Variance Reduction via Antithetic Sampling

Daniel Gonzalez
Department of Mathematics, Florida State University

April 26, 2019

Abstract

Variational Bayesian inference methods are a family of techniques approximating intractable integrals, often employing stochastic optimization techniques. These methods typically use independent Monte Carlo samples to estimate gradients. In this paper, we give an overview of a technique employed in (arXiv:1810.02555) for generating Normally-distributed Monte Carlo samples which are correlated, but more representative, and which can thus be used to reduce the variance in estimation.

1 Introduction

Gradient-based optimization methods are a robust and powerful way of attacking a wide class of problems in applied mathematics, science, and engineering. Optimizing expectations using gradient optimization (e.g. gradient descent) is particularly prevalent in machine learning, including variational inference and reinforcement learning. Problems in these fields typically involve computing expectations and gradients which are analytically intractable. Instead, practitioners frequently employ Monte Carlo methods to estimate those expectations and their gradients. While Monte Carlo methods lead to unbiased estimation of the target expectations and gradients, they suffer from high variance when a small amount of samples are used. The variance in a Monte Carlo estimator can be naïvely reduced by using more samples, but the computational cost quickly becomes impractical.

One way of reducing the variance in the Monte Carlo estimators while maintaining a low number of samples is through the method of *antithetic variates*. According to the antithetic variates approach, for every sample generated from a given distribution for the Monte Carlo simulation, we also generate a corresponding *antithetic* sample to which it is negatively correlated. Intuitively, this allows us to decrease the distance between the moments of the samples and the moments of the quantities we are estimating (called population moments), therefore reducing the variance in the estimation.

Antithetic sampling, however, does not come without its drawbacks. One key challenge in its application to machine learning is ensuring that the antithetic samples are correctly distributed to allow for unbiased estimation. Another concern, which is addressed in the original paper being reviewed but not here, is the differentiability of the antithetic process. This is of particular interest in gradient-based optimization methods the optimization will typically fail if the entire estimation process is not end-to-end differentiable.

In this paper, which is a review of Differentiable Antithetic Sampling for Variational Inference, we focus our attention on the generation of Gaussian variates using antithetic sampling. This is based on a method given by Marsaglia & Good in 1980 for generating Gaussian variates with a specified sample mean and sample variance. We provide pseudocode for the algorithms, and present numerical results demonstrating the algorithms' efficacy.

