WORKING DRAFT
Bit Centering: A Technical Report

Christopher R. Aberger[2], Christopher De Sa[1], and Christopher Ré[2]
[1] Cornell University, Ithaca, NY
[2] Stanford University, Stanford, CA

July 7, 2018

Abstract

The motivation for this work is the rise of accelerators for analytics, in which a host machine with large memory but limited compute uses a high-performance accelerator that is capable of performing many low-precision operations cheaply. This technical report describes techniques to cope with low-precision inference and learning based on a novel *bit centering technique*. The key idea is to gracefully tradeoff representation using host-memory and an accelerator. Applications to learning are described including novel convergence rates. This allows one to train high-precision models by off-loading as much as possible to the low-precision accelerator. **Please note this draft is a working draft and may be revised without notice.**

1 Introduction

The technical setting for this technical report note is the recent resurgence of hardware accelerators for machine learning and other applications.[1] In such a scenario, there is a host device with a larger memory and general purpose processor (e.g., a CPU). The accelerator typically has many more computational units of lower precision and smaller memory. In turn, this motivates the use of low-precision arithmetic. We think of the accelerator as working on the inner loop of the computation and the host computation as infrequently (orders of magnitude less frequently) performing some high-precision computation to help direct (or center) the low-precision computation on the accelerator. This technical report describes a technique that uses limited amounts of host memory and computation to improve the low-precision computation of an accelerator.

As we will see, the conventional wisdom is that training algorithms using low-precision iterations are limited to low accuracy. Our goal is to formalize situations in which a modest amount of host memory allows us to achieve dramatically better error with minimal overhead.

The Core Idea: Bit Centering Bit centering is a technique in which we represent each number as the sum of a pre-computed full-precision *offset* and a low-precision *delta*:

$$x = o_x + \delta_x.$$

The benefit of bit centering, as compared with simply storing the number x itself in low-precision, comes from the fact that floating point computation has *relative* error. That is, if we do some computation in low precision that is intended to approximate some number $x \in \mathbb{R}$, and the result of that computation is a floating point number $\tilde{x}$, then

$$|\tilde{x} - x| \leq \epsilon |x|,$$

[1] The authors have a financial interest in SambaNova Systems, which produces machine learning and big data platform including hardware accelerators and an accelerated software stack.

where ϵ is some fixed-but-small number that is a function of the number of bits used. In comparison, if we compute only the delta in low-precision, using bit centering, the error will look like

$$|\tilde{x} - x| = |\tilde{\delta}_x - \delta_x| \leq \epsilon|\delta_x| = \epsilon|x - o_x|.$$

So if we can choose o_x so that it will be close to x (equivalently, so that δ_x will be small), then bit centering can produce a much more accurate estimate of x using the same number of bits.

In this document, we will describe bit centering, and compare it with other quantization strategies, when used to compute gradients for learning algorithms. The quantization strategies are:

- **High-precision.** Compute the whole algorithm in high-precision. This is the null quantization strategy.
- **Quantize last.** Compute the gradient in high-precision floating point, then quantize to low-precision floating point at the end. This is sometimes called *gradient quantization*.
- **Quantize first.** Quantize all the inputs to the gradient procedure into low-precision floating point, and compute the gradient in low-precision. This quantizes all operations, but can cause accuracy to suffer.
- **Bit centering.** Compute the gradient using bit-centered numbers with a pre-computed-on-the-host high-precision offset and a computed-on-the-accelerator low-precision floating-point delta.
- **Block floating point bit centering.** The same as bit centering, but using block-floating-point numbers (which share an exponent) for the delta rather than floating-point numbers. This lets us save memory and compute that would otherwise be used to store and process each number's exponent individually.

1.1 Notation

We will use $\tilde{x}$ to denote an approximation. We will follow the notation of Trefethen and Bau III [3] and their error model. Namely, given a number x and a floating point number system with machine error $\varepsilon_{\text{machine}}$, then if $\mathsf{fl}(x)$ is the floating point representation of x, there will exist an ϵ such that $|\epsilon| \leq \varepsilon_{\text{machine}}$ and:

$$\mathsf{fl}(x) = (1 + \varepsilon)x$$

Moreover, if $*$ denotes any basic operation in scalar arithmetic (but also fused-multiply and add on two floating point representations), and $\circledast$ denotes its floating point analogue, then for some ϵ such that $|\epsilon| \leq \varepsilon_{\text{machine}}$

$$x \circledast y = (1 + \varepsilon)(x * y).$$

This is sometimes called the fundamental theorem of floating point arithmetic. In this note, we will deal with just two floating point number systems to make our life easier: (1) a **low precision** number system with machine precision $\varepsilon_{\text{machine-lo}}$ and (2) a **high precision** number system with machine precision $\varepsilon_{\text{machine-hi}}$. We let the low-precision representation of a number x be denoted $\mathsf{fl}_{\text{lo}}(x)$ and the high-precision representation be denoted $\mathsf{fl}_{\text{hi}}(x)$. In the analysis that follows, we will use ε_{lo} to generally denote a value that satisfies $\varepsilon_{\text{lo}} \leq \varepsilon_{\text{machine-lo}}$ and ε_{hi} to generally denote a value that satisfies $\varepsilon_{\text{hi}} \leq \varepsilon_{\text{machine-hi}}$. We also define the precision ration k as

$$k = \frac{\varepsilon_{\text{machine-lo}}}{\varepsilon_{\text{machine-hi}}}.$$

Intuitively, we can imagine that the high-precision representation is a 32-bit floating point number, and the low-precision representation is an 8-bit float.

2 Case Study: Dot Product

To better understand the error improvement tradeoff of splitting versus full precision and low precision in host memory, we consider computing a dot product $x^T y$ for $x, y \in \mathbb{R}^n$.

