Backpropagation-through-the-Void

Backpropagation through the Void：Optimizing control variates for black-box gradient estimation

- 作者：Will Grathwohl, Dami Choi, Yuhuai Wu, et al.
- 机构：University of Toronto and Vector Institute

- 会议：ICLR 2018
- 地址：https://arxiv.org/abs/1711.00123
- 代码：https://github.com/duvenaud/relax

文章概要

摘要

基于梯度的优化是deep learning、reinforcement learning的基础，但很难应用在不可微分或者未知的机理中。文章引入了一种低variance、unbiased的gradient estimator框架，可适用于包含离散或者连续随机变量的黑盒函数。使用神经网络构造Control variate，网络参数与原始参数联合优化。

贡献

- 在unbiased、low variance的前提下，能处理不可微黑盒函数

研究内容

问题定义

估计关于分布参数 θ 的梯度 $\mathbb{E}_{p(b|\theta)}[f(b)]$.

$$
\begin{aligned}
\partial_\theta \mathbb{E}_{p(b|\theta)}[f(b)] &= \partial_\theta \int_{-\infty}^{\infty} f(b)p(b \mid \theta)db \\
&= \int_{-\infty}^{\infty} f(b)\partial_\theta p(b \mid \theta)db
\end{aligned}
$$

想用简单的Mote Carlo估计，得有个数学期望的表达式。

方法：

1. score-function: $\partial_\theta \mathbb{E}p(b \mid \theta)[f(b)] = \mathbb{E}p(b \mid \theta)\left[f(b)\partial_\theta \log(p(b \mid \theta))\right]$
2. reparameterization: $\hat{g}_{\text{reparam}}[f] = \frac{\partial}{\partial\theta} f(b) = \frac{\partial f}{\partial T}\frac{\partial T}{\partial \theta}, \quad \epsilon \sim p(\epsilon)\ ,\ b = T(\theta, \epsilon)$

Goal

1. 可以处理不可微的 f
2. 可以处理离散随机变量的 b
3. unbiased
4. low variance

方法：Control variate

由

$$\mathbb{E}p(b \mid \theta)\left[C(b)\partial_\theta \log(p(b \mid \theta))\right] = \partial\theta\mathbb{E}_{p(b|\theta)}[C(b)]$$

减一项加一项（保证了**unbiased**）：

$$\hat{g} = \mathbb{E}p(b \mid \theta)\left[(f(b) - C(b))\partial_\theta \log(p(b \mid \theta))\right] + \partial\theta\mathbb{E}_{p(b|\theta)}[C(b)]$$

- 等式右边第二项可以用Reparameterization来求（ C 已知）
- 当 $C = f$ ➡ Reparameterization（low variance，但需要 f 可微）
- 当 C 为常数 ➡ Score-function（high variance）
- 当 $C \neq f$ ➡ 不需要 f 可微，并且variance不会大于SF

1. 找一个**可微**函数 h 来分解随机变量 b .
2. 用估计的梯度更新 θ
3. 更新 C 来最小化Variance

方法：Concrete (Gumbel-softmax trick)

1. 采样 $u \sim \text{Unif}(0, 1)$
2. 定义 $z = h(u, \theta)$. 考虑 b 是二项分布的时候: $z = \log\left(\frac{\theta}{1-\theta}\right) + \log\left(\frac{u}{1-u}\right)$
3. Reparameterization: $b = H(z) \approx \sigma_\lambda(z) = \left(1 + \exp\left(-\frac{z}{\lambda}\right)\right)^{-1}$

$$\begin{aligned}\partial_\theta \mathbb{E}_{p(b|\theta)}[f(b)] &\approx \partial_\theta \mathbb{E}_{p(z|\theta)}\left[f\left(\sigma_\lambda(z)\right)\right] \\ &= \partial_\theta \mathbb{E}_{p(u)}\left[f\left(\sigma_\lambda(h(u,\theta))\right)\right] \\ &= \mathbb{E}_{p(u)}\left[\partial_\theta f\left(\sigma_\lambda(h(u,\theta))\right)\right]\end{aligned}$$

- 处理了 b 是离散随机变量的情况
- Biased，（unbiased需要 $\lambda \to 0$ ）
- λ 大，小variance，大bias

- λ 小，大variance，小bias
- 也就必须要调 λ ，做bias-variance trade-off

REBAR

Control variates:

$$\hat{g}=\mathbb{E}p(b \mid \theta)\left[(f(b)-C(b)) \partial_{\theta} \log (p(b \mid \theta))\right]+\partial \theta \mathbb{E}_{p(b \mid \theta)}[C(b)]$$

做法：使用Control variates并结合Concrete trick.

