BACKPROPAGATING THROUGH ANYTHING BY OPTIMIZING DIFFERENTIABLE CONTROL VARIATES

Anonymous authors
Paper under double-blind review

ABSTRACT

Gradient-based optimization is the foundation of deep learning and reinforcement learning. Even when the mechanism being optimized is unknown or not differentiable, optimization using high-variance or biased gradient estimates is still often the best strategy. We introduce a general framework for learning low-variance, unbiased gradient estimators for black-box functions of random variables. These estimators can be jointly trained with model parameters or policies, and are applicable in both discrete and continuous settings. We give unbiased, adaptive analogs of state-of-the-art reinforcement learning methods such as deep deterministic policy gradients and advantage actor-critic. We also demonstrate this framework for training discrete latent-variable models.

1 INTRODUCTION

Gradient-based optimization has been key to most recent advances in machine learning and deep reinforcement learning. The back-propagation algorithm (**?**), also known as reverse-mode automatic differentiation (**??**) computes exact gradients of deterministic, differentiable objective functions. The reparameterization trick (**???**) allows backpropagation to give unbiased, low-variance estimates of gradients of expectations of continuous random variables. This has allowed effective stochastic optimization of large probabilistic latent-variable models.

Unfortunately, backpropagation cannot be straightforwardly applied to problems involving discrete random variables, or when the function being optimized is a black box (**?**). This is the case in most reinforcement learning settings, or when fitting probabilistic models with discrete latent variables. Much recent work has been devoted to constructing gradient estimators for these situations. In reinforcement learning, advantage actor-critic methods (**?**) and deep deterministic policy gradients (**?**) give low-variance but biased(MAY NEED TO CLARIFY THIS, a2s is not biased) gradient estimates by jointly optimizing policy parameters together with baselines. In discrete latent-variable models, low-variance but biased gradient estimates can be given by continuous relaxations of discrete variables (**??**).

A recent advance by **?** used low-variance but biased gradients from continuous relaxations to construct unbiased, low-variance gradient estimates. Furthermore, **?** showed how to tune the free parameters of these relaxations to minimize their variance during training.

We generalize the method of **?** to learn a free-form control variate parameterized by a neural network, giving lower-variance, unbiased gradient estimates. Importantly, our method is applicable even when no continuous relaxation is available, as in reinforcement learning. We derive improved variants of popular reinforcement learning methods with unbiased gradients and more stable training dynamics.

2 BACKGROUND: GRADIENT ESTIMATORS

How to choose the parameters of a distribution to maximize an expectation? This problem comes up in reinforcement learning, where we must choose a policy π on actions a to maximize the expected reward $\mathbb{E}_{p(a|\pi)}\left[r(a)\right]$. It also comes up in fitting latent-variable models, when we wish to maximize the marginal probability $p(x|\theta) = \sum p(x|z)p(z|\theta) = \mathbb{E}_{p(z|\theta)}\left[p(x|z)\right]$. In this paper, we'll consider

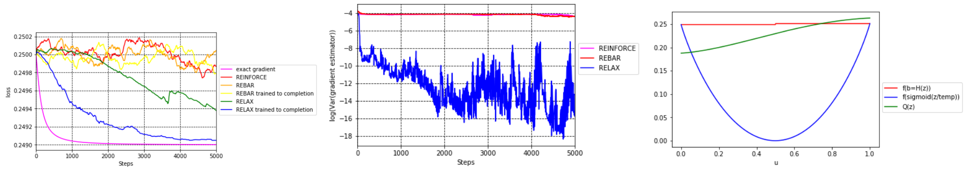

Figure 1: *Left:* Estimator losses. *Centre:* Estimator variance. *Right:*Learned relaxation function.

the general problem of optimizing

$$\mathcal{L}(\theta) = \mathbb{E}_{p_\theta(b)}[\, f(b)] \,. \tag{1}$$

Later, we will discuss the case when $f(b)$ depends directly on θ.

When the parameters θ are high-dimensional, gradient-based optimization is a appealing because it provides information about how to adjust each parameter individually. Stochastic optimization is essential for large problems, but is only guaranteed to converge to a fixed point of the original objective (**?**) when the stochastic gradients $\hat{g}$ are unbiased, i.e. $\mathbb{E}\left[\hat{g}\right] = \nabla_\theta \mathbb{E}_{p(b|\theta)}\left[f(b)\right]$.