To simplify the discussion, we will consider bit recentering on x to start. That is, we write $x = o_x + \delta_x$. The idea is that o_x will be stored in high precision on the host and we can do limited operations on it, while δ_x is stored in low precision on the accelerator.

Consider computing $x^T y$ where $x = o_x + \delta_x$ with x and y in $\mathbb{R}^n_+$ (both stored in high precision) and storing the result in low precision (i.e. the floating point representation with machine error $\varepsilon_{\text{machine-lo}}$). We are going to evaluate three methods of computing this expression: *quantize last*, which computes the dot product in high precision and quantizes at the end; *quantize first*, which quantizes x and y into low precision and then computes the dot product in low precision; and *bit centering*. Our results in this section are summarized in the following table, which also compares to *high-precision*, a strategy which just computes the dot product in high precision and stores the result in high precision as well.

In the derivations that follow, we will use the well-known fact [1] that for a nonnegative floating-point dot product with machine error $\varepsilon_{\text{machine}}$, the result of the computation will be, for some $|\varepsilon| \leq \varepsilon_{\text{machine}}$,

$$\mathsf{fl}(x)^T \mathsf{fl}(y) = x^T y + |x|^T |y| n\varepsilon$$

where $|x|$ denotes the absolute value of x applies entrywise, and where we are ignoring terms proportional to ε^2. (Note that this is for using naive summation; if we used pairwise summation the n would be replaced by a $\log n$ factor.) For non-negative x and y, this is equivalent to

$$\mathsf{fl}(x)^T \mathsf{fl}(y) = x^T y(1 + n\varepsilon),$$

a fact we use in the subsequent derivation.

Quantization strategy	**# LP Ops**	**# HP Ops**	**Result**
High-precision	—	n multiplies $n-1$ adds	$(x^T y)\,(1 + \varepsilon_{\text{hi}} \cdot n)$
Quantize last	—	n multiplies $n-1$ adds	$(x^T y)\,(1 + \varepsilon_{\text{hi}} \cdot (n+k))$
Quantize first	n multiplies $n-1$ adds	—	$(x^T y)(1 + \varepsilon_{\text{hi}} \cdot kn)$
Bit centering	n multiplies n adds	n multiplies $n-1$ adds precomputed on host	$(x^T y)\,(1 + \varepsilon_{\text{hi}} \cdot (n+k))$ $+(\delta_x^T y)\,(\varepsilon_{\text{hi}} \cdot (kn + n + 2k + 1))$ $+(x^T y) \cdot \varepsilon_{\text{hi}} \cdot k$

Quantize last We compute the dot product first in high precision, producing an intermediate value which will satisfy

$$\mathsf{fl}_{\text{hi}}(x)^T \mathsf{fl}_{\text{hi}}(y) = x^T y \cdot (1 + \varepsilon_{\text{hi}} \cdot n).$$

If we then quantize that high-precision intermediate value to low precision, we will get

$$\begin{aligned}
\mathsf{fl}_{\text{lo}}(\mathsf{fl}_{\text{hi}}(x)^T \mathsf{fl}_{\text{hi}}(y)) &= (x^T y)(1 + \varepsilon_{\text{hi}} \cdot n)(1 + \varepsilon_{\text{lo}}) \\
&= (x^T y)\left(1 + \varepsilon_{\text{hi}} \cdot \left(n + \frac{\varepsilon_{\text{machine-lo}}}{\varepsilon_{\text{machine-hi}}}\right)\right) \\
&= (x^T y)\,(1 + \varepsilon_{\text{hi}} \cdot (n+k))\,.
\end{aligned}$$

Note again that we drop terms that are $O(\epsilon^2)$, as is standard in Trefethen and Bau III [3]. While this is a small relative error, all of the computation we did here was in high precision. Can we do better?

Quantize first If we first quantize to low precision and then do the multiplication, then

$$\mathsf{fl}_{\text{lo}}(x)^T \mathsf{fl}_{\text{lo}}(y) = (x^T y)(1 + \varepsilon_{\text{lo}} \cdot n) = (x^T y)(1 + \varepsilon_{\text{hi}} \cdot kn)$$

That is, we suffer the extra error *multiplicatively*, rather than additively, as we did in the quantize-last case. Can we do better? Our goal is to smoothly interpolate between the two approaches.

Hybrid method: bit centering with an offset Let's compare this with bit centering, which uses an offset. Here, we decompose $x = o_x + \delta_x$ and represent o_x in high precision and δ_x in low-precision. So part of the dot product is computed in high precision and rounded (the offset), and part is computed in low precision (the delta). We expect that if the delta is small, this algorithm will have lower error. Let's upper bound the error of this algorithm:

$$\begin{aligned}
&\mathsf{fl}_{\text{lo}}(\mathsf{fl}_{\text{hi}}(o_x)^T \mathsf{fl}_{\text{hi}}(y)) + \mathsf{fl}_{\text{lo}}(\delta_x)^T \mathsf{fl}_{\text{lo}}(y) \\
&\quad = \left((o_x^T y)\,(1 + \varepsilon_{\text{hi}} \cdot (n + k)) + (\delta_x^T y)(1 + \varepsilon_{\text{hi}} \cdot kn)\right)(1 + \varepsilon_{\text{lo}}) \\
&\quad = (o_x^T y)\,(1 + \varepsilon_{\text{hi}} \cdot (n + 2k)) + (\delta_x^T y)(1 + \varepsilon_{\text{hi}} \cdot k(n + 1)) \\
&\quad = ((x - \delta_x)^T y)\,(1 + \varepsilon_{\text{hi}} \cdot (n + 2k)) + (\delta_x^T y)(1 + \varepsilon_{\text{hi}} \cdot k(n + 1)) \\
&\quad = (x^T y)\,(1 + \varepsilon_{\text{hi}} \cdot (n + k)) + (\delta_x^T y)\,(\varepsilon_{\text{hi}} \cdot (kn + n + 2k + 1)) + (x^T y) \cdot \varepsilon_{\text{hi}} \cdot k
\end{aligned}$$

This formula smoothly interpolates between the two above formulae. We've written this so that the first term is identical to quantize last, the second term is the quantize first error, and the third term is a small extra error from doing the extra addition to add the offset and delta components together. Since the quantize last term is the best we could hope for, we can view the second and third terms as additional error. When δ_x is small enough this approach improves error—as we suspected. In contrast, if δ_x is large and the offset is 0 we suffer like quantize first, and suggests that when δ_x is small the hybrid strategy is likely to provide an improvement.