具体在于选择一个 C 能够尽可能降低variance.

降低手段一

C 的选择利用Concrete中的近似项

$$C=f\left(\sigma_{\lambda}(z)\right)$$

$$\mathbb{E}_{p(z \mid \theta)}\left[C(b) \partial_{\theta} \log (p(b \mid \theta))\right]=$$

$$\partial_{\theta} \mathbb{E} p(z \mid \theta) f(\sigma \lambda(z))=\mathbb{E}_{p(z \mid \theta)}\left[f\left(\sigma_{\lambda}(z)\right) \partial_{\theta} \log (p(z \mid \theta))\right]$$

原理： C 与 f 越相关，降低的方差就越大.

> https://www.wikiwand.com/en/Control_variates

降低手段二

> Law of Total variance: $\mathbb{E}[\operatorname{Var}(z \mid b)]=\operatorname{Var}(z)-\operatorname{Var}(\mathbb{E}(z \mid b))$
>
> https://www.wikiwand.com/en/Law_of_total_variance

说明平均而言 $z|b$ 的variance会更小，借助 $p(z \mid b)$ 来进一步减小variance

$$\mathbb{E} p(z \mid \theta)\left[f\left(\sigma_{\lambda}(z)\right) \partial_{\theta} \log (p(z \mid \theta))\right]=\mathbb{E} p(b)\left[\partial_{\theta} \mathbb{E} p(z \mid b)\left[f\left(\sigma_{\lambda}(z)\right)\right]\right]+\mathbb{E} p(b)\left[\mathbb{E}_{p(z \mid b)}\left[f\left(\sigma_{\lambda}(z)\right) \partial_{\theta} \log (p(b))\right]\right]$$

等式右边第一项就可以用Reparameterization：

$$\mathbb{E} p(b)\left[\partial \theta \mathbb{E} p(z \mid b)\left[f\left(\sigma_{\lambda}(z)\right)\right]\right]=\mathbb{E} p(b)\left[\mathbb{E}_{p(v)}\left[\partial_{\theta} f\left(\sigma_{\lambda}(\tilde{z})\right)\right]\right]$$

其中 $v \sim \operatorname{Unif}(0,1), \tilde{z}=\tilde{h}(v, b, \theta)$

第二项：

$$\mathbb{E} p(b)\left[\mathbb{E} p(z \mid b)\left[f\left(\sigma_{\lambda}(z)\right) \partial_{\theta} \log (p(b \mid \theta))\right]\right]=\mathbb{E} p(b)\left[\mathbb{E} p(v)\left[f\left(\sigma_{\lambda}(\tilde{z})\right) \partial_{\theta} \log (p(b \mid \theta))\right]\right]$$

最后的梯度估计为：

$$\hat{g} \theta=\mathbb{E} p(u, v)\left[\left(f(H(z))-f\left(\sigma_{\lambda}(\tilde{z})\right)\right) \partial_{\theta} \log (p(b))+\partial_{\theta} f\left(\sigma_{\lambda}(z)\right)-\partial_{\theta} f\left(\sigma_{\lambda}(\tilde{z})\right)\right]$$

降低手段三

λ 非超参，而是可优化的参数，优化目标是减少variance

$$\begin{aligned}\hat{g}_\lambda &= \partial_\lambda\left(\mathbb{E}\left[\hat{g}_\theta^2\right]-\left(\mathbb{E}\left[\hat{g}_\theta\right]\right)^2\right)\\&=\partial_\lambda\mathbb{E}\left[\hat{g}_\theta^2\right]\end{aligned}$$

$$\lambda \leftarrow \lambda-\alpha * \hat{g}_\lambda$$

问题：

- 只有一个 λ 来控制variance，容易under fit
- 梯度估计要求 f 可微

RELAX

将 C 替换成神经网络 C_ϕ ， ϕ 为网络的weights

$$\hat{g}\text{reLAX}=\mathbb{E}p(u,v)\left[\left[f(b)-\left[C_\phi(\tilde{z})\right]\right]\partial_\theta\log p(b\mid\theta)-\partial_\theta\left[C_\phi(\tilde{z})\right]\right]+\partial_\theta\left[C_\phi(z)\right]$$

- 在unbiased的前提下，利用网络weights ϕ 来控制降低variance
- f 不需要可微

整体算法流程

Algorithm 1 LAX: Optimizing parameters and a gradient control variate simultaneously.