The score-function gradient estimator One of the most generally-applicable gradient estimators is known as the score-function estimaor, or REINFORCE (**?**):

$$\hat{g}_{\text{reinforce}} = f\left(b\right) \nabla_\theta \log p(b|\theta), \qquad b \sim p(b|\theta) \tag{2}$$

This estimator is unbiased [todo: give conditions], but in general has high variance (CAN ME MAKE THAT CLAIM?). Intuitively, this estimator is limited by the fact that it doesn't use any information about how f depends on b, only on the final outcome of evaluating $f(b)$.

The reparameterization trick When f is continuous and differentiable, and the latent variables b can be written as a deterministic, differentiable function of a random draw from a fixed distribution, the reparameterization trick (**???**) creates a low-variance, unbiased gradient estimator by making the dependence of b on θ explicit:

$$\hat{g}_{\text{reparam}} = \nabla_\theta f\left(b(\theta, \epsilon)\right), \qquad \epsilon \sim p(\epsilon) \tag{3}$$

This gradient estimator is used when training high-dimensional, continuous latent-variable models, such as variational autoencoders or GANs. One intuition for why this gradient estimator is preferable to REINFORCE is that it depends on $\partial f / \partial b$, which directly exposes the dependence of f on b.

Concrete relaxation When b is discrete and f is known, one general approach is to differentiate a continuous relaxation of the discrete random variables. **?** and **?** developed a differentiable relaxation of the categorical distribution, called the concrete distribution:

$$\hat{g}_{\text{concrete}} = \nabla_\theta f\left(\sigma(\log\theta - \log(-\log(\mathbf{u})))\right), \qquad \mathbf{u} \sim \text{uniform}[0,1] \tag{4}$$

where σ_λ is the softmax function with temperature λ. This gradient estimator is fairly effective in practice, but produces biased gradients. Additionally, it is not clear how to set the temperature λ.

Control variates Control variates are a general trick for reducing the variance of a Monte Carlo estimator. Given an estimator $f(b)$, a control variate is a function $\hat{f}(b)$ with a known mean $\mathbb{E}_{p(b)}\left[\hat{f}\right]$. Subtracting the control variate from our estimator and adding its mean gives us a new estimator:

$$\hat{f}_{\text{new}}(b) = f(b) - \hat{f}(b) + \mathbb{E}_{p(b)}[\hat{f}(b)] \tag{5}$$

This new estimator has the same expectation as the old one:

$$\mathbb{E}_{p(b)}\left[f_{\text{new}}\right] = \mathbb{E}_{p(b)}\left[f(b) - \hat{f}(b) + \mathbb{E}_{p(b)}\left[\hat{f}(b)\right]\right] = \mathbb{E}_{p(b)}\left[f(b)\right] \tag{6}$$

Importantly, the new estimator has lower variance than $f(b)$ if $\hat{f}(b)$ is positively correlated with $f(b)$.

2.0.1 Reducing gradient variance through control variates

$$\frac{\partial}{\partial \theta}\mathbb{E}_{p(z)}[f(\sigma_\lambda(z))] = \mathbb{E}_{p(z)}[f(\sigma_\lambda(z))\frac{\partial}{\partial \theta}\log p(z)]. \tag{7}$$

This introduces a second reparameterization $\tilde{z}$ of $p(z|b)$, which depends on another sample $v \sim \text{Unif}[0,1]$.