2 Generating Gaussian Variates

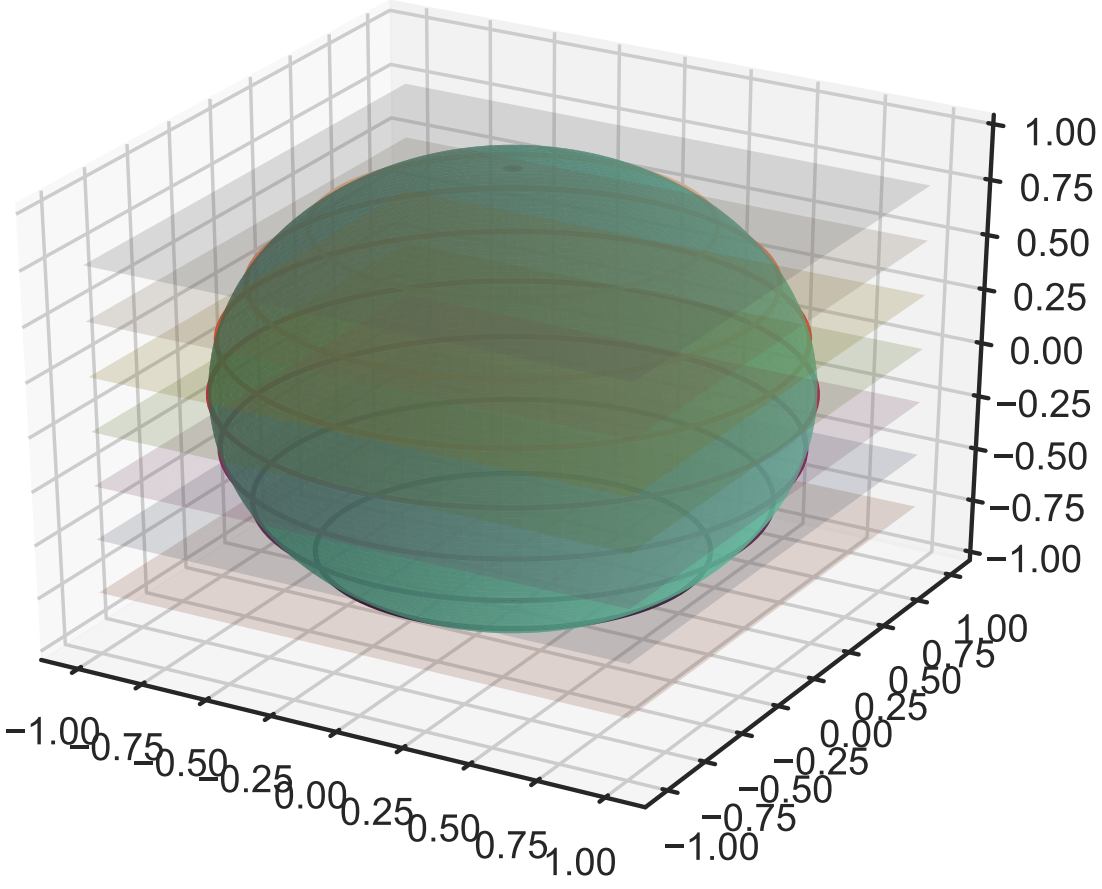
Given a population mean μ and a population variance σ^2, our ultimate goal is to sample from $\mathcal{N}(\mu, \sigma^2)$ in as efficient a way as possible. Towards this goal, we will need to know how to sample from $\mathcal{N}(\mu, \sigma^2)$ subject to constraints on the sample distribution we generate. Explicitly, we need to know how to generate k samples $x_1, x_2, \ldots x_k \sim \mathcal{N}(\mu, \sigma^2)$ subject to the constraints

$$\frac{1}{k}\sum_{i=1}^{k} x_i = \eta \tag{1}$$

$$\frac{1}{k}\sum_{i=1}^{k} (x_i - \eta)^2 = \delta^2. \tag{2}$$

With Equation 1 and Equation 2, we specify sample moments η and δ^2 which we require of our generated samples, the sample mean and sample variance respectively. Typically, we are interested in generating our samples such that $\eta = \mu$ and $\delta^2 = \sigma^2$ for unbiased estimation. Below, we describe a solution given by Marsaglia & Good to the above problem of generating Gaussian variates with given sample moments. This solution is attractive due to its simplicity, straightforwardness, and low computational overhead (also because it was given by Marsaglia, and we have to represent FSU). First, we give an intuitive overview of the algorithm.

Figure 1: Marsaglia's solution visualized for a fixed sample mean (geometrically interpreted as a sphere) and various sample variances (given by the planes). The solution for a given sample mean and variance is the circle of points at the intersection of the sphere with a plane.



2.1 Marsaglia's Solution

2.1.1 Intuition

Since the samples $x_1, \ldots, x_n \sim \mathcal{N}(\mu, \sigma^2)$ we wish to generate are independent, we can write their joint density function as

$$p(x_1, \ldots, x_n) = \left(2\pi\sigma^2\right)^{-n/2} e^{-1/2\sigma^2 \sum_{i=1}^{n}(x_i-\mu)^2}. \tag{3}$$

Thinking geometrically, we can envision the constraint in Equation 1 as defining a hyperplane in $\mathbb{R}^k$, and we can think of the constraint in Equation 2 as defining the surface of a hypersphere in $\mathbb{R}^k$. Therefore, by generating $x_1, \ldots, x_k$ satisfying the sample mean and sample variance conditions given by Equation 1 and Equation 2 respectively, we are actually trying to sample points that lie on the intersection of an k-dimensional hyperplane and an k-dimensional hypersphere, which would be a "circle" in $\mathbb{R}^k$ (actually a hypersphere in $k-1$ dimensions). Let this intersection be $\mathcal{X} \subseteq \mathbb{R}^k$. We can now crucially note that the joint density function is constant on $\mathcal{X}$. Indeed, for all $\mathbf{x} = (x_1, \ldots, x_k) \in \mathcal{X}$, we can observe