Require: $f(\cdot)$, $\log p(b|\theta)$, reparameterized sampler $b=T(\theta,\epsilon)$, neural network $c_\phi(\cdot)$, step sizes α_1,α_2

while not converged **do**

- $\epsilon\sim p(\epsilon)$ ▷ Sample noise
- $b\leftarrow T(\epsilon,\theta)$ ▷ Compute input
- $\hat{g}_\theta\leftarrow\left[f(b)-c_\phi(b)\right]\nabla_\theta\log p(b|\theta)+\nabla_\theta c_\phi(b)$ ▷ Estimate gradient of objective
- $\hat{g}_\phi\leftarrow\partial\hat{g}_\theta^2/\partial\phi$ ▷ Estimate gradient of variance of gradient
- $\theta\leftarrow\theta-\alpha_1\hat{g}_\theta$ ▷ Update parameters
- $\phi\leftarrow\phi-\alpha_2\hat{g}_\phi$ ▷ Update control variate

end while

return θ

Algorithm 2 RELAX: Low-variance control variate optimization for black-box gradient estimation.

Require: $f(\cdot)$, $\log p(b|\theta)$, reparameterized samplers $b = H(z)$, $z = S(\epsilon, \theta)$ and $\tilde{z} = S(\epsilon, \theta|b)$, neural network $c_\phi(\cdot)$, step sizes α_1, α_2

while not converged **do**

- $\epsilon_i, \widetilde{\epsilon}_i \sim p(\epsilon)$ ▷ Sample noise
- $z_i \leftarrow S(\epsilon_i, \theta)$ ▷ Compute unconditional relaxed input
- $b_i \leftarrow H(z_i)$ ▷ Compute input
- $\widetilde{z}_i \leftarrow S(\widetilde{\epsilon}_i, \theta|b_i)$ ▷ Compute conditional relaxed input
- $\hat{g}_\theta \leftarrow [f(b_i) - c_\phi(\widetilde{z}_i)]\, \nabla_\theta \log p + \nabla_\theta c_\phi(z_i) - \nabla_\theta c_\phi(\widetilde{z}_i)$ ▷ Estimate gradient
- $\hat{g}_\phi \leftarrow \partial \hat{g}_\theta^2 / \partial \phi$ ▷ Estimate gradient of variance of gradient
- $\theta \leftarrow \theta - \alpha_1 \hat{g}_\theta$ ▷ Update parameters
- $\phi \leftarrow \phi - \alpha_2 \hat{g}_\phi$ ▷ Update control variate

end while

return θ

实验

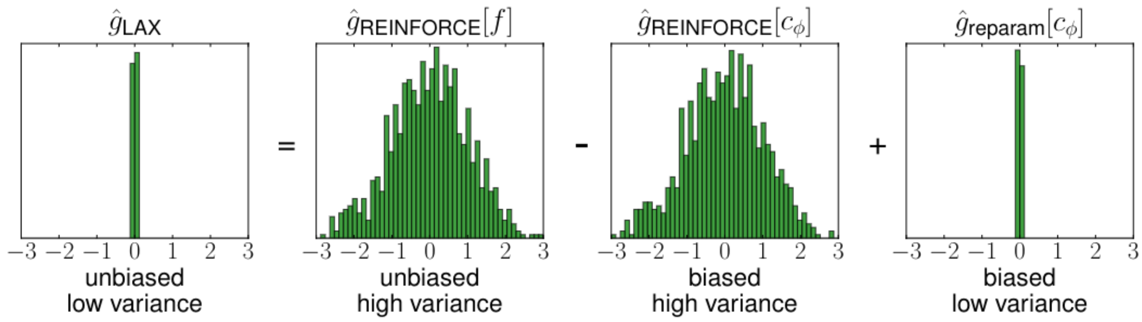

Figure 2: Histograms of samples from the gradient estimators that create LAX. Samples generated from our one-layer VAE experiments (Section 6.2).

