Unfavorable SNR in Forward Gradients

Charles Frye

February 2022

The forward gradient method of [1] allows the computation of approximate gradients without the need for a separate "backwards pass" to compute gradients.

As a rule of thumb, the backwards pass has the same computational complexity, in the big-$\mathcal{O}$ sense, as the forwards pass, and so doesn't change the complexity class of optimization algorithms that require forwards passes. The backwards pass typically incurs a fixed (low) multiplicative cost relative to forward computation and occasionally requires a large amount of memory, and so there can be meaningful practical benefits to avoiding it.

There is, however, a catch to the forward gradient method that is not clearly stated in [1]: the signal-to-noise ratio (SNR) using the forward gradient method is at most $\mathcal{O}\left(1/n\right)$, where n is the number of parameters.

High SNR is good, and so this bound is bad news. Noise increases the effective Lipschitz constant and so decreases the highest stable learning rate. Lower learning rates mean slower convergence (in iteration count), which can dwarf any gains from faster iterations. Specifically, this bound implies that for any model, once n is large enough, descent by forward gradients will converge more slowly in wall time than will backward gradients[1].

We therefore should expect (and do observe) poorer performance when directly substituting the forward gradient method for the more typical backwards gradient. Perhaps additional tricks, like gradient clipping or a special optimizer, might make the forward gradient method more applicable in more settings.

Specifically, denoting the (full) gradient ∇ and its forward estimate g, we have that the ith entry of g has variance:

$$\mathbb{V}\left[g\right]_i \approx 4\|\nabla\|^2 \tag{1}$$

Assuming that the gradient ∇ has $\mathcal{O}\left(1\right)$ norm and homogeneous entries, we can obtain an asymptotic, in parameter count, expression for the SNR:

$$\frac{\nabla_i^2}{\mathbb{V}\left[g\right]_i} \approx \frac{1/n}{4\|\nabla\|^2} = \mathcal{O}\left(1/n\right) \tag{2}$$

[1] Space constraints might still favor forward gradients.

The expression for the variance arises fundamentally because each entry in g is a sum of products of normally-distributed variables – that is, a χ^2-distributed variable.

Forward Gradient Descent

Let ∇ be the full-batch gradient. We calculate an estimator g as in the forward gradient method [1]:

$$g \stackrel{\text{def}}{=} \left(\nabla^\top v\right) \cdot v \tag{3}$$

where $v \sim \mathcal{N}(0, I)$ is a random Gaussian vector of length n. Effectively, the forward gradient method produces a biased random walk, taking larger steps when the randomly chosen vector v happens to align with the gradient. $\nabla^\top v$ can be calculated efficiently as part of the same forward pass that computes the loss, as it is a Jacobian-vector product.

It is proven in [1] that the expectation of g is ∇. This is sufficient to prove convergence of the algorithm by any number of established means, treating the forward gradient as the (noisy) gradient oracle.

But what about the variance?

We can break down the variance of the ith entry of g as

$$\begin{aligned}
\mathbb{V}\left[g_i\right] &= \mathbb{V}\left[\left(\nabla^\top v\right) \cdot v_i\right] && \text{(Def'n of } g\text{)} \\
&= \mathbb{V}\left[\sum_j v_j v_i \nabla_j\right] && \text{(Def'n of } ^\top\text{)} \\
&= \mathbb{V}\left[v_i^2 \nabla_i\right] + \mathbb{V}\left[\sum_{j \neq i} v_j v_i \nabla_j\right] + 2 \cdot \operatorname{Cov}\left(v_i^2 \nabla_i, \sum_{j \neq i} v_j v_i \nabla_j\right) \\
& && \text{(Variance of sum formula)}
\end{aligned}$$

Notice that the variance of a single entry g_i depends on a (random) sum over all of the entries.