A difference of this approach from quantize last is that we pre-compute $\mathsf{fl}_{\text{lo}}(\mathsf{fl}_{\text{hi}}(o_x)^T \mathsf{fl}_{\text{hi}}(y))$ on the host and then load it into memory. In terms of accelerator compute, it costs at worst a $2/n$ additional relative computation time over quantize first (since we need to do an extra add and an extra load from the host, compared with quantize first). This may save compute overall if we are evaluating dot products with y for multiple different values of δ_x, but the same offset o_x.

3 Learning Using Bit Centering

We want to use bit centering for SGD and SGD-like algorithms. To do this, we need to compute gradients using bit centering. That is, we need to apply bit centering to compute things like $\nabla f_i(w)$. First, we need to decide how to assign the offset. One natural way to do this is to choose some o_w that we believe is close to w, and let o_w be the offset for storing w. That is, we store w using bit centering as

$$w = o_w + \delta_w.$$

If we do this, the natural way to assign the offset for $\nabla f_i(w)$ is to assign $\nabla f_i(o_w)$. Then the gradient, represented using bit centering, will look like

$$\nabla f_i(w) = \nabla f_i(o_w) + \left(\nabla f_i(o_w + \delta_w) - \nabla f_i(o_w)\right),$$

where the term on the left is the offset and the term on the right is the delta. To compute gradients using bit centering, we would pre-compute $\nabla f_i(o_w)$ in high precision for some o_w, and then compute $\nabla f_i(o_w + \delta_w) - \nabla f_i(o_w)$ using low-precision arithmetic. For bit centering to be effective we need to compute this in such a way that our approximation satisfies a good relative error bound similar to the one we had for the dot product example. However, if we just compute this naively using low-precision arithmetic by computing $\nabla f_i(o_w + \delta_w)$ and $\nabla f_i(o_w)$ and then subtracting them, we will observe *catastrophic cancellation* resulting in a loss of precision for the delta component, because $\nabla f_i(o_w + \delta_w)$ and $\nabla f_i(o_w)$ are very close to each other and cancel out to a value near 0 when δ is small. In order to prevent this

loss of precision, we need to compute $\nabla f_i(o_w + \delta_w) - \nabla f_i(o_w)$ in another way, one that does not involve computing the terms independently and subtracting them. This "other way" must depend on the computational structure of the functions f_i. Next, we will present ways that we can do this, first for simple linear-model objectives, and then for more general objectives, such as deep neural nets.

3.1 Gradient Estimation for Linear Models

A regularized linear model is a model with loss of the form

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} l_i(w^T x_i) + \frac{1}{2}\sigma\|w\|^2$$

where $x_i \in \mathbb{R}^d$ is the training example, $w \in \mathbb{R}^d$ are the weights, $l_i : \mathbb{R} \to \mathbb{R}$ is the component loss function, d is the number of dimensions, and σ is the regularization parameter. Examples of linear model problems include linear regression and logistic regression. For this class of problems, the individual component functions can be written simply as

$$f_i(w) = l_i(w^T x_i) + \frac{1}{2}\sigma\|w\|^2,$$

and the gradient of this is

$$g = f_i'(w) = l_i'(w^T x_i)x_i + \sigma w.$$

It is straightforward to imagine computing this using the quantize-first and quantize-last strategies, by simply doing all the computations in low-precision or high-precision, respectively. To be concrete, in the quantize-last strategy, we do

$$\tilde{g} = \mathsf{fl}_{\mathrm{lo}}(l_i'(\mathsf{fl}_{\mathrm{hi}}(w)^T \mathsf{fl}_{\mathrm{hi}}(x_i))\mathsf{fl}_{\mathrm{hi}}(x_i) + \mathsf{fl}_{\mathrm{hi}}(\sigma)\mathsf{fl}_{\mathrm{hi}}(w))$$

and in the quantize-first strategy, we do

$$\tilde{g} = l_i'(\mathsf{fl}_{\mathrm{lo}}(w)^T \mathsf{fl}_{\mathrm{lo}}(x_i))\mathsf{fl}_{\mathrm{lo}}(x_i) + \mathsf{fl}_{\mathrm{lo}}(\sigma)\mathsf{fl}_{\mathrm{lo}}(w).$$

On the other hand, if we compute this gradient using bit centering as described above for $w = o_w + \delta_w$, the offset will be

$$o_g = l_i'(o_w^T x_i)x_i + \sigma o_w$$

and the delta will be

$$\begin{aligned}\delta_g &= \left(l_i'(w^T x_i)x_i + \sigma w\right) - \left(l_i'(o_w^T x_i)x_i + \sigma o_w\right) \\ &= \left(l_i'(o_w^T x_i + \delta_w^T x_i) - l_i'(o_w^T x_i)\right) x_i + \sigma\delta_w.\end{aligned}$$

