BACKPROPAGATING THROUGH THE VOID: OPTIMIZING GRADIENT CONTROL VARIATES FOR BLACK-BOX EXPECTATIONS

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 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 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, there are many objective functions relevant to the machine learning community for which backpropagation cannot be applied. In reinforcement learning, for example, the function being optimized is unknown to the agent and is treated as a black box (**?**). Similarly, when fitting probabilistic models with discrete latent variables, discrete sampling operations create discontinuities giving the objective function zero gradient with respect to its parameters. Much recent work has been devoted to constructing gradient estimators for these situations. In reinforcement learning, advantage actor-critic methods (**?**) give unbiased gradient estimates with reduced variance obtained by jointly optimizing the policy parameters with an estimate of the value function. 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 a continuous relaxation to construct a control variate for functions of discrete random variables. Low-variance estimates of the expectation of the control variate can be computed using the reparameterization trick to produce an unbiased estimator with lower variance than previous methods. Furthermore, **?** showed how to tune the free parameters of these relaxations to minimize their variance during training.

In this work we generalize the method of **?** to learn a free-form control variate parameterized by a neural network, giving a lower-variance, unbiased gradient estimator which can be applied to a wider variety of problems with greater flexibility. Most notably, our method is applicable even when no continuous relaxation is available, as in reinforcement learning or black box function optimization. Furthermore, we derive improved variants of popular reinforcement learning methods with unbiased, action-dependent gradient estimates and lower variance.

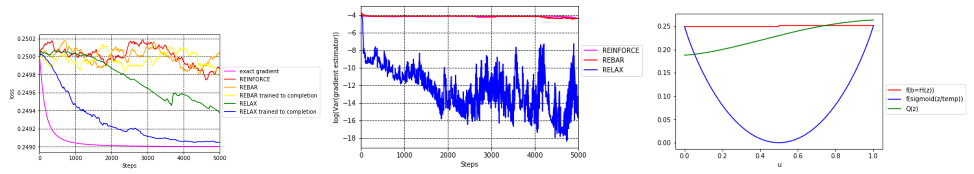

Figure 1: *Left:* Estimator losses. *Centre:* Estimator variance. *Right:*Learned relaxation function.

2 Background: Building gradient estimators

How can we choose the parameters of a distribution to maximize an expectation? This problem comes up in reinforcement learning, where we must choose the parameters θ of a policy distribution $\pi(a|s,\theta)$ to maximize the expected reward $\mathbb{E}_{\tau\sim\pi}\left[R\right]$ over state-action trajectories τ. 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 the general problem of optimizing

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

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 scalablility, but is only guaranteed to converge to a fixed point of the original objective when the stochastic gradients $\hat{g}$ are unbiased, i.e. $\mathbb{E}\left[\hat{g}\right] = \frac{\partial}{\partial\theta}\mathbb{E}_{p(b|\theta)}\left[f(b)\right]$ (**?**).

How can we build unbiased, stochastic estimators of $\frac{\partial}{\partial\theta}\mathcal{L}(\theta)$? There are several standard methods:

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

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

This estimator is unbiased, but in general has high variance. 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 $f(b)$.

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:

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

This gradient estimator is often 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 exposes the dependence of f on b.

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

$$\hat{g}_{\text{new}}(b) = g(b) - \hat{g}(b) + \mathbb{E}_{p(b)}[\hat{g}(b)] \tag{4}$$

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

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

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

3 The LAX estimator

In this section, we introduce a gradient estimator that combines these three strategies to give an estimator can potentially have as low variance the reparameterization-trick estimator, even when the function being optimized is not differentiable.

We first consider the case where b is a continuous random variable.

3.1 Continuous Variables

We wish to compute $\frac{\partial}{\partial\theta}\mathbb{E}_{p(b|\theta)}[f(b)]$ where $p(b|\theta)$ is a distribution with continuous support that admits a reparameterization T such that $T(\theta,\epsilon) = b \sim p(b|\theta)$ where $\epsilon \sim p(\epsilon)$. We assume that f is a black-box or involves operations such that $\frac{\partial}{\partial\theta}f = 0$ almost everywhere or is not computable. Given the reparameerization, we can re-write this objective as an expectation over ϵ as $\frac{\partial}{\partial\theta}\mathbb{E}_{p(\epsilon)}[f(T(\epsilon,\theta)]$.

