Backpropagation Through
Backpropagation Through The Void
Backpropagation Through the Void

Will Grathwohl
Dami Choi
Yuhuai Wu
Geoff Roeder
David Duvenaud
Where do we see this guy?

\[ \mathcal{L}(\theta) = \mathbb{E}_{p(b|\theta)}[f(b)] \]

- Variational Inference
- Hamiltonian Monte Carlo
- Policy Optimization
- Hard Attention
Bayesian optimization doesn’t scale yet

- Bayesopt is usually expensive, relative to model evals
- Global surrogate models not good enough in high dim.
- Even for expensive black-box functions, gradient-based optimization is embarrassingly competitive
- Can we add some cheap model-based optimization to REINFORCE?

Shahriari et al., 2016
REINFORCE (Williams, 1992)

\[ \hat{g}_{\text{REINFORCE}}[f] = f(b) \frac{\partial}{\partial \theta} \log p(b|\theta), \quad b \sim p(b|\theta) \]

- Unbiased
- Works for any f, not differentiable or even unknown
- high variance
\[ \hat{g}_{\text{REINFORCE}}[f] = f(b) \frac{\partial}{\partial \theta} \log p(b|\theta), \quad b \sim p(b|\theta) \]
Reparameterization Trick:

\[ \hat{g}_{\text{reparam}}[f] = \frac{\partial f}{\partial b} \frac{\partial b}{\partial \theta} \]

\[ b = T(\theta, \epsilon), \epsilon \sim p(\epsilon) \]

- Usually lower variance
- Unbiased
- Gold standard, allowed huge continuous models
- Requires \( f(b) \) to be known and differentiable
- Requires \( p(b|\theta) \) to be differentiable
\[ \hat{g}_{\text{reparam}}[f] = \frac{\partial f}{\partial b} \frac{\partial b}{\partial \theta} \quad b = T(\theta, \epsilon), \epsilon \sim p(\epsilon) \]
Concrete/Gumbel-softmax

\[ \hat{g}_{\text{concrete}}[f] = \frac{\partial f}{\partial \sigma(z/t)} \frac{\partial \sigma(z/t)}{\partial \theta} \]

\[ z = T(\theta, \epsilon), \epsilon \sim p(\epsilon) \]

- Tune variance vs bias
- Works well in practice for discrete models
- Biased
  - \( f(b) \) must be known and differentiable
  - \( p(z|\theta) \) must be differentiable
- Uses undefined behavior of \( f(b) \)
Control Variates

- Allow us to reduce variance of a Monte Carlo estimator

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

- Variance is reduced if \( \text{corr}(g, c) > 0 \)

- Need to adapt \( g \) as problem changes during optimization
Our Approach

$$\hat{g}_{LAX} = \text{REINFORCE}[f] - \text{REINFORCE}[c_\phi] + \text{reparam}[c_\phi]$$

$$= [f(b) - c_\phi(b)] \frac{\partial}{\partial \theta} \log p(b|\theta) + \frac{\partial}{\partial \theta} c_\phi(b)$$
Our Approach

\[ \hat{g}_{\text{LAX}} = g_{\text{REINFORCE}}[f] - g_{\text{REINFORCE}}[c_\phi] + g_{\text{reparam}}[c_\phi] \]
Optimizing the Control Variate

For any unbiased estimator we can get Monte Carlo estimates for the gradient of the variance of \( \hat{g} \)

Use to optimize \( c_\phi \)

Got trick from Ruiz et al. and REBAR paper

\[
\frac{\partial}{\partial \phi} \text{Variance}(\hat{g}) = \mathbb{E} \left[ \frac{\partial}{\partial \phi} \hat{g}^2 \right]
\]
A self-tuning gradient estimator

- Jointly optimize original problem and surrogate together with stochastic optimization
- Requires higher-order derivatives