We can compute this in low-precision without loss of precision by using the following algorithm. To do this, we make the following assumptions. First, we assume that our floating point computation is accurate enough to compute $l_i'(u)$ for any u up to machine precision. That is, for any u, and for either high or low precision,

$$l_i'(\mathsf{fl}(u)) = (1+\varepsilon)l_i'(u).$$

Second, we assume that we have the ability to approximate scalar functions accurately using low-precision arithmetic. Given any "sufficiently nice" scalar function $h : \mathbb{R} \to \mathbb{R}$ with $h(0) = 0$ and any range $[-a, a]$ about zero, we suppose that we can (via some amount of high-precision computation) produce a low-precision-computable routine $\tilde{h}$ that can compute $h(x)$ for any low-precision number $x \in [-a, a]$ to within relative accuracy

$$\left|\tilde{h}(x) - h(x)\right| \leq \zeta \cdot |x| \cdot \max_{u\in[-a,a]}\left|\frac{h(u)}{u}\right| \cdot \varepsilon_{\text{machine-lo}}$$

for some constant $\zeta \geq 1$. Methods for doing this include low-precision Chebyshev approximation and linear interpolation: any desired method may be used, as is appropriate for the accelerator hardware. We further assume that l_i' is part of the class of functions that are sufficiently nice for this approximation method to work. Third, we assume that

l_i' is Lipschitz continuous with parameter L. Finally, we assume that we know some bound on δ_w, and can guarantee that $\|\delta_w\| \leq R$. Now the algorithm.

Algorithm 1 Bit Centering for Linear Models

given: loss function l, training example x, and delta maximum range R

high-precision precomputation on host:
given: offset o_w
$\phi \leftarrow \mathsf{fl}_{\mathrm{hi}}(o_w)^T \mathsf{fl}_{\mathrm{hi}}(x)$
$a \leftarrow R \cdot \|\mathsf{fl}_{\mathrm{hi}}(x)\|$
define: $h(\psi) = l'(\phi + \psi) - l'(\phi)$
compute LP function approximation: $\tilde{h} \approx h$ in range $[-a, a]$
store: $\tilde{h}$ and $\mathsf{fl}_{\mathrm{lo}}(x)$ and $\mathsf{fl}_{\mathrm{lo}}(\sigma)$
output: $o_g = l'(\phi) \cdot \mathsf{fl}_{\mathrm{hi}}(x) + \mathsf{fl}_{\mathrm{hi}}(\sigma) \cdot \mathsf{fl}_{\mathrm{hi}}(o_w)$

low-precision computation on accelerator:
given: delta δ_w
load: $\tilde{h}$ and $\mathsf{fl}_{\mathrm{lo}}(x)$ and $\mathsf{fl}_{\mathrm{lo}}(\sigma)$
$\psi \leftarrow \mathsf{fl}_{\mathrm{lo}}(\delta_w)^T \mathsf{fl}_{\mathrm{lo}}(x)$
output: $\delta_g = \tilde{h}(\psi) \cdot \mathsf{fl}_{\mathrm{lo}}(x) + \mathsf{fl}_{\mathrm{lo}}(\sigma) \cdot \mathsf{fl}_{\mathrm{lo}}(\delta_w)$

To compute each of these low-precision gradients requires $3d$ low-precision multiplies on the accelerator (d to compute $\mathsf{fl}_{\mathrm{lo}}(\delta_w)^T \mathsf{fl}_{\mathrm{lo}}(x)$, d to compute $\tilde{h}(\psi) \cdot \mathsf{fl}_{\mathrm{lo}}(x)$, and d to compute $\mathsf{fl}_{\mathrm{lo}}(\sigma) \cdot \mathsf{fl}_{\mathrm{lo}}(\delta_w)$), plus an additional $O(1)$ low-precision compute to compute the approximation $\tilde{h}(\psi)$. It also requires $2d$ low-precision loads (for δ_w and $\mathsf{fl}_{\mathrm{lo}}(x)$) plus an addition $O(1)$ low-precision loads for h_i and $\mathsf{fl}_{\mathrm{lo}}(\sigma)$. Thus, the total on-accelerator cost of bit centering for linear models is $3d + O(1)$ low-precision multiplies plus $2d + O(1)$ low-precision loads. This can be compared with the cost of the quantize-last strategy, which would require $3d$ high-precision multiplies plus $2d + 1$ high-precision loads. Thus, through the addition of only $O(1)$ extra low-precision loads and compute, compared with quantize-last, we have converted all the computation to low-precision. Note that this comes at the cost of some high-precision computation, which may be precomputed and done on the host as it only depends on o_w. The amount of this pre-computation is also $4d + O(1)$ high-precision multiplies plus $2d + 1$ high-precision loads and $d + O(1)$ low-precision stores.

Next, we will analyze the errors produced by this method. But first, we summarize our results in a table.

Quant. strat.	# LP Ops	# HP Ops	Result error (ℓ_2 norm)
High-precision	—	$2d$ loads $3d$ multiplies d adds 1 compute of l'	$\Big(2 \cdot \lvert l'(w^T x)\rvert \cdot \lVert x\rVert + \lVert w\rVert \cdot \lVert x\rVert^2 \cdot Ld$ $+\lvert\sigma\rvert\lVert w\rVert\Big) \cdot \varepsilon_{\text{hi}}$
Quantize last	—	$2d$ loads $3d$ multiplies d adds 1 compute of l'	$\Big((2+k) \cdot \lvert l'(w^T x)\rvert \cdot \lVert x\rVert + \lVert w\rVert \cdot \lVert x\rVert^2 \cdot Ld$ $+(1+k) \cdot \lvert\sigma\rvert\lVert w\rVert\Big) \cdot \varepsilon_{\text{hi}}$
Quantize first	$3d$ multiplies d adds 1 compute of l'	$2d$ loads	$\Big(2\lvert l'(w^T x)\rvert \cdot \lVert x\rVert + \lVert w\rVert \cdot \lVert x\rVert^2 \cdot Ld$ $+\lvert\sigma\rvert\lVert w\rVert\Big) \cdot k \cdot \varepsilon_{\text{hi}}$
Bit centering	$2d + O(1)$ loads $3d$ multiplies d adds 1 eval of $\tilde{h}$	$2d$ loads $4d$ multiplies $2d-1$ adds 1 approx of $\tilde{h}$ $2d + O(1)$ lp stores precomputed on host	error for δ_g: $\Big(2L \cdot \lVert x\rVert^2 \cdot \zeta + L \cdot \lVert x\rVert^2 \cdot d \cdot k$ $+\lvert\sigma\rvert \cdot k\Big) \cdot \lVert\delta_w\rVert \cdot \varepsilon_{\text{hi}}$