$$\begin{aligned}
\sum_{i=1}^{k}(x_i-\mu)^2 &= \sum_{i=1}^{k}(x_i-\eta)^2 + 2(\eta-\mu)\sum_{i=1}^{k}(x_i-\eta) + k(\eta-\mu)^2 \\
&= \sum_{i=1}^{k}(x_i-\eta)^2 + k(\eta-\mu)^2 \\
&= k\delta^2 + k(\eta-\mu)^2,
\end{aligned}$$

since $\sum_{i=1}^{k}(x_i-\eta) = \left(\sum_{i=1}^{k} x_i\right) - k\eta = k\eta - k\eta = 0$. Using this result, we can rewrite the density function in Equation 3 as

$$p(x_1, \ldots, x_k) = \left(2\pi\sigma^2\right)^{-k/2} e^{-1/2\sigma^2\left(k\delta^2 + k(\eta-\mu)^2\right)}, \tag{4}$$

which is independent of $x_1, \ldots, x_k$ and only dependent on $(\mu, \sigma^2, \eta, \delta^2)$, allowing us to conclude that the joint density function is constant on $\mathcal{X}$.

This important fact tells us that the probability distribution of $x_1, \ldots x_k$ conditional on satisfying the constraints $x_1, \ldots, x_k \in \mathcal{X}$ is just the uniform distribution on $\mathcal{X}$. Therefore, in order to generate our desired samples $x_1, \ldots, x_k$, we simply need to sample uniformly from the surface of a hypersphere in $\mathbb{R}^{k-1}$. More precisely, we can generate $\mathbf{x} = (x_1, \ldots, x_k) \in \mathcal{X}$ given a point $\mathbf{z} = (z_1, \ldots, z_k)$ on the surface of the $(k-1)$-dimensional unit sphere by solving the linear system

$$\mathbf{x} = \left(\sqrt{k}\delta\right)\mathbf{z}B + \eta\mathbf{v} \tag{5}$$

where $\mathbf{v} = (1, \ldots, 1) \in \mathbb{R}^k$ and $B \in R^{(k-1)\times k}$ is a matrix satisfying $BB^T = I$ and $B\mathbf{v}^T = 0$. By noting that $\|\mathbf{z}\|^2 = \mathbf{z}\mathbf{z}^T = 1$, we can easily verify that the solution to Equation 5, with these choices of $\mathbf{v}$ and B, satisfies the constraints on our sample moments:

$$\mathbf{x}\mathbf{v}^T = \sum_{i=1}^{k} x_i = k\eta$$

$$(\mathbf{x}-\eta\mathbf{v})(\mathbf{x}-\eta\mathbf{v})^T = k\delta^2\left(\mathbf{z}BB^T\mathbf{z}^T\right) = k\delta^2\left(\mathbf{z}\mathbf{z}^T\right) = k\delta^2$$

Note that we have written our vectors above horizontally instead of vertically, so when computing inner products and matrix-vector products, our transposes are in different locations in order to remain consistent with the standard definitions of the operations. We now provide a pseudocode description of the algorithm.

2.1.2 Pseudocode

In this algorithm, we assume that the desired sample mean is a random variable $\eta \sim \mathcal{N}(\mu, {}^{\sigma^2}/_{k})$ and that the desired sample variance is also a random variable satisfying $\frac{(k-1)\delta^2}{\sigma^2} \sim \chi^2_{k-1}$. We generate $\epsilon_1, \ldots, \epsilon_k \sim \mathcal{N}(0,1)$ using the Beasley-Springer-Moro algorithm.

Algorithm 1 MarsagliaSample

Input: Desired number of samples $k \in \mathbb{N}$
i.i.d. samples $\epsilon_1, \ldots, \epsilon_{k-1} \sim \mathcal{N}(0,1)$
Sample mean $\eta \sim \mathcal{N}(\mu, \frac{\sigma^2}{k})$
Sample variance δ^2 satisfying $\frac{(k-1)\delta^2}{\sigma^2} \sim \chi^2_{k-1}$
Output: $x_1, \ldots, x_k \sim \mathcal{N}(\mu, \sigma^2)$ with sample mean η and sample variance δ^2