Toy experiment

最小化目标： $\mathbb{E}_{p(b|\theta)}\left[(b-t)^2\right]$

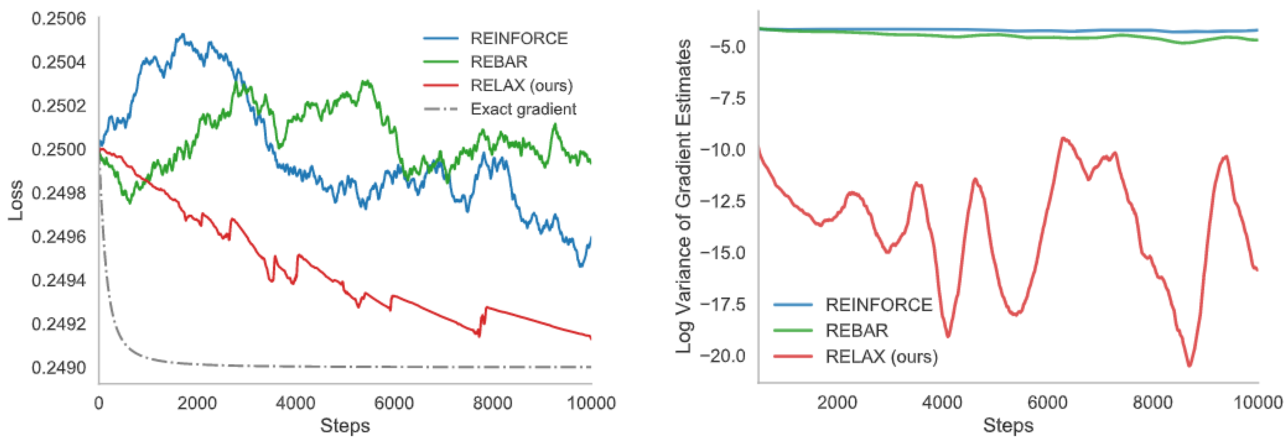

Figure 1: *Left:* Training curves comparing different gradient estimators on a toy problem: $\mathcal{L}(\theta) = \mathbb{E}_{p(b|\theta)}[(b-0.499)^2]$ *Right:* Log-variance of each estimator's gradient.

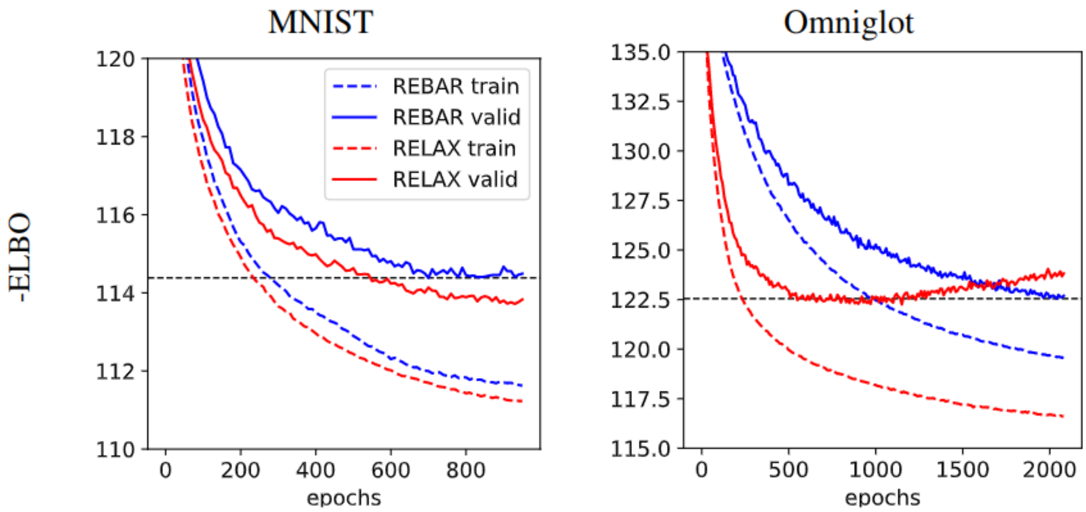

Figure 4: Training curves for the VAE Experiments with the one-layer linear model. The horizontal dashed line indicates the lowest validation error obtained by REBAR.