The control variate has the following form:

$$f(\sigma_\lambda(\tilde{z}))\frac{\partial}{\partial \theta}\log p(H(z)) \tag{8}$$

and noting that

$$\mathbb{E}_{p(u,v)}[f(\sigma_\lambda(\tilde{z}))\frac{\partial}{\partial \theta}\log p(H(z))] = \mathbb{E}_{p(u,v)}[\frac{\partial}{\partial \theta}f(\sigma_\lambda(\tilde{z})) - \frac{\partial}{\partial \theta}f(\sigma_\lambda(z))] \tag{9}$$

gives us the REBAR gradient estimator:

$$\frac{\partial}{\partial \theta}\mathbb{E}_{p(b)}[f(b)] = \mathbb{E}_{p(u,v)}[f(\sigma_\lambda(\tilde{z}))\frac{\partial}{\partial \theta}\log p(H(z)) - \eta f(\sigma_\lambda(\tilde{z}))\frac{\partial}{\partial \theta}\log p(H(z)) + \eta\frac{\partial}{\partial \theta}f(\sigma_\lambda(z)) - \eta\frac{\partial}{\partial \theta}f(\sigma_\lambda(\tilde{z}))] \tag{10}$$

where η is trained to minimize the variance of the estimator.

The special form of $\tilde{z}$ yields a lower-variance gradient estimate because a number of the random variables are conditionally marginalized out of the estimator. Two features of this control variate make it particularly effective: its high correlation with the REINFORCE gradient, and a low-variance, reparameterized form of certain terms in the estimator.

3 A General Family of Gradient Estimators

The REBAR estimator uses a control variate that evaluates the original loss function at relaxed inputs, reparameterized both unconditionally (denoted z, and conditionally, denoted $\tilde{z}$). The central result of this paper is that learning the function in the control variate leads to even better convergence properties. Specifically, we generalize the conditional marginalization and control variate of REBAR to the following form:

$$\mathbb{E}_{p(u,v)}[r(\tilde{z};\phi)\frac{\partial}{\partial \theta}\log p(H(z))] = \mathbb{E}_{p(u,v)}[\frac{\partial}{\partial \theta}r(\tilde{z};\phi) - \frac{\partial}{\partial \theta}r(z;\phi)], \tag{11}$$

where r is a neural network with parameters ϕ.

The generalized REBAR estimator replaces the loss function evaluations in the control variate with an adaptive r function which is trained via gradient decent to minimize the variance of the estimator. As shown in **?**, this can be easily computed. Denoting the RELAX estimator as $r(\phi)$ we obtain:

$$\frac{\partial}{\partial \phi}\mathbb{V}(r(\phi)) = \frac{\partial}{\partial \phi}\mathbb{E}[r(\phi)^2] + \frac{\partial}{\partial \phi}\mathbb{E}[r(\phi)]^2 = \frac{\partial}{\partial \phi}\mathbb{E}[r(\phi)^2] = \mathbb{E}[\frac{\partial}{\partial \phi}r(\phi)^2] \tag{12}$$

Where the second equality comes from the fact that for all ϕ, the RELAX estimator is unbiased and therefore $\frac{\partial}{\partial \phi}\mathbb{E}[r(\phi)]^2 = 0$.

4 Scope and Limitations

One major limitation of the REBAR estimator and the concrete relaxation is that they requires the function being optimized, whose input is only defined at discrete inputs, to also accept continuous inputs, and to be differentiable w.r.t. those inputs. This makes REBAR and the concrete relaxation inapplicable for optimizing black-box functions, as in reinforcement learning settings where the environment is unknown.

Following **?**, the following overview focuses on a single discrete Bernoulli random variable. However, the generalization to categorial variables is straightforward.

5 OPTIMIZING CONTINUOUS BLACK-BOX FUNCTIONS

Deep deterministic policy gradients (**?**)

Also: (**?**)

6 EXPERIMENTS

We demonstrate the effectiveness of our estimator on a number of challenging optimization problems. Following **?** we begin with a simple toy example to illuminate the potential of our method and then continue to the more relevant problems of optimize binary VAE's and reinforcement learning (OR GRAPH STRUCTURE LEARNING).

6.1 TOY EXPERIMENT

We seek to minimize $\mathbb{E}_{b\sim p(b|\theta)}[(b-t)^2]$ as a function of the parameter θ where $p(b|\theta) = \texttt{Bern}(b|\theta)$. **?** set the target $t = .45$. We focus on the more challenging case where $t = .499$. With this setting of the target, REBAR and competing methods suffer from high variance and are unable to discover the optimal solution $\theta = 0$.