Error analysis. The error analysis for o_g is straightforward. From the algorithm statement, we have that

$$o_g = l'(\mathsf{fl}_{\text{hi}}(o_w)^T \mathsf{fl}_{\text{hi}}(x)) \cdot \mathsf{fl}_{\text{hi}}(x) + \mathsf{fl}_{\text{hi}}(\sigma) \cdot \mathsf{fl}_{\text{hi}}(o_w).$$

First, notice that by the dot product error analysis,

$$|\mathsf{fl}_{\text{hi}}(o_w)^T \mathsf{fl}_{\text{hi}}(x) - o_w^T x| \leq |o_w|^T |x| \cdot d \cdot \varepsilon_{\text{machine-hi}}.$$

This means that if l' is Lipschitz continuous, then

$$|l'(\mathsf{fl}_{\text{hi}}(o_w)^T \mathsf{fl}_{\text{hi}}(x)) - l'(o_w^T x)| \leq |o_w|^T |x| \cdot Ld \cdot \varepsilon_{\text{machine-hi}} + |l'(o_w^T x)| \cdot \varepsilon_{\text{machine-hi}}$$

where L is the Lipschitz constant of the function l', and the last term results from our assumption that we can compute the function l' to machine precision. It also follows that, if x_j is the jth component of the vector x,

$$\begin{aligned}
&|l'(\mathsf{fl}_{\text{hi}}(o_w)^T \mathsf{fl}_{\text{hi}}(x))\mathsf{fl}_{\text{hi}}(x_j) - l'(o_w^T x)x_j| \\
&\quad \leq |l'(\mathsf{fl}_{\text{hi}}(o_w)^T \mathsf{fl}_{\text{hi}}(x))\mathsf{fl}_{\text{hi}}(x_j) - l'(\mathsf{fl}_{\text{hi}}(o_w)^T \mathsf{fl}_{\text{hi}}(x))x_j| + |l'(\mathsf{fl}_{\text{hi}}(o_w)^T \mathsf{fl}_{\text{hi}}(x))x_j - l'(o_w^T x)x_j| \\
&\quad \leq |l'(\mathsf{fl}_{\text{hi}}(o_w)^T \mathsf{fl}_{\text{hi}}(x))x_j| \cdot \varepsilon_{\text{machine-hi}} + |o_w|^T |x| \cdot |x_j| \cdot Ld \cdot \varepsilon_{\text{machine-hi}} + |l'(o_w^T x)| \cdot |x_j| \cdot \varepsilon_{\text{machine-hi}} \\
&\quad \leq 2|l'(o_w^T x)x_j| \cdot \varepsilon_{\text{machine-hi}} + |o_w|^T |x| \cdot |x_j| \cdot Ld \cdot \varepsilon_{\text{machine-hi}}
\end{aligned}$$

where in the last line we implicitly dropped terms in ε^2. It follows that the error for the jth component of o_g, which we denote $o_{g,j}$, is

$$\begin{aligned}
|o_{g,j} - \big(l'(o_w^T x)x_j + \sigma o_{w,j}\big)| &= |l'(\mathsf{fl}_{\text{hi}}(o_w)^T \mathsf{fl}_{\text{hi}}(x))\mathsf{fl}_{\text{hi}}(x_j) + \mathsf{fl}_{\text{hi}}(\sigma)\mathsf{fl}_{\text{hi}}(o_{w,j}) - \big(l'(o_w^T x)x_j + \sigma o_{w,j}\big)| \\
&\leq |l'(\mathsf{fl}_{\text{hi}}(o_w)^T \mathsf{fl}_{\text{hi}}(x))\mathsf{fl}_{\text{hi}}(x_j) - l'(o_w^T x)x_j| + |\mathsf{fl}_{\text{hi}}(\sigma)\mathsf{fl}_{\text{hi}}(o_{w,j}) - \sigma o_{w,j}| \\
&\leq 2|l'(o_w^T x)x_j| \cdot \varepsilon_{\text{machine-hi}} + |o_w|^T |x| \cdot |x_j| \cdot Ld \cdot \varepsilon_{\text{machine-hi}} + |\sigma o_{w,j}|\varepsilon_{\text{machine-hi}}.
\end{aligned}$$

Or, in terms of infinity norms, the error will be

$$\lVert o_g - \big(l'(o_w^T x)x + \sigma o_w\big)\rVert_\infty \leq \Big(2|l'(o_w^T x)|\lVert x\rVert_\infty + |o_w|^T |x| \cdot \lVert x\rVert_\infty \cdot Ld + |\sigma|\lVert o_w\rVert_\infty\Big) \cdot \varepsilon_{\text{machine-hi}}$$

Note that basically the same analysis would apply to give us a bound on the error of the quantize-first and quantize-last strategies. Specifically, for quantize-first, everything will be the same, except in low-precision, so we get

$$\begin{aligned}\|g_{\text{quantize-first}} - \big(l'(o_w^T x)x + \sigma w\big)\|_\infty &\leq \Big(2|l'(w^T x)|\|x\|_\infty + |w|^T|x| \cdot \|x\|_\infty \cdot Ld + |\sigma|\|w\|_\infty\Big) \cdot \varepsilon_{\text{machine-lo}} \\ &\leq \Big(2|l'(w^T x)|\|x\|_\infty + |w|^T|x| \cdot \|x\|_\infty \cdot Ld + |\sigma|\|w\|_\infty\Big) \cdot k \cdot \varepsilon_{\text{machine-hi}}.\end{aligned}$$

