b)\mathop{\mathbb{E}}_{p(z|b)}\left[\frac{\partial}{\partial\theta}(\log p(z|b)+\log p(b))\right]\right] \\
&= \mathop{\mathbb{E}}_{p(b)}\left[f(b)\frac{\partial}{\partial\theta}\log p(b)\right],
\end{aligned}$$

where the equality in the second line follows from the fact that when $b = H(z)$, $p(z) = p(z|b)p(b)$.

A similar manipulation holds for the control variate

$$\begin{aligned}
\mathop{\mathbb{E}}_{p(z)}\left[f(\sigma_\lambda(z))\frac{\partial}{\partial\theta}\log p(z)\right] &= \mathop{\mathbb{E}}_{p(b)}\left[\mathop{\mathbb{E}}_{p(z|b)}\left[f(\sigma_\lambda(z))\frac{\partial}{\partial\theta}\log p(z)\right]\right] \\
&= \mathop{\mathbb{E}}_{p(b)}\left[\mathop{\mathbb{E}}_{p(z|b)}\left[f(\sigma_\lambda(z))\frac{\partial}{\partial\theta}(\log p(z|b)+\log p(b))\right]\right] \\
&= \mathop{\mathbb{E}}_{p(b)}\left[\frac{\partial}{\partial\theta}\mathop{\mathbb{E}}_{p(z|b)}\left[f(\sigma_\lambda(z))\right]\right] + \mathop{\mathbb{E}}_{p(b)}\left[\mathop{\mathbb{E}}_{p(z|b)}\left[f(\sigma_\lambda(z))\right]\frac{\partial}{\partial\theta}\log p(b)\right]
\end{aligned}$$

where again we used the fact that when $b = H(z)$, $p(z) = p(z|b)p(b)$.

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 $v \sim \text{Uniform}(0,1)$ 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 $g'(v,b,\theta)$. Note that g' is also differentiable.

6.3 MULTILAYER STOCHASTIC NETWORK

Suppose we have multiple layers of stochastic units (i.e., $b = \{b_1, b_2, \ldots, b_n\}$). In this case, $p(b)$ 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 $p(z_{1:n})$. We can define a relaxed distribution over $z_{1:n}$ where we replace the hard threshold function $H(z)$ with a continuous relaxation $\sigma_\lambda(z)$. This induces a distribution over $b_{1:n}$ where $b_i = H(z_i)$. We refer to these relaxed distributions as $q(z_{1:n})$ and $q(b_{1:n})$.

The high variance REINFORCE term of the gradient also decomposes

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

Focusing on a single term, we have

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

which suggests the following control variate

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

for the middle expectation. However, this requires n passes through the network (noting the switch between sampling from p to q). We propose two approaches for reducing the computational complexity.

We can follow a single trajectory from q by considering a related gradient

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

and control variate

$$\mathop{\mathbb{E}}_{q(z_i|b_i,\sigma_\lambda(z_{i-1}))}\left[\mathop{\mathbb{E}}_{q(z_{i+1:n}|z_i)}[f(\sigma_\lambda(z))]\right]\frac{\partial}{\partial\theta}\log q(b_i|\sigma_\lambda(z_{i-1})), \tag{4}$$

where as in the main text, we can use the reparameterization trick to estimate the required expectations to debias the control variate. Interestingly, this can be seen as a control variate for the original gradient by reparameterizing the p and q distributions by uniform randomness. Next, we can estimate the outer expectation over $p(b_{1:i-1})$ by importance sampling. In particular,

$$\begin{aligned}
&\mathop{\mathbb{E}}_{p(b_{1:i-1})}\left[\mathop{\mathbb{E}}_{p(b_i|b_{i-1})}\left[\mathop{\mathbb{E}}_{p(b_{i+1:n}|b_i)}[f(b)]\frac{\partial}{\partial\theta}\log p(b_i|b_{i-1})\right]\right] \\
&= \mathop{\mathbb{E}}_{p(z_{1:i-1})}\left[\mathop{\mathbb{E}}_{p(b_i|H(z_{i-1}))}\left[\mathop{\mathbb{E}}_{p(b_{i+1:n}|b_i)}[f(b)]\frac{\partial}{\partial\theta}\log p(b_i|b_{i-1})\right]\right] \\
&= \mathop{\mathbb{E}}_{q(z_{1:i-1})}\left[\frac{p(z_{1:i-1})}{q(z_{1:i-1})}\mathop{\mathbb{E}}_{p(b_i|H(z_{i-1}))}\left[\mathop{\mathbb{E}}_{p(b_{i+1:n}|b_i)}[f(b)]\frac{\partial}{\partial\theta}\log p(b_i|b_{i-1})\right]\right].
\end{aligned}$$

Now, we can use Eq. 4 as the control variate for the middle expectation. Notably, we can use a single trajectory from q to compute all of the terms (as i ranges from 1 to n).

Alternatively, we can use the control variate

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

where we train Q to minimize[2]

$$\mathop{\mathbb{E}}_{p(b_{1:i})}\left[\left(\mathop{\mathbb{E}}_{p(b_{i+1:n}|b_i)}[f(b)] - \mathop{\mathbb{E}}_{p(z_i|b_i)}[Q(\sigma_\lambda(z_i),\theta,b_{1:i-1})]\right)^2\right].$$

This avoids the extra computation, and the Q-function could potentially learn to compensate for the continuous relaxation.

[2]Note that this ignores the variance contribution from the reparameterizable terms. We leave evaluating this approach to future work.

6.4 IMPLEMENTATION DETAILS

We used Adam with a constant learning rate in 0.0003. We centered the input to the inference network. All binary variables took values in $\{-1, 1\}$, which we found to be better than $\{0, 1\}$. All of the unbiased estimators used input dependent baselines as described in (Mnih & Gregor, 2014).

6.4.1 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.

6.4.2 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 $z|b = g'(v, b, \theta)$. 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.