$\gamma = \sqrt{k}\delta$
$s = \sum_{i=1}^{k} \epsilon_i^2$
for $i \in \{1, \ldots, k\}$ **do**
  $z_i = \epsilon_i((k-i)(k-i+1)s)^{-1/2}$
end for
$x_1 = (1-k)\gamma z_1 + \eta$
$x_k = \gamma \sum_{i=1}^{k-1} z_i + \eta$
for $i \in \{2, \ldots, k-1\}$ **do**
  $x_i = \gamma\left(\sum_{j=1}^{i-1}(j-k)z_j\right) + \eta$
end for
return $x_1, \ldots, x_k$

The above algorithm is only proven to produce an output with the desired properties if the sample mean η and sample variance δ^2 are *random variables* as described above. Therefore, if we attempt to naïvely use the algorithm with some deterministic values like $\eta = \mu$ and $\delta^2 = \sigma^2$, for example, we are not guaranteed to receive $x_1, \ldots, x_k$ with the desired sample moments.

To solve this problem, we generate η using the Beasley-Springer-Moro algorithm, which implements a numerical version of the Inverse Transformation method for the Gaussian CDF. We then compute δ^2 by first generating $\frac{(k-1)\delta^2}{\sigma^2}$, which needs to have the χ^2 distribution with $k-1$ degrees of freedom. In order to sample from that distribution, we use the fact that the L_2 norm squared of $k-1$ standard Normal variates is a χ^2 variate with $k-1$ degrees of freedom. Therefore, we generate $k-1$ standard normal variates using the Beasley-Springer-Moro algorithm, and then compute their squared L_2 norm, multiply by σ^2 and then divide by $k-1$ in order to obtain δ^2.

3 Antithetic Sampling

Now that we know how to generate $x_1, \ldots, x_k$ with given sample moments η and δ^2, we want to know how to reduce the variance in the sampling using antithetic variates. More precisely, given a set of k independent samples $x_1, \ldots, x_k \sim \mathcal{N}(\mu, \sigma^2)$, we want to generate a new set of k Gaussian samples $\tilde{x}_1, \ldots, \tilde{x}_k$ such that the combined sample moments of the two sample sets still satisfy $\frac{1}{2k}\sum_{i=1}^{k} x_i + \tilde{x}_i = \eta$ and $\frac{1}{2k}\sum_{i=1}^{k}(x_i-\mu)^2 + (\tilde{x}_i-\mu)^2 = \delta^2$. This second set of samples is *antithetic* to the original sample set.

This essentially all comes down to how to choose new sample moments $\tilde{\eta}$ and $\tilde{\delta}^2$ in such a way that they are "opposite" to η and δ^2, which we can accomplish through a clever use of the Inverse Transformation method of generating variates. Given a random variable X with CDF F_X, we

can obtain a random variable $Y = F_X(X)$ which is distributed uniformly on $[0, 1]$. We can then generate an *intermediate antithetic* variate $\tilde{Y} = 1 - Y$ which is also distributed uniformly on $[0, 1]$. Finally, we obtain a random variable *antithetic* to X by computing $\tilde{X} = F_X^{-1}(\tilde{Y})$ using the inverse CDF. Thus, we obtain X and $\tilde{X}$ marginally distributed identically, but with non-zero correlation.

For our sample moments, if we let F_η represent a Gaussian CDF and F_δ represent a Chi-squared CDf, we can derive $\tilde{\eta}$ and $\tilde{\delta^2}$ by computing

$$\tilde{\eta} = F_\eta^{-1}(1 - F_\eta(\eta)) \tag{6}$$

$$\frac{(k-1)\tilde{\delta^2}}{\sigma^2} = F_\delta^{-1}\left(1 - F_\delta\left(\frac{(k-1)\delta^2}{\sigma^2}\right)\right). \tag{7}$$

Therefore, we can easily derive a pseudocode for generating antithetic variates. However, the Gaussian and Chi-squared cumulative distribution functions are not generally easily invertible. Since η is normally distributed, computing the inverse CDF of η to simplifies to

$$\tilde{\eta} = 2 * \mu - \eta.$$

For the inverse Chi-squared CDF, we use a result given by Hawkins and Wixley, which was documented in a survey paper by Canal in 2005. This approach yields a closed form given by

$$\tilde{\lambda} = v\left(2\left(1 - \frac{3}{16v} - \frac{7}{512v^2} + \frac{231}{8192v^3}\right) - \left(\frac{\lambda}{v}\right)^{1/4}\right)^4$$

where $\lambda \sim \chi^2_v$ with $v = k - 1$ degrees of freedom. The pseudocode is then as follows.