Figure **??** plots the learned relaxations for a fixed value of θ. It can be seen that RELAX learns a relaxation whose derivative points in the direction of decreased loss for all values of reparameterization noise u, whereas REBAR's fixed relaxation only does so for values of $u > t$.

6.2 DISCRETE VARIATIONAL AUTOENCODER

As in (**?**), we benchmark the RELAX estimator on the task of training a variational autoencoder (**??**) where all random variables are Bernoulli taking values in $\{-1, 1\}$. As in **?**, we compare training the variational lower-bound across the MNIST and Omniglot (**?**) datasets. As in **?** we test models with 1 and 2 of Bernoulli random variables with linear mappings between them and a model with 1 layer of Bernoulli random variables with non-linear mappings between layers.

We found that due to the complicated structure of the loss function, the RELAX estimator performed worse than REBAR. Instead we add a learned relaxation to REBAR's control variate which we denote relaxed-REBAR. Our estimator takes the form of (**??**) with

$$\bar{r}(z) = r(z) + f(\sigma_\lambda(z))$$

where $r(z)$ is a learned neural network and $f(\sigma_\lambda(z))$ is the Concrete relaxation of REBAR with temperature parameter λ. In all experiments, adding the learned $r(z)$ reduced the variance of the gradients and improved the final results.

	NVIL	MuProp	REBAR	REBAR	RELAX
MNIST			**?**	ours	
Nonlinear	-102.2	-101.5	-101.1	-83.02	**-79.49**
Linear 1 layer	-112.5	-111.7	-111.6	-111.66	**-111.22**
Linear 2 Layer	-99.6	-99.07	-98.8	-98.23	**-98.04**
Omniglot					
Nonlinear	-110.4	-109.58	-108.72	-62.28	**-58.55**
Linear 1 layer	-117.44	-117.09	-116.83	-116.75	**-116.62**
Linear 2 Layer	-109.98	-109.55	-108.99	-108.74	**-108.59**

Table 1: Training variational lower bound after training.

In (**?**), a separate REBAR estimator was used to estimate the gradients of each model parameter (each weight matrix and bias vector). To apply our estimator to this formulation, we would need to learn a separate relaxation for each model parameter. To get around this, we instead place one gradient estimator on each activation that parameterizes a layer of Bernoulli variables. We then back-propagate this gradient estimate to produce a gradient estimate for each model parameter. To provide a fair comparison, we re-implemented REBAR in this way (denoted REBAR-ours in table 6.2). We believe this explains the large difference in performance between our implementation and that of (**?**) for the nonlinear models since there are 3 layers of parameters that all share the same gradient estimator. In the linear models, each layer has its own gradient estimator making our implementation closer to that of (**?**).

6.3 Reinforcement Learning

Reinforcement learning is strong motivating problem for methods similar to ours. In reinforcement learning we seek to optimize the paremters of a policy distribution $\pi(a|s;\phi)$ to maximize the (often discounted) sum of future rewards given that policy $\mathbb{E}_{\pi(\phi)}[\sum_{t=1}^{\infty} r_t]$. We can view the sum of future rewards as a black-box function of our policy's actions $f(a)$. Thus, as before we have reduced the problem to that of estimating $\frac{\partial \mathbb{E}_{\pi(a|\phi)}[f(a)]}{\partial \phi}$ which is the standard policy gradient algorithm **?**.

We test our approach on simple reinforcement learning environments with discrete actions. We use the RELAX estimator and compare with the advantage actor critic algorithm (A2C **?**) as a baseline. In all experiments we utilized the same learning rate for the policy network for RELAX and A2C to so differences in performance depended solely on the control variate used.

To compare these approaches in the most illustrative setting possible we run these algorithms on one environment at a time, running each episode to completion. After completion, we generate the discounted reward for each timestep, treat the episode as a single batch of data, and perform one step of gradient decent. We test our alrgorithm on the Cart-Pole and Lunar-Lander environments from the OpenAI Gym **?**. We run the Cart-Pole and Lunar-Lander environments for 250 and 1000 episodes, respectively and plot reward and the log-variance of the policy gradients in figure X.