For quantize last, everything will again be the same, except that we will get an extra error term from quantizing into low-precision at the end, so

$$\begin{aligned}&\|g_{\text{quantize-last}} - \big(l'(w^T x)x + \sigma w\big)\|_\infty \\ &\quad\leq \Big(2|l'(w^T x)|\|x\|_\infty + |w|^T|x| \cdot \|x\|_\infty \cdot Ld + |\sigma|\|w\|_\infty + k \cdot \|l'(w^T x)x + \sigma w\|_\infty\Big) \cdot \varepsilon_{\text{machine-hi}} \\ &\quad\leq \Big((2+k) \cdot |l'(w^T x)|\|x\|_\infty + |w|^T|x| \cdot \|x\|_\infty \cdot Ld + (1+k) \cdot |\sigma|\|w\|_\infty\Big) \cdot \varepsilon_{\text{machine-hi}}.\end{aligned}$$

The analysis for the low-precision section is similar. First, notice that

$$|\mathsf{fl}_{\text{lo}}(\delta_w)^T \mathsf{fl}_{\text{lo}}(x) - \delta_w^T x| \leq |\delta_w|^T|x| \cdot d \cdot \varepsilon_{\text{machine-lo}}.$$

It follows that, since $h(\psi) = l'(\phi + \psi) - l'(\phi)$, h has the same Lipschitz constant as l', and so

$$\begin{aligned}|\tilde{h}(\mathsf{fl}_{\text{lo}}(\delta_w)^T \mathsf{fl}_{\text{lo}}(x)) - h(\delta_w^T x)| &\leq |\tilde{h}(\mathsf{fl}_{\text{lo}}(\delta_w)^T \mathsf{fl}_{\text{lo}}(x)) - h(\mathsf{fl}_{\text{lo}}(\delta_w)^T \mathsf{fl}_{\text{lo}}(x))| + |h(\mathsf{fl}_{\text{lo}}(\delta_w)^T \mathsf{fl}_{\text{lo}}(x)) - h(\delta_w^T x)| \\ &\leq \zeta \cdot |\mathsf{fl}_{\text{lo}}(\delta_w)^T \mathsf{fl}_{\text{lo}}(x)| \cdot \max_{u \in [-a,a]} \left|\frac{h(u)}{u}\right| \cdot \varepsilon_{\text{machine-lo}} + L \cdot |\mathsf{fl}_{\text{lo}}(\delta_w)^T \mathsf{fl}_{\text{lo}}(x)) - \delta_w^T x| \\ &\leq \zeta \cdot |\delta_w^T x| \cdot \max_{u \in [-a,a]} \left|\frac{h(u)}{u}\right| \cdot \varepsilon_{\text{machine-lo}} + L \cdot |\mathsf{fl}_{\text{lo}}(\delta_w)^T \mathsf{fl}_{\text{lo}}(x)) - \delta_w^T x|,\end{aligned}$$

where in the last line we are implicitly also dropping terms that are $O(\varepsilon^2)$. Next, we need to bound this max term. To do so, notice that for any u,

$$\left|\frac{h(u)}{u}\right| = \left|\frac{l'(\phi + u) - l'(\phi)}{u}\right| \leq L.$$

So,

$$\begin{aligned}|\tilde{h}(\mathsf{fl}_{\text{lo}}(\delta_w)^T \mathsf{fl}_{\text{lo}}(x)) - h(\delta_w^T x)| &\leq L \cdot |\delta_w^T x| \cdot \zeta \cdot \varepsilon_{\text{machine-lo}} + L \cdot |\mathsf{fl}_{\text{lo}}(\delta_w)^T \mathsf{fl}_{\text{lo}}(x)) - \delta_w^T x| \\ &\leq L \cdot |\delta_w^T x| \cdot \zeta \cdot \varepsilon_{\text{machine-lo}} + L \cdot |\delta_w|^T|x| \cdot d \cdot \varepsilon_{\text{machine-lo}}.\end{aligned}$$

It also follows that, if x_j is the jth component of the vector x,

$$\begin{aligned}&|\tilde{h}(\mathsf{fl}_{\text{lo}}(\delta_w)^T \mathsf{fl}_{\text{lo}}(x))\mathsf{fl}_{\text{lo}}(x_j) - h(\delta_w^T x)x_j| \\ &\quad\leq |\tilde{h}(\mathsf{fl}_{\text{lo}}(\delta_w)^T \mathsf{fl}_{\text{lo}}(x))\mathsf{fl}_{\text{lo}}(x_j) - \tilde{h}(\mathsf{fl}_{\text{lo}}(\delta_w)^T \mathsf{fl}_{\text{lo}}(x))x_j| + |\tilde{h}(\mathsf{fl}_{\text{lo}}(\delta_w)^T \mathsf{fl}_{\text{lo}}(x))x_j - h(\delta_w^T x)x_j| \\ &\quad\leq |\tilde{h}(\mathsf{fl}_{\text{lo}}(\delta_w)^T \mathsf{fl}_{\text{lo}}(x))x_j| \cdot \varepsilon_{\text{machine-lo}} + L \cdot |\delta_w^T x| \cdot |x_j| \cdot \zeta \cdot \varepsilon_{\text{machine-lo}} + L \cdot |\delta_w|^T|x| \cdot |x_j| \cdot d \cdot \varepsilon_{\text{machine-lo}} \\ &\quad\leq 2L \cdot |\delta_w^T x| \cdot |x_j| \cdot \zeta \cdot \varepsilon_{\text{machine-lo}} + L \cdot |\delta_w|^T|x| \cdot |x_j| \cdot d \cdot \varepsilon_{\text{machine-lo}}.\end{aligned}$$