We introduce a new, differentiable function r_ϕ and note that:

$$\begin{aligned}\frac{\partial}{\partial\theta}\mathbb{E}_{p(b|\theta)}[f(b)] &= \mathbb{E}_{p(b|\theta)}\left[[f(b) - r_\phi(b)]\cdot\frac{\partial}{\partial\theta}\log p(b) + \frac{\partial}{\partial\theta}r_\phi(b)\right] \\ &= \mathbb{E}_{p(\epsilon)}\left[[f(T(\epsilon,\theta)) - r_\phi(T(\epsilon,\theta))]\cdot\frac{\partial}{\partial\theta}\log p(b) + \frac{\partial}{\partial\theta}r(T(\epsilon,\theta))\right]\end{aligned}$$

Which allows us to define our estimator below as:

$$\hat{g}_{\text{LAX}} = (f(b) - r_\phi(b))\frac{\partial}{\partial\theta}\log p(b|\theta) + \frac{\partial}{\partial\theta}r_\phi(b) \qquad b = T(\epsilon,\theta), \epsilon \sim p(\epsilon). \tag{6}$$

We note that our estimator is unbiased for all choices of r_ϕ, i.e $\mathbb{E}_\epsilon[g_\phi] = \frac{\partial}{\partial\theta}\mathbb{E}_{p(b|\theta)}[f(b)]$.

3.2 Optimizing the gradient control variate with gradients

This estimator is unbiased for any control variate, but how can we tune the control variate to minimize its variance? The answer is that we can use the gradient of the variance of our gradient estimator to tune its parameters ϕ with stochastic gradient-based optimization, at the same time as we optimize the parameters of our model or policy.

We can compute these gradients in at least two ways: We could naïvely differentiate through the empirical variance over some mini-batch. Or, following **?**, we can construct an unbiased, single-sample estimator by using the fact that our gradient estimator is unbiased:

$$\frac{\partial}{\partial\phi}\text{Var}(g_\phi) = \frac{\partial}{\partial\phi}\mathbb{E}[g_\phi^2] - \frac{\partial}{\partial\phi}\mathbb{E}[g_\phi]^2 = \frac{\partial}{\partial\phi}\mathbb{E}[g_\phi^2] = \mathbb{E}\left[\frac{\partial}{\partial\phi}g_\phi^2\right] = 2g\frac{\partial g}{\partial\phi}. \tag{7}$$

From inspection of the square of the estimator in (7) we can see that this loss also encourages $r(b)$ to approximate $f(b)$, but with a weighting based on $\frac{\partial}{\partial\theta}\log p(b)$. Moreover, as $r \to f$ then $\hat{g}_{\text{RELAX}} \to \frac{\partial}{\partial\theta}r$. Thus, optimization in this way encourages a balance between the variance of the reparameterization estimator and the variance of the REINFORCE estimator.

This method of directly minimizing the variance of the gradient estimator stands in contrast to other methods such as [Q-Prop?] which set trained the control variate to minimize the squared error $(f(b) - r(b))^2$.

[Todo: talk about the variance of the gradient estimator of the variance of the gradient estimators?]

[t] $f(\cdot)$, $\log p(b|\theta)$, reparameterized sampler $b = T(\theta,\epsilon)$, neural network $r_\phi(\cdot)$ not converged $\epsilon_i \sim p(\epsilon)$ Sample noise $b_i \leftarrow T(\epsilon_i,\theta)$ Compute input $g_\theta \leftarrow [f(b_i) - r_\phi(b_i)]\,\nabla_\theta\log p + \nabla_\theta r(b_i)$ Estimate gradient $g_phi \leftarrow 2g_\theta\frac{\partial g_\theta}{\partial\phi}$ Estimate gradient of variance of gradient $\theta \leftarrow \theta + \alpha_1 g_theta$ Update parameters $\phi \leftarrow \phi + \alpha_2 g_phi$ Update control variate **return** θ

3.3 Discrete Variables

In the case where $p(b|\theta)$ is a distribution over discrete variables, a few more details arise. We assume there exist a continuous distribution $p(z|\theta)$ and a determanistic mapping H such that $H(z) = b \sim$

$p(b|\theta)$. If $p(b)$ is Bernoulli then $p(z|\theta) = \text{Logistic}(z|\theta)$ and $H(z) = \mathbb{I}(z > 0)$. If b is categorical, then $p(z|\theta) = (z|\theta)$ and $H(z) = (z)$. We also assume that $p(z|\theta)$ is reparameterizable. Moreover, we assume the distribution $p(z|b, \theta)$ is also reparameterizable meaning there exists $\hat{T}$ such that $\hat{T}(\hat{\epsilon}, b, \theta) = \hat{z} \sim p(z|b, \theta)$ (see Appendix for details).