The covariance term is 0:

$$\begin{aligned}
\operatorname{Cov}\left(v_i^2 \nabla_i, \sum_{j \neq i} v_j v_i \nabla_j\right) &= \mathbb{E}\left[v_i^2 \nabla_i \cdot \sum_{j \neq i} v_j v_i \nabla_j\right] - \mathbb{E}\left[v_i^2 \nabla_i\right] \mathbb{E}\left[\sum_{j \neq i} v_j v_i \nabla_j\right] \\
& && \text{(Def'n of covariance)} \\
&= \mathbb{E}\left[v_i^2 \nabla_i \cdot \sum_{j \neq i} v_j v_i \nabla_j\right] && \left(\mathbb{E}\left[v_i^2\right] = 0\right) \\
&= \nabla^\top \mathbb{E}\left[v^2\right] && \text{(Def'n of } ^\top\text{, Linearity of } \mathbb{E}\text{)} \\
&= 0 && \left(\mathbb{E}\left[v_j^2\right] = 0\right)
\end{aligned}$$

and so we have:

$$\mathbb{V}\left[g_i\right] = \mathbb{V}\left[v_i^2 \nabla_i\right] + \mathbb{V}\left[\sum_{j \neq i} v_j v_i \nabla_j\right] \tag{4}$$

We tackle the simpler term first:

$$\mathbb{V}\left[v_i^2 \nabla_i\right] = \nabla_i^2 \, \mathbb{V}\left[v_i^2\right] \qquad \text{(Scaling rule for } \mathbb{V}\text{)}$$

and notice that v_i is a unit normal random variable. Products of unit normals follow distributions from the χ^2 family. These distributions are indexed by the degree-of-freedom parameter k and their variances are given by $2k$. In this case, $k = 1$ and so the variance is 2.

$$\mathbb{V}\left[v_i^2 \nabla_i\right] = 2 \cdot \nabla_i^2 \tag{}$$

Now we tackle the sum term:

$$\mathbb{V}\left[\sum_{j \neq i} v_j v_i \nabla_j\right] = \sum_{j \neq i} \mathbb{V}\left[v_j v_i \nabla_j\right] + 2 \cdot \sum_{j \neq i \neq k,\ j < k} \operatorname{Cov}\left(v_j v_i \nabla_j, v_k v_i \nabla_k\right)$$

(Variance of sum formula)

Switching things up this time, we try the harder term first and peek into the covariances in the sum:

$$\begin{aligned}
\operatorname{Cov}\left(v_j v_i \nabla_j, v_k v_i \nabla_k\right) &= \nabla_j \nabla_k \operatorname{Cov}\left(v_j v_i, v_k v_i\right) && \text{(Bilinearity of the covariance)} \\
\operatorname{Cov}\left(v_j v_i, v_k v_i\right) &= \mathbb{E}\left[v_j v_i v_i v_k\right] + \mathbb{E}\left[v_j v_i\right] \cdot \mathbb{E}\left[v_k v_i\right] && \text{(Def'n of the covariance)} \\
&= \mathbb{E}\left[v_j\right] \mathbb{E}\left[v_i v_i\right] \mathbb{E}\left[v_k\right] + \mathbb{E}\left[v_j\right] \mathbb{E}\left[v_i\right] \cdot \mathbb{E}\left[v_k\right] \mathbb{E}\left[v_i\right] && \text{(Product rule for } \mathbb{E}\text{)} \\
&= 0 && \text{(Expectations of } v_i\text{, } v_j\text{, and } v_k\text{)}
\end{aligned}$$

That is, the covariance terms are all identically 0,[2] so we can return our focus to the simpler term in the variance of the sum of products:

$$\begin{aligned}
\mathbb{V}\left[\sum_{j \neq i} v_j v_i \nabla_j\right] &= \sum_{j \neq i} \mathbb{V}\left[v_j v_i \nabla_j\right] + 0 && \text{(From above)} \\
&= \sum_{j \neq i} \nabla_j^2 \, \mathbb{V}\left[v_j v_i\right] && \text{(Scaling rule for } \mathbb{V}\text{)} \\
&= \sum_{j \neq i} \nabla_j^2 \cdot 4 && (v_j v_i \sim \chi^2(2))
\end{aligned}$$

[2]Intuition: knowing the value of $v_j v_i$, e.g. that it is far from its mean, does help you guess the value of $v_k v_i$, e.g. that it will be far from its mean, so the variables are dependent. But knowing which direction $v_j v_i$ has deviated from its mean doesn't help you guess which direction $v_k v_i$ has deviated from its mean, since the signs of v_j and v_k could differ or match with equal probability, and therefore the variables are uncorrelated.

where the final line is obtained by again noticing a product of normals, aka χ^2, this time with 2 degrees of freedom and so a variance of $2 \cdot 2 = 4$.