where in the second-to last line we implicitly dropped terms that are $O(\varepsilon^2)$, and in the last line we leveraged the fact that h is Lipschitz continuous with parameter L. It follows that the error for the jth component of δ_g, which we denote $\delta_{g,j}$, is

$$\begin{aligned}|\delta_{g,j} - \big(h(\delta_w^T x)x_j + \sigma\delta_{w,j}\big)| &= |\tilde{h}(\mathsf{fl}_{\text{lo}}(\delta_w)^T \mathsf{fl}_{\text{lo}}(x))\mathsf{fl}_{\text{lo}}(x_j) + \mathsf{fl}_{\text{lo}}(\sigma)\mathsf{fl}_{\text{lo}}(\delta_{w,j}) - \big(h(\delta_w^T x)x_j + \sigma\delta_{w,j}\big)| \\ &\leq |\tilde{h}(\mathsf{fl}_{\text{lo}}(\delta_w)^T \mathsf{fl}_{\text{lo}}(x))\mathsf{fl}_{\text{lo}}(x_j) - h(\delta_w^T x)x_j| + |\mathsf{fl}_{\text{lo}}(\sigma)\mathsf{fl}_{\text{lo}}(\delta_{w,j}) - \sigma\delta_{w,j}| \\ &\leq \Big(2L \cdot |\delta_w^T x| \cdot |x_j| \cdot \zeta + L \cdot |\delta_w|^T|x| \cdot |x_j| \cdot d + |\sigma\delta_{w,j}|\Big) \cdot \varepsilon_{\text{machine-lo}}.\end{aligned}$$

It follows that we can bound this in terms of the infinity norm as

$$\|\delta_g - \big(h(\delta_w^T x)x + \sigma\delta_w\big)\|_\infty \leq \Big(2L \cdot |\delta_w^T x| \cdot \|x\|_\infty \cdot \zeta + L \cdot |\delta_w|^T |x| \cdot \|x\|_\infty \cdot d + |\sigma| \cdot \|\delta_w\|_\infty\Big) \cdot \varepsilon_{\text{machine-lo}}$$

or in terms of the ℓ_2-norm as

$$\begin{aligned}
\|\delta_g - \big(h(\delta_w^T x)x + \sigma\delta_w\big)\| &\leq 2L \cdot |\delta_w^T x| \cdot \|x\| \cdot \zeta \cdot \varepsilon_{\text{machine-lo}} + L \cdot |\delta_w|^T |x| \cdot \|x\| \cdot d \cdot \varepsilon_{\text{machine-lo}} + |\sigma| \cdot \|\delta_w\| \cdot \varepsilon_{\text{machine-lo}} \\
&\leq 2L \cdot \|\delta_w\| \cdot \|x\|^2 \cdot \zeta \cdot \varepsilon_{\text{machine-lo}} + L \cdot \|\delta_w\| \|x\|^2 \cdot d \cdot \varepsilon_{\text{machine-lo}} + |\sigma| \cdot \|\delta_w\| \cdot \varepsilon_{\text{machine-lo}} \\
&= \Big(L \cdot \|x\|^2 \cdot (2\zeta + d) + |\sigma|\Big) \cdot \|\delta_w\| \cdot \varepsilon_{\text{machine-lo}}.
\end{aligned}$$

Notice a very important property of this error bound: it is proportional to δ_w. That is, as our offset guess becomes closer to the true value of w (equivalently, as δ_w gets smaller), the error of our δ_g computation becomes smaller at the same rate.

3.2 Full Learning Algorithms

Combining everything together produces the following full algorithm for learning on linear models. We call this method *Bit-Centered SGD*.

Algorithm 2 BC-SGD (Bit-Centered SGD) for Linear Models

given: N loss functions l_i and training examples x_i in high precision, number of epochs K, epoch length T, step size α
given: initial iterate $w_1 = o_{w,1}$ in high precision
for $k = 1$ **to** K **do**
  for $i = 1$ **to** N **do**
    $\phi_{i,k} \leftarrow \mathsf{fl}_{\text{hi}}(x_i)^T \mathsf{fl}_{\text{hi}}(o_{w,k})$
    define: $h_i(\psi) = \alpha l_i'(\phi_i + \psi) - \alpha l_i'(\phi_i)$
    compute approximation: $\tilde{h}_i \approx h_i$
    quantize: $\tilde{x}_i \leftarrow \mathsf{fl}_{\text{lo}}(x_i)$
    quantize: $\tilde{g}_{k,i} \leftarrow \mathsf{fl}_{\text{lo}}(\mathsf{fl}_{\text{hi}}(\alpha) \cdot l_i'(\mathsf{fl}_{\text{hi}}(x_i)^T \mathsf{fl}_{\text{hi}}(o_{w,k})) \cdot \mathsf{fl}_{\text{hi}}(x_i))$
  end for
  $\delta_{w,k,0} \leftarrow \mathsf{fl}_{\text{lo}}(0)$
  for $t = 1$ **to** T **do**
    sample i uniformly from $\{1, \ldots, N\}$
    $\psi_{k,t} \leftarrow \mathsf{fl}_{\text{lo}}(\tilde{x}_i)^T \mathsf{fl}_{\text{lo}}(\delta_{w,k,t-1})$
    $u_{k,t} \leftarrow \tilde{h}_i(\mathsf{fl}_{\text{lo}}(\psi_{k,t})) \cdot \mathsf{fl}_{\text{lo}}(\tilde{x}_i)$
    update model: $\delta_{w,k,t} \leftarrow \mathsf{fl}_{\text{lo}}(\delta_{w,k,t-1}) - \mathsf{fl}_{\text{lo}}(\mathbf{u}_{k,t}) - \mathsf{fl}_{\text{lo}}(\tilde{g}_{k,i})$
  end for
  $o_{w,k+1} \leftarrow \mathsf{fl}_{\text{hi}}(o_{w,k}) + \mathsf{fl}_{\text{hi}}(\delta_{w,k,T})$
end for
return $o_{w,K+1}$

Notice that the entire inner loop (here outlined in red) is computed only using low-precision arithmetic. Furthermore, the bulk of the outer-loop computation (the part outlined in blue) is embarassingly parallel across training examples. The bit centering method also naturally lends itself to a variant of *stochastic variance reduced gradient* (SVRG) Johnson and Zhang [2]: all that we need to do is replace the individual stored gradients $\tilde{g}_{k,i}$ with the average gradient.