4 Conditional reparameterization

In this section, we incorporate a further refinement of our estimator, based on a technique due to **?**.

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:

$$\hat{g}_{\text{concrete}} = \frac{\partial}{\partial\theta} f\left(\sigma_\lambda(\log\theta - \log(-\log u))\right), \qquad u \sim \text{uniform}[0, 1] \tag{8}$$

where σ_λ is the function with temperature λ. A similar relaxation for the Bernoulli distribution has been developed as well (see Appendix for details) This gradient estimator is fairly effective in practice, but produces biased gradients, hindering its usage. Additionally, it is not clear how to set the temperature λ and it is often treated as a hyperparameter.

REBAR The recently-developed REBAR method (**?**) estimates the gradient of expectations of functions of Bernoulli random variables. This method uses the REINFORCE estimator with a control variate. The control variate is derived from the original loss function evaluated at continuously-relaxed inputs (**??**). The expectation of this control variate is estimated with low variance via the reparameterization trick.

Samples are drawn from $p(b|\theta)$ with a determanistic function of a continuous, reparameterizable random variable z.

$$\begin{aligned} z &:= g(u, \theta) = \log\frac{\theta}{1-\theta} + \log\frac{u}{1-u}, \qquad u \sim \text{uniform}[0, 1] \\ b &:= \mathbb{I}(z > 0) \end{aligned}$$

Thus, we can think of $p(b|\theta)$ as $\int p(b|z, \theta)p(z|\theta)dz$. The REBAR estimator is derived from this realization, noting:

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

where σ_λ is the sigmoid function with temperature λ. The REBAR estimator is the single-sample monte-carlo estimator derived from the last line above. The REBAR estimator is computed as:

$$\begin{aligned} \hat{g}_{\text{REBAR}} = \left(f(b) - f(\sigma_\lambda(z)\right)\frac{\partial}{\partial\theta}\log p(b|\theta) + \frac{\partial}{\partial\theta}f(\sigma_\lambda(z)) - \frac{\partial}{\partial\theta}f(\sigma_\lambda(\tilde{z})) \\ z \sim p(z|\theta), \tilde{z} \sim p(z|b, \theta), b = \mathbb{I}(z > 0) \end{aligned}$$

Details of how to sample from $p(z|b, \theta)$ as well as an extension to categorical random variables can be found in the Appendix. **?** also introduced a way to optimize the temperature parameter λ via gradient decent to minimize the variance of the estimator.

4.1 THE RELAX ESTIMATOR

Introducing the parametric function r_ϕ, we derive our estimator similarly to **?** with

$$\begin{aligned}
\frac{\partial}{\partial\theta}\mathbb{E}_{p(b|\theta)}[f(b)] &= \frac{\partial}{\partial\theta}\mathbb{E}_{p(b|\theta)}\left[f(b)\right] - \mathbb{E}_{p(z|\theta)}\left[r_\phi(z)\right] + \frac{\partial}{\partial\theta}\mathbb{E}_{p(z|\theta)}\left[r_\phi(z)\right] \\
&= \frac{\partial}{\partial\theta}\mathbb{E}_{p(b|\theta)}\left[f(b) - \mathbb{E}_{p(z|b,\theta)}\left[r_\phi(z)\right]\right] + \frac{\partial}{\partial\theta}\mathbb{E}_{p(z|\theta)}\left[r_\phi(z)\right] \\
&= \mathbb{E}_{p(b|\theta)}\left[\left(f(b) - \mathbb{E}_{p(z|b,\theta)}\left[r_\phi(z)\right]\right)\frac{\partial}{\partial\theta}\log p(b|\theta) - \frac{\partial}{\partial\theta}\mathbb{E}_{p(z|b,\theta)}\left[r_\phi(z)\right]\right] \\
&\quad + \frac{\partial}{\partial\theta}\mathbb{E}_{p(z|\theta)}\left[r_\phi(z)\right] \\
&= \mathbb{E}_{p(\epsilon,\hat{\epsilon})}\Big[\left(f(H(T(\epsilon,\theta))) - r_\phi(T(\epsilon,\theta))\right)\frac{\partial}{\partial\theta}\log p(H(T(\epsilon,\theta))|\theta) \\
&\qquad\qquad - \frac{\partial}{\partial\theta}r_\phi(\hat{T}(\hat{\epsilon},H(T(\epsilon,\theta)),\theta))\Big] \\
&\quad + \frac{\partial}{\partial\theta}\mathbb{E}_{p(\epsilon)}\left[r_\phi(T(\epsilon,\theta))\right]
\end{aligned}$$