3.1 Pseudocode

Algorithm 2 AntitheticSample

Input: i.i.d. samples $x_1, \ldots, x_k \sim \mathcal{N}(\mu, \sigma^2)$
Population mean and variance μ and σ^2
Output: $\tilde{x}_1, \ldots, \tilde{x}_k \sim \mathcal{N}(\mu, \sigma^2)$ with sample mean $\tilde{\eta}$ and sample variance $\tilde{\delta}^2$

$v = k - 1$
$\eta = \frac{1}{k}\sum_{i=1}^{k} x_i$
$\delta^2 = \frac{1}{k}\sum_{i=1}^{k}(x_i - \mu)^2$
$\tilde{\eta} = 2\mu - \eta$
$\lambda = {}^{v\delta^2}/_{\sigma^2}$
$\tilde{\lambda} = v\left(2\left(1 - \frac{3}{16v} - \frac{7}{512v^2} + \frac{231}{8192v^3}\right) - \left(\frac{\lambda}{v}\right)^{1/4}\right)^4$
$\tilde{\delta}^2 = {}^{\tilde{\lambda}\sigma^2}/_{v}$
$\tilde{x}_1, \ldots, \tilde{x}_k =$ MarsagliaSample$(k, \tilde{\eta}, \tilde{\delta}^2)$
return $\tilde{x}_1, \ldots, \tilde{x}_k$

4 Numerical Experiments

To test how these algorithms work empirically, we compared the performance of Algorithm 1 to Algorithm 2 when tasked with sampling from a standard normal distribution using different numbers of samples k. The algorithms are compared to the "true" standard normal distribution, as given by the `norm.pdf()` function in SciPy's `stats` package.

The data sets labelled MarsagliaSample are computed using $2k$ samples by using Algorithm 1 with $2k$ set as the number of desired samples. The data sets labelled AntitheticSample are computed by first generating k samples using Algorithm 1, and then using those as inputs to produce k more samples from Algorithm 2. In either case, the resulting graphs are generated by fitting a Gaussian curve to the raw data produced by the algorithms. The `norm.pdf()` function was again used to fit those curves. The results are visualized below.

