Variance reduction techniques for stochastic optimization

Matthew W. Hoffman

University of Cambridge

February 12, 2015

Variance reduction in Monte Carlo methods

We're interested in computing

$$\mathbb{E}_{p(x)}\left[f(x)\right] \approx \underbrace{\frac{1}{N}\sum_{i=1}^{N} f(x^i)}_{F} \quad \text{where } x^i \sim p(x)$$

but F is now a random variable

Problem: F may have high variance

Solution: replace F with a new quantity F' with the same expectation, but lower variance

$$\begin{aligned} \mathbb{E}[F'] &= \mathbb{E}[F] = \mathbb{E}[f(x)], \\ \mathrm{var}[F'] &\leq \mathrm{var}[F]. \end{aligned}$$

Control variates

Consider an additional function $\phi(x)$ whose expectation $\mu_\phi = \mathbb{E}[\phi(x)]$ we know. We can introduce this function and write

$$\mathbb{E}\left[f(x)\right] = \underbrace{\mathbb{E}\left[f(x) - \phi(x)\right]}_{\text{use Monte Carlo here}} + \underbrace{\mu_\phi}_{\text{we know this}}$$

Nothing ground-breaking, but what about the variance?

$$\mathrm{var}\left[f(x) - \phi(x)\right] = \mathrm{var}\left[f(x)\right] - 2\,\mathrm{cov}\left[f(x), \phi(x)\right] + \mathrm{var}\left[\phi(x)\right]$$

i.e. we can get a reduction in variance if f and ϕ are **strongly correlated**

> ϕ is our **control variate**—so-called because it allows us to control the variance of our estimate

A few observations:

- we want $\text{cov}\left[f(x), \phi(x)\right] > \frac{1}{2}\,\text{var}\left[\phi(x)\right]$
- the control variate which minimizes the variance is easy, $\phi(x) = f(x)$, but this assumes we already know the integral
- instead we will often use simple variates of the form $a\phi$ and **optimize a assuming we are given ϕ**

Scaling the control variate

Now lets multiply the control variate by a scalar a,

$$\begin{aligned} f'(x) &= f(x) - a(\phi(x) - \mu_\phi) \\ \mathrm{var}\left[f'(x)\right] &= \mathrm{var}\left[f(x)\right] - 2a\,\mathrm{cov}\left[f(x), \phi(x)\right] + a^2\,\mathrm{var}\left[\phi(x)\right] \end{aligned}$$

we can easily see by taking its derivative that this is minimized by

$$a = \frac{\mathrm{cov}\left[f(x), \phi(x)\right]}{\mathrm{var}\left[\phi(x)\right]}.$$

Plugging this in and dividing by the original variance we get

$$\frac{\mathrm{var}\left[f'(x)\right]}{\mathrm{var}\left[f(x)\right]} = 1 - \mathrm{corr}^2\left[f(x), \phi(x)\right]$$

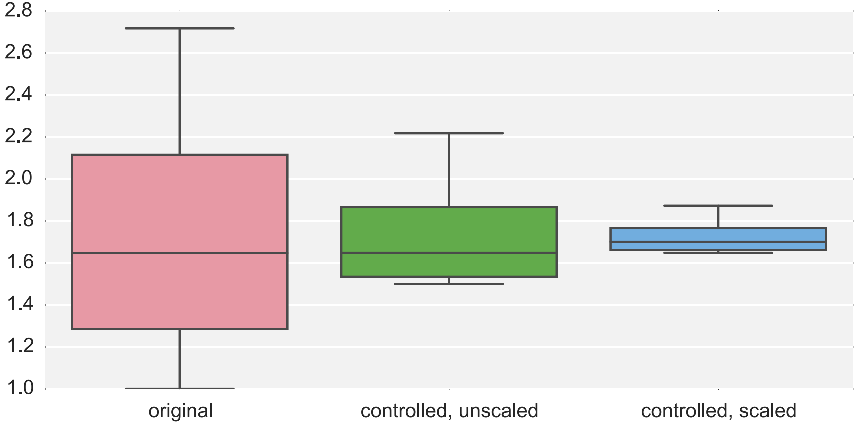
i.e. the reduction in variance is directly related to their correlation.

[Ross, 2006, Ch. 8]

- in computing a we typically won't have $\mathrm{var}[\phi(x)]$ and $\mathrm{cov}[f(x), \phi(x)]$ but we can still use their empirical estimates.

A simple example

Consider computing the expectation $\mathbb{E}[e^x]$ where $x \sim \mathcal{U}(0, 1)$ and use as control variate $\phi(x) = x - 0.5$.



[Ross, 2006, Ex. 8d]

Simple code generating the last plot:

```
x = np.random.rand(n)
f = np.exp(x)
phi = x - 0.5

# NOTE: requires touching all our data!
_, cov, _, var = np.cov([f, phi]).ravel()
f1 = f - phi
f2 = f - (cov/var) * phi
```

The comment is important; sometimes it may be too costly (or impossible) to view all our data.