which allows us to define our estimator as

$$\begin{aligned}
\hat{g}_{\text{RELAX}} = \left(f(b) - r_\phi(\hat{z})\right)\frac{\partial}{\partial\theta}\log p(b|\theta) + \frac{\partial}{\partial\theta}r_\phi(z) - \frac{\partial}{\partial\theta}r_\phi(\tilde{z}) \\
b = H(z), z \sim p(z|\theta), \tilde{z} \sim p(z|b,\theta)
\end{aligned} \tag{9}$$

This estimator is also unbiased i.e $E_{\epsilon,\hat{\epsilon}}[g_\phi] = \frac{\partial}{\partial\theta}\mathbb{E}_{p(b|\theta)}[f(b)]$.

5 SCOPE AND LIMITATIONS

Unfortunately, REBAR and concrete require the function being optimized, whose input is only defined at discrete inputs, to also accept continuous inputs, be differentiable w.r.t. those inputs, and behave predictably with respect to those continuous inputs. While often true, these are strong assumptions to make. Furthermore, REBAR and concrete require that the function being optimized is known. This makes REBAR and the concrete relaxation inapplicable for optimizing black-box functions, as in reinforcement learning settings where the reward is an unknown function of the environment.

In contrast, LAX and RELAX can be used in these settings. LAX and RELAX only require that we can query the function being optimized, and can sample from and differentiate $p(b|\theta)$.

Can RELAX be used to optimize deterministic black-box functions? The answer is yes, with the caveat that one must introduce stochasticity to the inputs. Thus, RELAX is most suitable for problems where one is already optimizing a distribution over inputs, such as in inference or reinforcement learning.

6 APPLICATIONS

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 optimizing binary VAE's and reinforcement learning.

6.1 TOY EXPERIMENT

We seek to minimize $\mathbb{E}_{p(b|\theta)}[(b-t)^2]$ 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 $\theta = 0$.

The fixed Concrete relaxation of REBAR is unable to produce a gradient whose signal outweighs the sample noise and is therefore unable to solve this problem noticeably faster than REINFORCE. 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.

MNIST	NVIL	MuProp	REBAR **?**	REBAR ours	RELAX
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 use our gradient estimator to approximate $g_\phi \approx \frac{\partial}{\partial\theta}\mathbb{E}_{q(b|\theta)}[f(b)]$ where $x \cdot W = \theta$ is the parameters of the Bernoulli latent variables, W is our layer's weight matrix. We then obtain an estimate of $\frac{\partial}{\partial W}\mathbb{E}_{q(b|\theta)}[f(b)] = g_\phi \cdot \frac{\partial\theta}{\partial W}$. We note this gives us unbiased gradients because

$$\mathbb{E}_\epsilon[g_\phi(\epsilon) \cdot \frac{\partial\theta}{\partial W}] = \mathbb{E}_\epsilon[g_\phi(\epsilon)] \cdot \frac{\partial\theta}{\partial W} = \frac{\partial}{\partial\theta}\mathbb{E}_{q(b|\theta)}[f(b)] \cdot \frac{\partial\theta}{\partial W} = \frac{\partial}{\partial W}\mathbb{E}_{q(b|\theta)}[f(b)] \tag{10}$$

To provide a fair comparison, we re-implemented REBAR in this way (denoted REBAR-ours in table 5.2). 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 **?** 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

$$\tag{11}$$

$$E_{a_{1:T},s_{1:T}}\left[\frac{\log \pi(a_t|s_t,\theta)}{\partial \theta} \cdot b(s_t)\right] = E_{a_{1:t-1},s_{1:t}}\left[E_{a_{t:T},s_{t+1:T}}\left[\frac{\log \pi(a_t|s_t,\theta)}{\partial \theta} \cdot b(s_t)\right]\right] \tag{12}$$

$$= E_{a_{1:t-1},s_{1:t}}\left[b(s_t) \cdot E_{a_{t:T},s_{t+1:T}}\left[\frac{\log \pi(a_t|s_t,\theta)}{\partial \theta}\right]\right] \tag{13}$$

$$= E_{a_{1:t-1},s_{1:t}}\left[b(s_t) \cdot \frac{\partial}{\partial \theta} E_{a_{t:T},s_{t+1:T}}[1]\right] \tag{14}$$

$$= 0 \tag{15}$$

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.