We can now combine our two expressions for the variance terms in the sum-of-variance decomposition

$$\mathbb{V}\left[g_i\right] = \mathbb{V}\left[v_i^2 \nabla_i\right] + 4 \cdot \sum_{j \neq i} \nabla_j^2 \tag{5}$$

When n is large, the $i \neq j$ sum predominates, and so we can introduce the approximation[3]

$$\mathbb{V}\left[g_i\right] = \mathbb{V}\left[v_i^2 \nabla_i\right] + 4 \cdot \sum_{j \neq i} \nabla_j^2 \tag{6}$$

$$\approx 4 \cdot \sum_{j} \nabla_j^2 = 4\|\nabla\|^2 \tag{7}$$

Is this so bad? At first glance, things look fine – the variance is finite and it's on the scale of the gradient norm, which is often manipulated to be $\mathcal{O}(1)$ to obtain favorable numerics in optimization.

But consider what this means for the signal-to-noise ratio: if the gradient norm is of fixed size, the entries must be getting smaller as more parameters are added to the model. If they are homogeneous, each ∇_i should be of absolute value around $\frac{1}{\sqrt{n}}$, and so we obtain

$$\frac{\nabla_i^2}{\mathbb{V}\left[g\right]_i} \approx \frac{1/n}{4\|\nabla\|^2} = \mathcal{O}\left(1/n\right) \tag{8}$$

as desired.

Commentary

The proof above, like [1], assumes full-batch gradients, rather than mini-batch gradients. Experiments and rough calculations suggest that the problem of forward gradient variance becomes worse in the stochastic setting.

We note that, as a product of normals, g has unfavorable characteristics in its higher moments, not just in its variance (or second moment). In simulation, we observed highly kurtotic (heavy-tailed) distributions with prominent peaks at 0.

There are two intuitions for why this is the case: one at the level of individual entries and one at the level of the entire vector. When the randomly drawn value of v_i is small, the gradient value is multiplied by the much smaller value v_i^2 and

[3] Upon reflection, our final result looks very similar to the result for a $\chi^2(2)$, but generalized to a multivariate, rather than univariate, normal. The variance is $4 = 2 \cdot k$ and we scale it by the gradient norm. This suggests a much more direct, perhaps index-free, method of proof.

vice versa when the randomly drawn value of v_i is large. Alternatively, as the dimension of the space increases, two randomly chosen vectors become increasingly orthogonal. The inner product $\nabla^\top v$ thus has a very sparse distribution and the vectors $\nabla^\top v \cdot v$ have norms with a very right-skew distribution.

Occasional large parameter updates produced by heavy-tailed noise can cause numerical issues for gradient descent algorithms, e.g. by throwing the parameters into an area of the function where the local Lipschitz constant is much higher, turning a convergent learning rate into a divergent one.

These distributional issues suggest that an alternative distribution for the entries of v might help speed up forward gradient. Without a priori information about the gradient vector, the isotropy of the unit normal is a helpful property; an anisotropic distribution might be even less likely to produce vectors v that . Note also that the proof of unbiasedness in [1] requires that the components of the vector be independent, and the unit normal is the only isotropic distribution with independent components.

One useful form of anisotropic distribution would be to produce probing vectors v that are more likely to point in the direction of the gradient. Indeed, if the distribution of v is a Dirac delta at ∇, the variance of g drops to 0 and forward gradients align perfectly with true gradients. Of course, in that case there is no need to compute a gradient estimate, because the gradient is already at hand, but perhaps there is a happy medium between complete ignorance and complete knowledge of the gradient where the forward gradient becomes optimal.

References

[1] Atılım Güneş Baydin, Barak A. Pearlmutter, Don Syme, Frank Wood, and Philip Torr. Gradients without backpropagation, 2022.