6.4 RL Introduction

We seek to compute

$$\frac{\partial \mathbb{E}_\tau[R]}{\partial \theta} = \mathbb{E}\Big[\sum_{t=1}^{T} \frac{\log \pi(a_t|s_t,\theta)}{\partial \theta} \sum_{t'=t}^{T} r_t\Big]$$

but the estimator on the right hand side can have potentially high variance. Instead, we typically compute

$$\frac{\partial \mathbb{E}_\tau[R]}{\partial \theta} = \mathbb{E}_\tau\Big[\sum_{t=1}^{T} \frac{\log \pi(a_t|s_t,\theta)}{\partial \theta} [(\sum_{t'=t}^{T} r_{t'}) - b(s_t)]\Big]$$

This is unbiased because

$$\tag{13}$$

$$\begin{aligned}
E_{a_{1:T},s_{1:T}}\Big[\frac{\log \pi(a_t|s_t,\theta)}{\partial \theta} \cdot b(s_t)\Big] &= E_{a_{1:t-1},s_{1:t}}\Big[E_{a_{t:T},s_{t+1:T}}\Big[\frac{\log \pi(a_t|s_t,\theta)}{\partial \theta} \cdot b(s_t)\Big]\Big] && (14)\\
&= E_{a_{1:t-1},s_{1:t}}\Big[b(s_t) \cdot E_{a_{t:T},s_{t+1:T}}\Big[\frac{\log \pi(a_t|s_t,\theta)}{\partial \theta}\Big]\Big] && (15)\\
&= E_{a_{1:t-1},s_{1:t}}\Big[b(s_t) \cdot \frac{\partial}{\partial \theta} E_{a_{t:T},s_{t+1:T}}[1]\Big] && (16)\\
&= 0 && (17)
\end{aligned}$$

Where $b(s_t)$ is trained to minimize $(b(s_t) - \sum_{i=t}^{T} r_t)^2$. This is unbiased for any choice of b since b does not depend on a_t. We are interested in replacing $b(s_t)$ with a function $m(a_t, s_t)$ which is a function of both actions and states.

This changes the first equation to

$$\frac{\partial \mathbb{E}_\tau[R]}{\partial \theta} = \mathbb{E}_\tau\Big[\sum_{t=1}^{T} \frac{\log \pi(a_t|s_t,\theta)}{\partial \theta} [(\sum_{t'=t}^{T} r_{t'}) - m(a_t,s_t)]\Big]$$

but this makes the estimator biased as $\mathbb{E}_\tau[\frac{\log \pi(a_t|s_t,\theta)}{\partial\theta} m(a_t, s_t)] \neq 0$. Because of the dependence on a_t we cannot pull this out of the expectation as we could in line 2 of the above equation.

Instead we subtract out a term who's expectation should be the same as the score function version of the control variate changing our estimator to

$$\frac{\partial \mathbb{E}_\tau[R]}{\partial\theta} = \mathbb{E}_\tau\Big[\sum_{t=1}^{T} \frac{\log \pi(a_t|s_t,\theta)}{\partial\theta}[(\sum_{t'=t}^{T} r_{t'}) - m(a_t, s_t)] + \frac{\partial m(a_t, s_t)}{\partial\theta}\Big].$$

For this estimator to be unbiased, we must have

$$\mathbb{E}_\tau\Big[\sum_{t=1}^{T} \frac{\log \pi(a_t|s_t,\theta)}{\partial\theta} m(a_t, s_t) - \frac{\partial m(a_t, s_t)}{\partial\theta}\Big] = 0$$

and we claim that $\forall t$, we have

$$\mathbb{E}_\tau\Big[\frac{\log \pi(a_t|s_t,\theta)}{\partial\theta} m(a_t, s_t) - \frac{\partial m(a_t, s_t)}{\partial\theta}\Big] = 0$$