6.4.1 Variance Reduction with RELAX

We begin the in the case where actions are continuous and any expectation over an action is replaced with an expectation over reparameterization variables ϵ

$$\mathbb{E}_a[f(a)] = \mathbb{E}_\epsilon[f(a(\epsilon))]$$

. 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

$$\begin{aligned}
\mathbb{E}_\tau\Big[\frac{\partial m(a_t,s_t)}{\partial\theta}\Big] &= \mathbb{E}_{a_{1:t-1},s_{1:t}}\Big[E_{a_{t:T},s_{t+1:T}}\Big[\frac{\partial m(a_t,s_t)}{\partial\theta}\Big]\Big] && (16)\\
&= \mathbb{E}_{a_{1:t-1},s_{1:t}}\Big[E_{a_t}\Big[\frac{\partial m(a_t,s_t)}{\partial\theta}\Big]\Big] && (17)\\
&= \mathbb{E}_{a_{1:t-1},s_{1:t}}\Big[\frac{\partial E_{a_t}[m(a_t,s_t)]}{\partial\theta}\Big] && (18)\\
&= \mathbb{E}_{a_{1:t-1},s_{1:t}}\Big[E_{a_t}\Big[\frac{\log\pi(a_t|s_t,\theta)}{\partial\theta} m(a_t,s_t)\Big]\Big] && (19)\\
&= E_\tau\Big[\frac{\log\pi(a_t|s_t,\theta)}{\partial\theta} m(a_t,s_t)\Big] && (20)
\end{aligned}$$

which completes our proof.

6.4.2 Discrete Case

In the discrete action setting our policy parameterizes a soft-max distribution which we use to sample actions. In the discrete case we define $z_t = f(\pi, u) = \sigma(\log\pi - \log(-\log(u)))$ where $u \sim \text{Unif}[0,1]$, $a_t = \text{argmax}(z_t)$, σ is the soft-max function.

We also define $\tilde{z}_t \sim p(z_t|a_t)$ so if the z_t are gumbel softmax samples, then $\tilde{z}_t$ are gumbel softmax samples constrainted so that the largest value is that of the index of a_t. See appendix for how to efficiently generate samples $\tilde{z}_t$.

In the discrete case, we use $m(\tilde{z}_t, s_t) \cdot \frac{\log\pi(a_t|s_t,\theta)}{\partial\theta}$ as our control variate giving us

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

which is biased, so we must add a term to remove this bias which will be $\frac{\partial m(z_t,s_t)}{\partial\theta} - \frac{\partial m(\tilde{z}_t,s_t)}{\partial\theta}$ making the full estimator

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

For this estimator to be unbiased, we must have

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

and we claim that $\forall t$

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

We introduce some notation that $a(z) = \text{argmax}(z)$

$$0 = \frac{\partial}{\partial\theta}\mathbb{E}_\tau\Big[m(z_t, s_t) - m(z_t, s_t)\Big] = \mathbb{E}_{a_{<t}, s_{\le t}}\Big[\frac{\partial}{\partial\theta}\mathbb{E}_{z_t|s_t}\Big[m(z_t, s_t)\Big] - \frac{\partial}{\partial\theta}\mathbb{E}_{a_t|s_t}\Big[\mathbb{E}_{z_t|a_t, s_t}\Big[m(z_t, s_t)\Big]\Big]\Big] \tag{21}$$