Antithetic variables

as a special case of control variates

Consider a random variable given as a function of uniform variates

$$X = h(U_1, \ldots, U_n), \text{ and}$$
$$W = h(1 - U_1, \ldots, 1 - U_n)$$

We can use $0.5(X - W)$ as a control variate in order to estimate $\mathbb{E}[X]$

- also known as antithetic variables, resulting in the estimator $0.5(X + W)$
- this relies X and W being negatively correlated,
- provably reduces the variance when h is monotonic (either decreasing or increasing) in its inputs

Vector-valued variates

Given vector-valued $\mathbf{f}(x)$ we should use a control variate $\phi(x)$ with expectation $\boldsymbol{\mu}_\phi$ and an appropriately-sized matrix $\mathbf{A}$ to define

$$\mathbf{f}'(x) = \mathbf{f}(x) - \mathbf{A}^\mathsf{T}(\phi(x) - \boldsymbol{\mu}_\phi)$$

Let's say then that we select $\mathbf{A}$ to minimize $\operatorname{tr}\left[\operatorname{cov}[\mathbf{f}'(x)]\right]$, leading to

$$\mathbf{A} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\Omega} + \boldsymbol{\Omega}^\mathsf{T})/2 \qquad \text{where} \qquad \begin{aligned} \boldsymbol{\Sigma} &= \operatorname{cov}[\phi(x)] \\ \boldsymbol{\Omega} &= \operatorname{cov}[\mathbf{f}(x), \phi(x)] \end{aligned}$$

This does require inversion of $\boldsymbol{\Sigma}$ though...

[Paisley et al., 2012, Wang et al., 2013]

If we assume that **A** is diagonal the optimal choice is given by

$$a_{ii} = \frac{\text{cov}[f_i(x), \phi_i(x)]}{\text{var}[\phi_i(x)]}$$

which is the same as the scalar case applied to each corresponding dimension

Assuming a single scalar value we get back

$$a = \frac{\sum_i \text{cov}[f_i(x), \phi_i(x)]}{\sum_i \text{var}[\phi_i(x)]}$$

which can be obtained by considering the scalar control variate case where we now just want to minimize the sum of variances (as done by Paisely et al.)

- for what follows I will assume $\mu_\phi = \mathbb{E}[\phi(x)] = 0$ and that the control variate contains any multiplier a

Stochastic gradient descent (SGD)

Often we find ourselves wanting to minimize some expected cost function

$$J(\theta) = \mathbb{E}_{p(x)}\left[c_\theta(x)\right]$$

whose gradient is given by

$$\begin{aligned}\nabla J(\theta) &= \int p(x)\,\nabla c_\theta(x)\,dx \\ &\approx \tfrac{1}{N}\sum_{i=1}^{N}\nabla c_\theta(x^i) \quad \text{for } x^i \sim p(\cdot)\end{aligned}$$

This is just a Monte Carlo estimate of something (that just happens to be a gradient)! So we can apply a suitable control variate.

SGD for logistic regression

For logistic regression we want to classify inputs $\mathbf{x}$ as one of two classes, $y \in \{-1, 1\}$, the likelihood for which is

$$p_{\boldsymbol{\theta}}(y|\mathbf{x}) = \sigma(y\boldsymbol{\theta}^\mathsf{T}\mathbf{x}) \qquad \sigma(z) = (1 + \exp(-z))^{-1} \text{ is the logistic}$$

The cost for a single observation is its log-likelihood, with gradient

$$\nabla c_\theta(\mathbf{x}, y) = y\mathbf{x}\sigma(-y\boldsymbol{\theta}^\mathsf{T}\mathbf{x})$$

The noisy gradient is

$$\nabla J(\theta) \approx \tfrac{1}{N} \sum_{i=1}^{N} \nabla c_\theta(\mathbf{x}^i, y^i)$$

[Wang et al., 2013]

SGD for logistic regression

For logistic regression we want to classify inputs $\mathbf{x}$ as one of two classes, $y \in \{-1, 1\}$, the likelihood for which is

$$p_{\boldsymbol{\theta}}(y|\mathbf{x}) = \sigma(y\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}) \qquad \begin{aligned} &\sigma(z) = (1 + \exp(-z))^{-1} \text{ is the logistic} \\ &\hat{\sigma}(z) = \sigma(\hat{z})(1 + \sigma(-\hat{z})(z - \hat{z})) \\ &\qquad \text{is its 1st-order Taylor exp.} \end{aligned}$$

The cost for a single observation is its log-likelihood, with gradient

$$\nabla c_{\theta}(\mathbf{x}, y) = y\mathbf{x}\sigma(-y\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x})$$
$$\phi(\mathbf{x}, y) = y\mathbf{x}\hat{\sigma}(-y\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}) - \boldsymbol{\mu}_{\phi}$$

The noisy gradient is

$$\nabla J(\theta) \approx \tfrac{1}{N} \sum_{i=1}^{N} \nabla c_{\theta}(\mathbf{x}^i, y^i) - \phi(\mathbf{x}^i, y^i)$$

[Wang et al., 2013]

What did I leave out?