We begin with

$$\begin{aligned}
\mathbb{E}_\tau\Big[\frac{\partial m(a_t, s_t)}{\partial\theta}\Big] &= \mathbb{E}_{a_{1:t-1}, s_{1:t}}\Big[E_{a_{t:T}, s_{t+1:T}}\Big[\frac{\partial m(a_t, s_t)}{\partial\theta}\Big]\Big] && (18)\\
&= \mathbb{E}_{a_{1:t-1}, s_{1:t}}\Big[E_{a_t}\Big[\frac{\partial m(a_t, s_t)}{\partial\theta}\Big]\Big] && (19)\\
&= \mathbb{E}_{a_{1:t-1}, s_{1:t}}\Big[\frac{\partial E_{a_t}[m(a_t, s_t)]}{\partial\theta}\Big] && (20)\\
&= \mathbb{E}_{a_{1:t-1}, s_{1:t}}\Big[E_{a_t}\Big[\frac{\log \pi(a_t|s_t,\theta)}{\partial\theta} m(a_t, s_t)\Big]\Big] && (21)\\
&= E_\tau\Big[\frac{\log \pi(a_t|s_t,\theta)}{\partial\theta} m(a_t, s_t)\Big] && (22)
\end{aligned}$$

which completes our proof.

?

[Do we use ADAM (**?**) for optimization?]

7 Related Work

? further reduce the variance of reparameterization gradients in an orthogonal way.

As gradient estimators become more complex, checking their unbiasedness numerically becomes difficult. The automatic theorem-proving-based unbiasedness checker developed by **?** may become relevant to this line of research.

NVIL (**?**), VIMCO (**?**)

? address the general problem of developing gradient estimators for deterministic black-box functions or discrete optimization. They introduce a sampling distribution, and optimize an objective similar to ours.

? also introduce a sampling distribution to build a gradient estimator, and consider optimizing the sampling distribution.

? reduce the variance of actor-critic gradient estimates by simply summing over all possible actions.

? estimate gradients using a form of finite differences, evaluating hundreds of different parameter values in pararallel to construct a gradient estimator. In contrast, our method is a simple-sample estimator.

Generalized Reparameterization Gradients REBAR and the generalization in this paper uses a mixture of score function and reparameterization gradients. A recent paper by **?** unifies these two gradient estimators as the generalized reparameterization gradient (GRG). This framework can help disentangle the various components of generalized REBAR.

REBAR innovation as further decomposition the correction term into secondary reparameterization components note this is a recursive application of the principles of GRG observe that the GRG suggests this recursive application to components of an estimator propose that other estimators could be similarly recursively decomposed?

8 Conclusions and Future Work

Other possible applications:

GANs (**?**) that generate text or other discrete objects.

Learning to parse (**?**)

VAEs with continuous latent variables but non-differentiable likelihood functions.

Acknowledgements

We thank Tian Qi Chen and Yuhuai Wu for helpful discussions.

9 Appendix A: Control Variates

Generalizing the reparameterization trick

Write sample from distribution $s(\epsilon)$ as $\epsilon = \mathcal{T}^{-1}(\mathbf{z}; \nu)$ for some invertible transform $\mathcal{T}$ with variational parameters ν. write out transformed density example: normal with standard normal s example: inverse CDF of Gaussian with uniform s write out expected gradient under transformation show decomposition of expected gradient into reparameterization and correction terms

10 Appendix B: Categorical variables

Let $G_{1:k} = -\log - \log(U_{i:k})$ be samples from the Gumbel distribution, and learnable parameters $(\alpha_1, \ldots, \alpha_k)$ be interpreted as some unnormalized parameterization of the discrete distribution under consideration. Then, consider the following sampling procedure: for each k, find the k that maximizes $\log \alpha_k - G_k$, and then set $D_k = 1$ and $D_{i \neq k} = 0$. The Gumbel-Max trick states that sampling from the discrete distribution is equivalent to taking this argmax, that is, $p(D_k = 1) = \alpha_k / \sum_{i=1}^{n} \alpha_i$.

Since taking an argmax is still a discontinuous operation, **?** and **?** proposed further relaxing the argmax operator through the softmax function with an additional temperature parameter λ:

$$x_k = \frac{\exp\{(\log \alpha_k + G_k)/\lambda\}}{\sum_{i=1}^{n} \exp\{(\log \alpha_i + G_i)/\lambda\}} \tag{23}$$

This relaxation allows values within the simplex, but in the low temperature limit, it becomes exactly the discrete argmax. One limitation of the concrete distribution is that it is a biased estimator except in limiting temperature. In other words, a small amount of bias is present for a non-zero temperature.