Figure 2: $k = 8$ samples

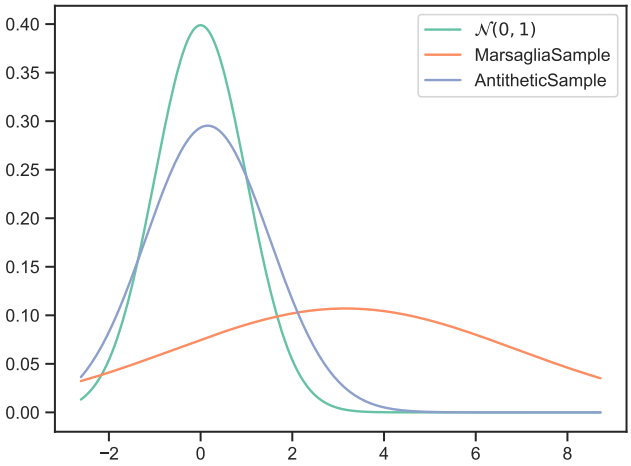


Figure 3: $k = 10$ samples

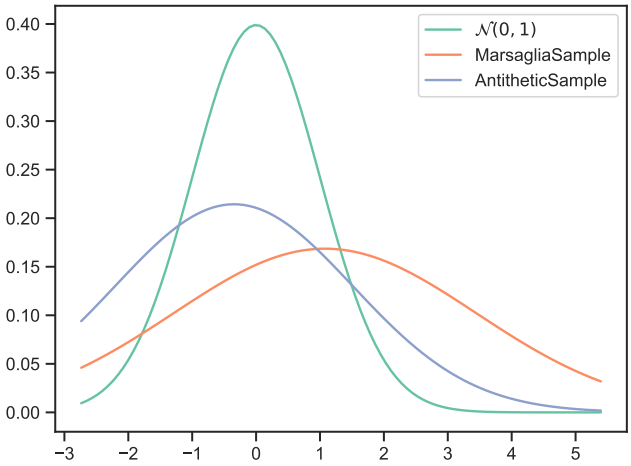
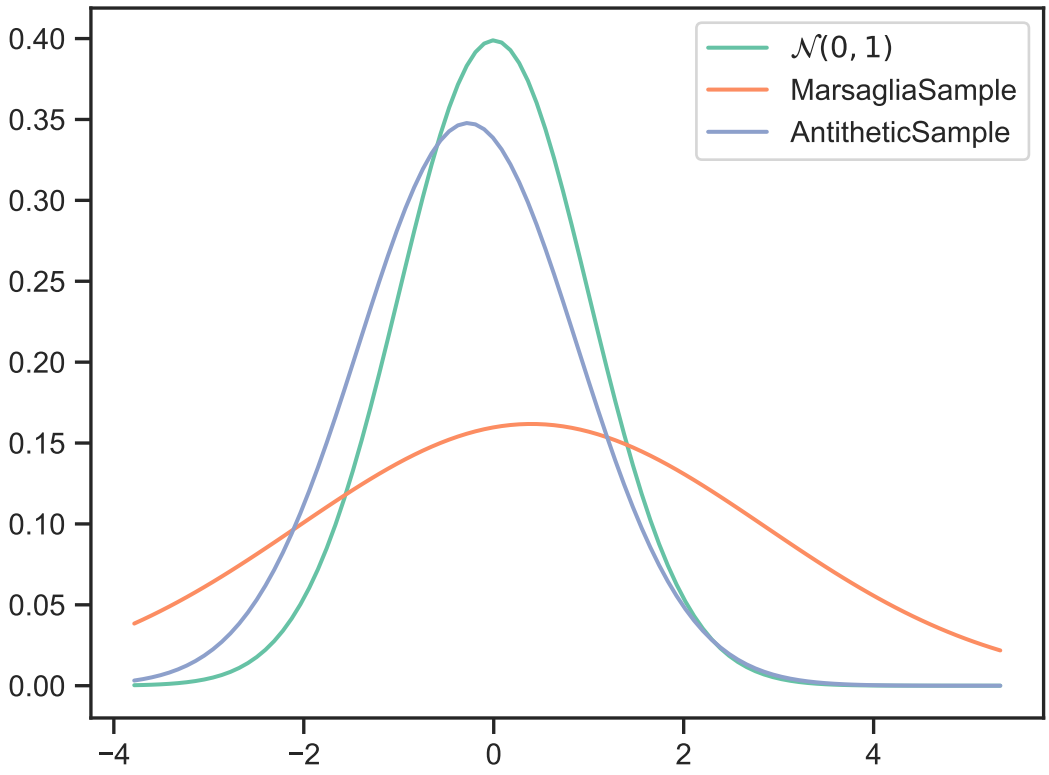


Figure 4: $k = 20$ samples



5 Conclusion

As we can see by the figures above, the antithetic approach of synthesizing the i.i.d. samples produced by Algorithm 1 with the antithetic variates produced by Algorithm 2 improves significantly upon the naïve approach that does not rely on antithetic variates. Even with as few as $k = 8$ regular and $k = 8$ antithetic variates generated, we already obtain a fairly faithful approximation to the standard normal curve, whereas the method which produces $k = 16$ variates without antithetic synthesis is clearly way off the mark. The situation improves mildly for the non-antithetic method with $k = 40$ samples, but it is still heavily outperformed by the method which synthesizes $k = 20$ regular and $k = 20$ antithetic samples.

References

- G. Marsaglia and I. Good. C69. Generating a normal sample with given sample mean and variance. *Journal of Statistical Computation and Simulation*, 11 (1):71-74, 1980.
- L. Canal. A normal approximation for the chi-square distribution. *Computational Statistics & Data Analysis*, 48(4):803-808, 2005.
- D. M. Hawkins and R. Wixley. A note on the transformation of chi-squared variables to normality. *The American Statistician*, 40(4):296-298, 1986.