We drop the outer expectation for brevity and continue

$$\frac{\partial}{\partial\theta}\mathbb{E}_{z_t|s_t}\Big[m(z_t, s_t)\Big] - \frac{\partial}{\partial\theta}\mathbb{E}_{a_t|s_t}\Big[\mathbb{E}_{z_t|a_t, s_t}\Big[m(z_t, s_t)\Big]\Big] \tag{22}$$

$$= \frac{\partial}{\partial\theta}\mathbb{E}_{z_t|s_t}\Big[m(z_t, s_t)\Big] - \mathbb{E}_{a_t|s_t}\Big[\mathbb{E}_{z_t|a_t, s_t}\Big[m(z_t, s_t)\frac{\log\pi(a_t|s_t, \theta)}{\partial\theta} - \frac{\partial}{\partial\theta}\mathbb{E}_{z_t|a_t, s_t}\Big[m(z_t, s_t)\Big]\Big] \tag{23}$$

$$= \mathbb{E}_{z_t|s_t}\Big[\frac{\partial}{\partial\theta}m(z_t, s_t)\Big] - \mathbb{E}_{a_t|s_t}\Big[\mathbb{E}_{z_t|a_t, s_t}\Big[m(z_t, s_t)\frac{\log\pi(a_t|s_t, \theta)}{\partial\theta} - \mathbb{E}_{z_t|a_t, s_t}\Big[\frac{\partial}{\partial\theta}m(z_t, s_t)\Big]\Big] \tag{24}$$

Thus due to the markov property of the MDP, then we can wrap all of these expectations into the expectation over trajectories giving us

$$\mathbb{E}_{a_{<t}, s_{\le t}}\Big[\mathbb{E}_{z_t|s_t}\Big[\frac{\partial}{\partial\theta}m(z_t, s_t)\Big] - \mathbb{E}_{a_t|s_t}\Big[\mathbb{E}_{z_t|a_t, s_t}\Big[m(z_t, s_t)\frac{\log\pi(a_t|s_t, \theta)}{\partial\theta} - \mathbb{E}_{z_t|a_t, s_t}\Big[\frac{\partial}{\partial\theta}m(z_t, s_t)\Big]\Big]\Big] \tag{25}$$

$$= \mathbb{E}_\tau\Big[\frac{\log\pi(a_t|s_t, \theta)}{\partial\theta}m(\tilde{z}_t, s_t) - \Big(\frac{\partial m(z_t, s_t)}{\partial\theta} - \frac{\partial m(\tilde{z}_t, s_t)}{\partial\theta}\Big)\Big] = 0 \tag{26}$$

6.4.3 Experiments

We run experiments on some small reinforcement learning tasks with both discrete and continuous actions. We compare our method against a baseline of advantage actor-critic. We do not bootstrap the rewards with the value function. We estimate the policy gradient using a single Monte Carlo sample produced from a one episode roll-out of the current policy. We note that improved performance could be achieved by using more samples, but we intended to test our model in the highest-possible variance setting. In all experiments the model architectures were identical between our model and the baseline and the learning rates were also constant across tests making it that all improvements derive from having a lower variance estimate of the policy gradient.

We test our approach in the discrete action domains of the CartPole and LunarLander problems as provided by the OpenAI gym **?**.

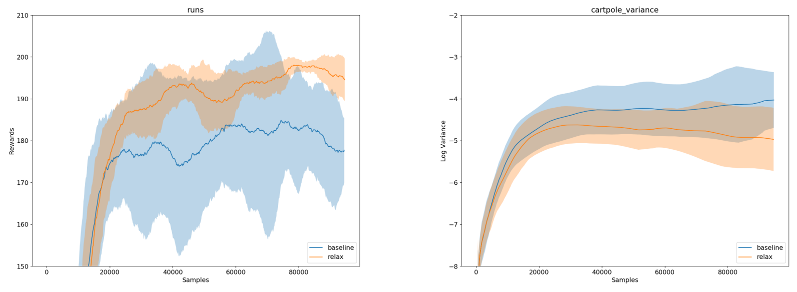

Figure 2: Cartpole Experiments. *Left:* Rewards *Right:* Log Variances.

?

[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

In this work, synthesized and generalized many of the standard approaches for constructing gradient estimators. We proposed a simple and generic gradient estimator that can be applied to expectations of known or black-box functions of discrete or continuous random variables. This approach is relatively simple to implement and adds minimal computational overhead.

The foundation of our approach is the score function gradient estimator with a control variate whose expectation can be estimated with low variance using the reparameterization trick. This control variate is neural network which is trained directly to minimize the variance of the estimated gradients. The central result of this paper is that learning the function in the control variate leads to even better convergence properties and lower variance gradient estimates.

Other possible applications:

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

Learning to parse (**?**)

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

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:

$$x_k = \frac{\exp\{(\log \alpha_k + G_k)/\lambda\}}{\sum_{i=1}^{n} \exp\{(\log \alpha_i + G_i)/\lambda\}} \tag{27}$$

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.