RELAX的梯度始终大于0，说明梯度估计中的Reparameterization部分提供的梯度方向信息始终是正确的

离散VAE

> To take advantage of the available structure in the loss function

$$c_\phi(z) = f\left(\sigma_\lambda(z)\right) + \hat{r}_\rho(z)$$

Dataset	Model	Concrete	NVIL	MuProp	REBAR	RELAX
MNIST	Nonlinear	−102.2	−101.5	-101.1	-81.01	**-78.13**
	linear one-layer	-111.3	−112.5	−111.7	-111.6	**-111.20**
	linear two-layer	-99.62	−99.6	−99.07	-98.22	**-98.00**
Omniglot	Nonlinear	−110.4	−109.58	-108.72	-56.76	**-56.12**
	linear one-layer	-117.23	−117.44	−117.09	-116.63	**-116.57**
	linear two-layer	-109.95	−109.98	−109.55	-108.71	**-108.54**

Table 1: Highest training ELBO for discrete variational autoencoders.

Dataset	Model	REBAR	RELAX
MNIST	one-layer linear	-114.32	**-113.62**
	two-layer linear	-101.20	**-100.85**
	Nonlinear	**-111.12**	119.19
Omniglot	one-layer linear	-122.44	**-122.11**
	two-layer linear	-115.83	**-115.42**
	Nonlinear	**-127.51**	128.20

Table 3: Highest obtained validation ELBO.

Model	Cart-pole	Lunar lander	Inverted pendulum
A2C	1152 ± 90	162374 ± 17241	6243 ± 164
LAX/RELAX	$\mathbf{472 \pm 114}$	$\mathbf{68712 \pm 20668}$	$\mathbf{2067 \pm 412}$

Table 2: Mean episodes to solve tasks. Definitions of solving each task can be found in Appendix E.

过拟合：

> We believe the decreasein validation performance for the nonlinear models was due to overfitting caused by improved optimization of an under-regularized model. We leave exploring this phenomenon to further work.