- the expectation $\boldsymbol{\mu}_\phi$ requires the mean and variance of both positive and negative inputs; **this requires a full pass over the data**
- note though that we need only do this once

SGD with parameterized distributions

Consider an objective where the distribution itself is parameterized

$$J(\theta) = \mathbb{E}_{p_\theta(x)}\left[c(x)\right]$$

and whose gradient is

$$\begin{aligned}
\nabla J(\theta) &= \int \nabla p_\theta(x)\, c(x)\, dx \\
&= \int p_\theta(x)\, \nabla \log p_\theta(x)\, c(x)\ dx \\
&\approx \tfrac{1}{N} \sum_{i=1}^{N} \nabla \log p_\theta(x^i)\, c(x^i) \quad \text{for } x^i \sim p_\theta(\cdot)
\end{aligned}$$

An aside: so far this has **nothing to do with control variates** or variance reduction in any way. We are just evaluating a gradient.

SGD for policy learning: REINFORCE, GPOMDP

We can now apply this to reinforcement learning where

- $x = (s_{0:T}, a_{0:T})$ represents a trajectory
- $c(x) = -\sum_t \gamma^t r(s_t, a_t)$ are summed (discounted) rewards
- the probability of trajectories has Markovian structure,

$$p_\theta(x) = \mu(s_0) \prod_t \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

Plugging this into the previous framework we get,

$$\nabla J(\theta) \approx -\tfrac{1}{N} \sum_{i=1}^{N} \Big[\sum_t \nabla \log \pi_\theta(a_t^i|s_t^i) \Big] \Big[\sum_t \gamma^t r(s_t^i, a_t^i) \Big]$$

this is **not quite** the REINFORCE algorithm

[Williams, 1992, Baxter and Bartlett, 2001]

Eliminating expectations in REINFORCE

Let $z_k = (s_k, a_k)$ be a state/action pair at time k. The previous gradient is a sum of many "cross-time" terms,

$$\begin{aligned} &\mathbb{E}[\gamma^k r(z_k) \nabla \log \pi_\theta(z_t)] \quad \text{for } k < t \\ &\qquad = \int p_\theta(z_k)\, \gamma^k r(z_k) \Big[\underbrace{\int p_\theta(z_t | z_k)\, \nabla \log \pi_\theta(z_t)\, dz_t}_{\text{expectation of a score}} \Big]\, dz_k \end{aligned}$$

By eliminating these terms (their expectation is zero!) we get REINFORCE,

$$\nabla J(\theta) \approx -\frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \sum_{k=t}^{T} \nabla \log \pi_\theta(a_t^i | s_t^i)\, \gamma^k r(s_k^i, a_k^i)$$

Note: if we didn't eliminate these terms they would **only add variance**

Control variates in REINFORCE (baselines)

In the same way that we eliminated zero-mean terms in the previous slide we can also add terms,

$$\nabla J(\theta) \approx -\frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \sum_{k=t}^{T} \nabla \log \pi_\theta(a_t^i | s_t^i) \left[\gamma^k r(s_k^i, a_k^i) - \hat{b}_k(s_k^i, a_k^i) \right]$$

which is called a **baseline**, i.e. a "baseline reward" to improve on

This can be interpreted as a control variate of the form

$$\phi(x) = \sum_{t=0}^{T} \sum_{k=t}^{T} \nabla \log \pi_\theta(a_t | s_t) \hat{b}_k(s_k, a_k)$$

which so long as b_k is computed using only state/action pairs **before** time k will have expectation zero

Choice of baseline

There is some analysis in Greensmith et al. providing an **optimal baseline** under various settings—a bit complicated (and different from the earlier analysis)

However, a common baseline to use is the averaged reward:

$$\hat{b}_k = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{K}\gamma^t r(s_t^i, a_t^i)$$

in some sense this is intuitive and gives rise to the **baseline** name:

by combining this with our gradient the reward provides us with an improvement over the average

Actor-critic methods

Another technique involves using the value function as a baseline,

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^{t} r(s_t, a_t) | s_0 = s\right]$$

which is similar to the averaged-reward baseline presented earlier

Actor-critic methods extend this to using compatible function approximation for the value-function (approximate using a linear function of the policy gradient)

The Natural Actor-Critic takes these ideas and applies the *natural gradient*. Whether this counts as a *variance reduction* technique is a bit murky.

[Sutton et al., 2000, Konda and Tsitsiklis, 2000, Peters and Schaal, 2006]

References I

J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, pages 319–350, 2001.

V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*, volume 13, pages 1008–1014, 2000.

J. Paisley, D. Blei, and M. Jordan. Variational bayesian inference with stochastic search. In *the International Conference on Machine Learning*, 2012.

J. Peters and S. Schaal. Policy gradient methods for robotics. In *International Conference on Intelligent Robots and Systems*, pages 2219–2225, 2006.

S. M. Ross. *Simulation*. Academic Press, 4 edition, 2006.

References II

R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 13, pages 1057–1063, 2000.

C. Wang, X. Chen, A. J. Smola, and E. P. Xing. Variance reduction for stochastic gradient optimization. In *Advances in Neural Information Processing Systems*, pages 181–189, 2013.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4): 229–256, 1992.