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 \( \alpha_1, \alpha_2 \)

while not converged do
    \( \epsilon \sim p(\epsilon) \)
    \( b \leftarrow T(\epsilon, \theta) \)
    \( \hat{g}_\theta \leftarrow [f(b) - c_\phi(b)] \nabla_\theta \log p(b|\theta) + \nabla_\theta c_\phi(b) \)
    \( \hat{g}_\phi \leftarrow \partial \hat{g}_\theta / \partial \phi \)
    \( \theta \leftarrow \theta - \alpha_1 \hat{g}_\theta \)
    \( \phi \leftarrow \phi - \alpha_2 \hat{g}_\phi \)
end while

return \( \theta \)
\[ \hat{g}_{\text{LAX}} = \left[ f(b) - c_\phi(b) \right] \frac{\partial}{\partial \theta} \log p(b|\theta) + \frac{\partial}{\partial \theta} c_\phi(b) \]
What about discrete variables?
Extension to discrete $p(b|\theta)$

$$\hat{g}_{DLAX} = f(b) \frac{\partial}{\partial \theta} \log p(b|\theta) - c_\phi(z) \frac{\partial}{\partial \theta} \log p(z|\theta) + \frac{\partial}{\partial \theta} c_\phi(z)$$

$b = H(z), z \sim p(z|\theta)$

$H(z) = b \sim p(b|\theta)$

- Unbiased for all $c_\phi$
Extension to discrete $p(b|\theta)$

$$\hat{g}_{\text{RELAX}} = \left[ f(b) - c_\phi(\tilde{z}) \right] \frac{\partial}{\partial \theta} \log p(b|\theta) + \frac{\partial}{\partial \theta} c_\phi(z) - \frac{\partial}{\partial \theta} c_\phi(\tilde{z})$$

$$b = H(z), \; z \sim p(z|\theta), \; \tilde{z} \sim p(z|b, \theta)$$

- Main trick introduced in REBAR (Tucker et al. 2017).
- We just noticed it works for any $c()$
- REBAR is special case of RELAX where $c$ is concrete relaxation
- We use autodiff to tune entire surrogate, not just temperature
Toy Example

\[ E_{p(b|\theta)} \left[ (t - b)^2 \right] \]

- Used to validate REBAR (used \( t = .45 \))
- We use \( t = .499 \)
- REBAR, REINFORCE extremely slow in this case
- Can RELAX improve?
• massively reduced variance

• Surrogate needs time to catch up
Analyzing the Surrogate

- REBAR’s fixed surrogate only adapts temperature param.

- RELAX surrogate balances REINFORCE variance and reparameterization variance

- Optimal surrogate is always smooth
Define functions, not computation graphs

```python
def relax(params, est_params, noise_u, noise_v, func_vals):
    samples = bernoulli_sample(params, noise_u)
    log_temperature, nn_params = est_params

    def surrogate(relaxed_samples):
        return nn_predict(nn_params, relaxed_samples)

    def surrogate_cond(params):
        cond_noise = conditional_noise(params, samples, noise_v)  # z tilde
        return concrete(params, log_temperature, cond_noise, surrogate)

    grad_surrogate = elementwise_grad(concrete)(params, log_temperature, noise_u, surrogate)
    surrogate_cond, grad_surrogate_cond = value_and_grad(surrogate_cond)(params)
    return reinforce(params, noise_u, func_vals - surrogate_cond) + \
        grad_surrogate - grad_surrogate_cond

def relax_all(params, est_params, noise_u, noise_v, f):
    # Returns objective, gradients, and gradients of variance of gradients.
    func_vals = f(bernoulli_sample(params, noise_u))
    var_vjp, grads = make_vjp(relax, argnum=1)(params, est_params, noise_u, noise_v, func_vals)
    d_var_d_est = var_vjp(2 * grads / grads.shape[0])
    return func_vals, grads, d_var_d_est
```
Discrete VAEs

\[
\log p(x) \geq \mathcal{L}(\theta) = \mathbb{E}_{q(b|x)}[\log p(x|b) + \log p(b) - \log q(b|x)]
\]