强化学习

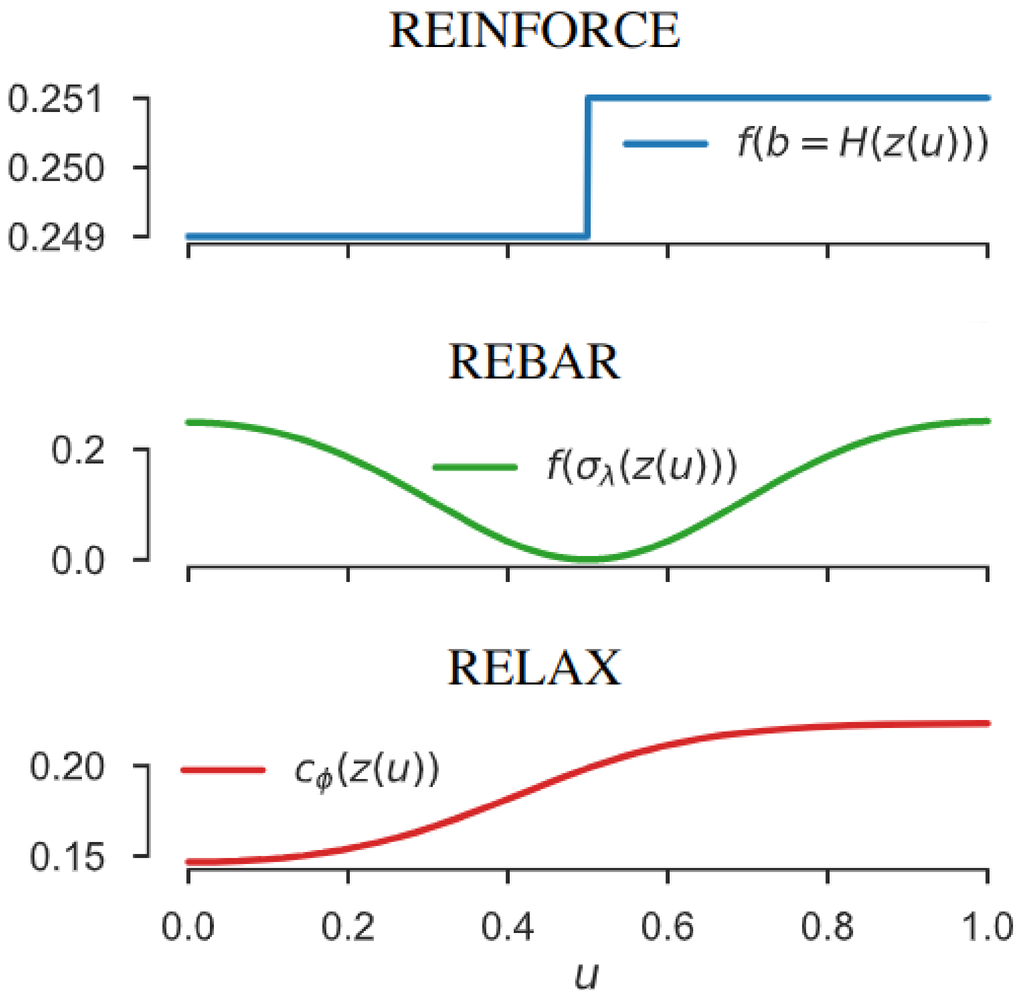

Figure 3: The optimal relaxation for a toy loss function, using different gradient estimators. Because REBAR uses the concrete relaxation of f, which happens to be implemented as a quadratic function, the optimal relaxation is constrained to be a warped quadratic. In contrast, RELAX can choose a free-form relaxation.

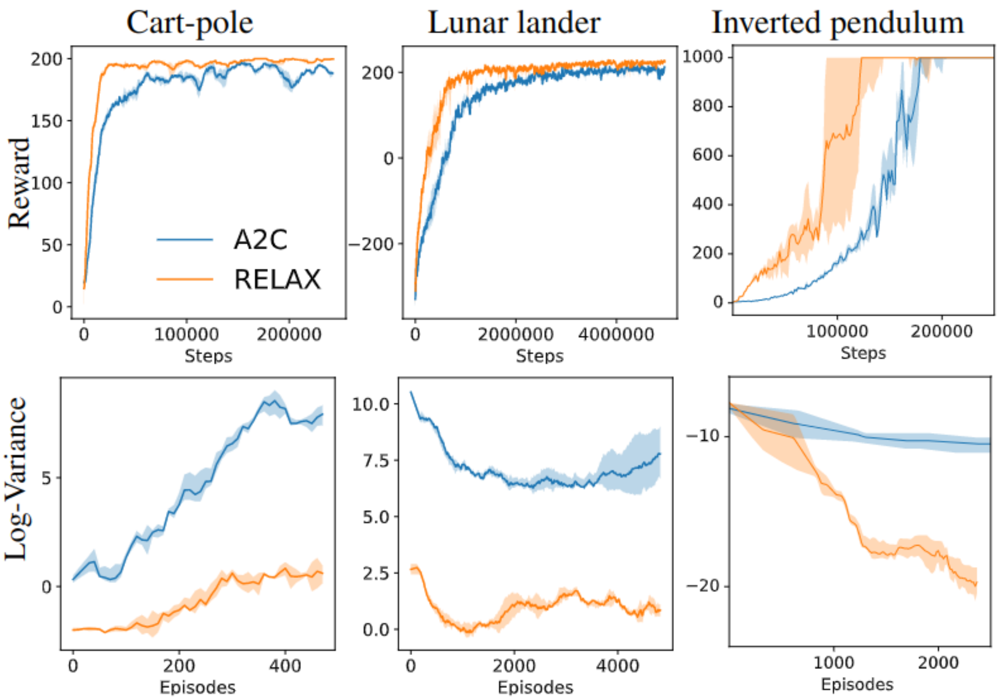

Figure 5: *Top row:* Reward curves. *Bottom row:* Log-variance of policy gradients. In each curve, the center line indicates the mean reward over 5 random seeds. The opaque bars in the top row indicate the 25th and 75th percentiles. The opaque bars in the bottom row indicate 1 standard deviation. Since the gradient estimator is defined at the end of each episode, we display log-variance per episode.