- Latent state is 200 Bernoulli variables
- Can’t use reparameterization trick
- Can still use our knowledge of structure of model, combining REBAR and RELAX:

\[
c_\phi(z) = f(\sigma_\lambda(z)) + r_\rho(z)
\]
## Bernoulli VAE Results

![MNIST and Omniglot ELBO plots](image)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Concrete</th>
<th>NVIL</th>
<th>MuProp</th>
<th>REBAR</th>
<th>RELAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>Nonlinear</td>
<td>-102.2</td>
<td>-101.5</td>
<td>-101.1</td>
<td>-81.01</td>
<td>-78.13</td>
</tr>
<tr>
<td></td>
<td>linear one-layer</td>
<td>-111.3</td>
<td>-112.5</td>
<td>-111.7</td>
<td>-111.6</td>
<td>-111.20</td>
</tr>
<tr>
<td></td>
<td>linear two-layer</td>
<td>-99.62</td>
<td>-99.6</td>
<td>-99.07</td>
<td>-98.22</td>
<td>-98.00</td>
</tr>
<tr>
<td>Omniglot</td>
<td>Nonlinear</td>
<td>-110.4</td>
<td>-109.58</td>
<td>-108.72</td>
<td>-56.76</td>
<td>-56.12</td>
</tr>
<tr>
<td></td>
<td>linear one-layer</td>
<td>-117.23</td>
<td>-117.44</td>
<td>-117.09</td>
<td>-116.63</td>
<td>-116.57</td>
</tr>
</tbody>
</table>

Table 1: Highest training ELBO for discrete variational autoencoders.
Rederiving Actor-Critic

\[ \hat{J}_{AC} = \sum_{t=1}^{T} \frac{\partial \log \pi(a_t|s_t, \theta)}{\partial \theta} \left[ \sum_{t'=t}^{T} r_{t'} - c_\phi(s_t) \right] \]

- \( c_\phi \) is an estimate of the value function

- This is exactly the REINFORCE estimator using an estimate of the value function as a control variate

- Why not use action in control variate?

- Dependence on action would add bias
LAX for RL

\[
\hat{J}_{LAX} = \sum_{t=1}^{T} \frac{\partial \log \pi(a_t | s_t, \theta)}{\partial \theta} \left[ \sum_{t'=t}^{T} r_{t'} - c_\phi(s_t, a_t) \right] + \frac{\partial}{\partial \theta} c_\phi(s_t, a_t)
\]

- Action-dependence in control variate
- unbiased for policy, and unbiased for baseline
- Standard baseline optimization methods minimize squared error from reward or value function. We directly minimize variance.
Faster convergence, but real story is unbiased critic updates.

Excellent criticism of experimental setup in “The Mirage of Action-Dependent Baselines in Reinforcement Learning” (Tucker et al. 2018). Better experiments would examine high-dimensional action regime.
RELAX Properties

• Pros:
  • unbiased
  • low variance (after tuning)
  • usable when $f(b)$ is unknown, or not differentiable
  • useable when $p(b|\theta)$ is discrete

• Cons:
  • need to define surrogate
  • when progress is made, need to wait for surrogate to adapt
  • Higher-order derivatives still awkward in TF and PyTorch
Local surrogates are a nice compromise

- Global surrogate models not good enough in high dim.
- Local surrogates are less demanding to construct
- Local minima less of a problem in high dim.
- Want low variance? Take gradient of variance

Shahriari et al., 2016
Aside: Evolution Strategies optimize a linear surrogate

\[ \hat{w} = (X^T X)^{-1} X^T y \]

\[ \approx \mathbb{E} [(X^T X)]^{-1} X^T y \]

\[ = [I \sigma^2]^{-1} X^T y \]

\[ = [I \sigma^2]^{-1} (\epsilon \sigma)^T y \]

\[ = \sum_i \frac{\epsilon_i y_i}{\sigma} \]

\[ = \sum_i \frac{\epsilon_i f(\epsilon_i \sigma)}{\sigma} \]

\[ \epsilon \sim \mathcal{N}(0, I) \]

\[ x = \epsilon \sigma \]
Aside: Evolution Strategies optimize a linear surrogate

- Throws away all observations each step
- Use a neural net surrogate, and experience replay
- Distributed ES algorithm works for any gradient-free optimization algorithm
- w/ students Geoff Roeder, Yuhuai (Tony) Wu, Jaiming Song
Future Work

- What does the optimal surrogate look like? Use calculus of variations.

- Train the surrogate off-policy: $c_\phi(a, s, \pi)$

- REBAR, RELAX for more complicated discrete objects, e.g. trees

- Mostly open problem: Control variates for sequential discrete choices.

Linderman et al., 2017

Mena et al., 2018
Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, Jesse Bettencourt, David Duvenaud

Thanks!

https://github.com/duvenaud/relax