Algorithm 3 BC-SVRG (Bit-Centered SVRG) for Linear Models

given: N loss functions l_i and training examples x_i in high precision, number of epochs K, epoch length T, step size α
given: initial iterate $w_1 = o_{w,1}$ in high precision
for $k = 1$ **to** K **do**
  for $i = 1$ **to** N **do**
    $\phi_{i,k} \leftarrow \mathsf{fl}_{\mathrm{hi}}(x_i)^T \mathsf{fl}_{\mathrm{hi}}(o_{w,k})$
    define: $h_i(\psi) = \alpha l_i'(\phi_i + \psi) - \alpha l_i'(\phi_i)$
    compute approximation: $\tilde{h}_i \approx h_i$
    quantize: $\tilde{x}_i \leftarrow \mathsf{fl}_{\mathrm{lo}}(x_i)$
  end for
  $g_k \leftarrow \frac{\mathsf{fl}_{\mathrm{hi}}(\alpha)}{N} \sum_{i=1}^{N} \cdot l_i'(\mathsf{fl}_{\mathrm{hi}}(x_i)^T \mathsf{fl}_{\mathrm{hi}}(o_{w,k})) \cdot \mathsf{fl}_{\mathrm{hi}}(x_i)$
  quantize: $\tilde{g}_k \leftarrow \mathsf{fl}_{\mathrm{lo}}(g_k)$
  $\delta_{w,k,0} \leftarrow \mathsf{fl}_{\mathrm{lo}}(0)$
  for $t = 1$ **to** T **do**
    sample i uniformly from $\{1, \ldots, N\}$
    $\psi_{k,t} \leftarrow \mathsf{fl}_{\mathrm{lo}}(\tilde{x}_i)^T \mathsf{fl}_{\mathrm{lo}}(\delta_{w,k,t-1})$
    $u_{k,t} \leftarrow \tilde{h}_i(\mathsf{fl}_{\mathrm{lo}}(\psi_{k,t})) \cdot \mathsf{fl}_{\mathrm{lo}}(\tilde{x}_i)$
    update model: $\delta_{w,k,t} \leftarrow \mathsf{fl}_{\mathrm{lo}}(\delta_{w,k,t-1}) - \mathsf{fl}_{\mathrm{lo}}(\mathbf{u}_{k,t}) - \mathsf{fl}_{\mathrm{lo}}(\tilde{g}_k)$
  end for
  $o_{w,k+1} \leftarrow \mathsf{fl}_{\mathrm{hi}}(o_{w,k}) + \mathsf{fl}_{\mathrm{hi}}(\delta_{w,k,T})$
end for
return $o_{w,K+1}$

Notice again that the entire inner loop of the algorithm (the part outlined in red) will be done *entirely* using low-precision loads and stores, since it only reads and writes the variables v, z, and $\tilde{h}$, all of which are low-precision values. Additionally, the entire inner loop of the algorithm will also be computed using only low-precision arithmetic. Furthermore, the bulk of the outer-loop computation (the part outlined in blue) is embarrassingly parallel across training examples. Figure 1 illustrates the convergence of BC-SGD and BC-SVRG on a toy logistic regression problem, with model dimension $d = 128$, number of training examples $N = 1024$. This problem was generated by first sampling a random generative model $w_{\text{gen}} \sim \mathcal{N}(0, I_d)$, where $\mathcal{N}$ denotes the normal distribution and $I_d \in \mathbb{R}^{d \times d}$ is the identity matrix, and then sampling n training examples (x_i, y_i) for $i \in \{1, \ldots, N\}$ as

$$x_i \sim \mathcal{N}(0, I_d), \qquad /du_i \sim \mathcal{N}(0, 1), \qquad y_i = \text{sign}(x_i^T w_{\text{gen}} + u_i).$$

This process results in a logistic regression problem with objective

$$f(w) = \frac{1}{N} \sum_{i=1}^{n} f_i(w) = \frac{1}{N} \sum_{i=1}^{n} \log\left(1.0 + \exp(-y_i x_i^T w)\right).$$

This problem will have condition number $\kappa = 1150$ near the global optimum (the condition number κ in this context is the ratio between the maximum Lipschitz constant L of ∇f_i and the strong convexity constant μ of f). For the experiments, we ran each algorithm for $K = 200$ epochs, each of which had $T = 1000$ inner iterations. For SGD and BC-SGD, we used a learning rate $\alpha = 0.03$, and for BC-SVRG we used $\alpha = 1.0$; these learning rates were hand-tuned.

Notice that the SVRG implementation has the very interesting property of having *high accuracy*. That is, it is able to converge to solutions of accuracy determined by the machine epsilon of the high-precision numbers, even though it uses low-precision numbers for its inner loop computations (in this case, sometimes very low precision numbers). The figure also illustrates an effect that has been previously observed: that randomized rounding, which rounds up or down at random when quantizing such that the quantized output is an unbiased estimator of the original value, performs better than standard round-to-nearest quantization.

Another interesting observation in these figures is that when the number of exponent bits is small, performance degrades for BC-SVRG, and the algorithm converges to a higher error than comparable runs with lower numbers of bits. This occurs because eventually, the low-precision δ values used in bit centering start underflowing, which limits convergence to a level at which δ values are still large enough to be representable in low precision. One way to address

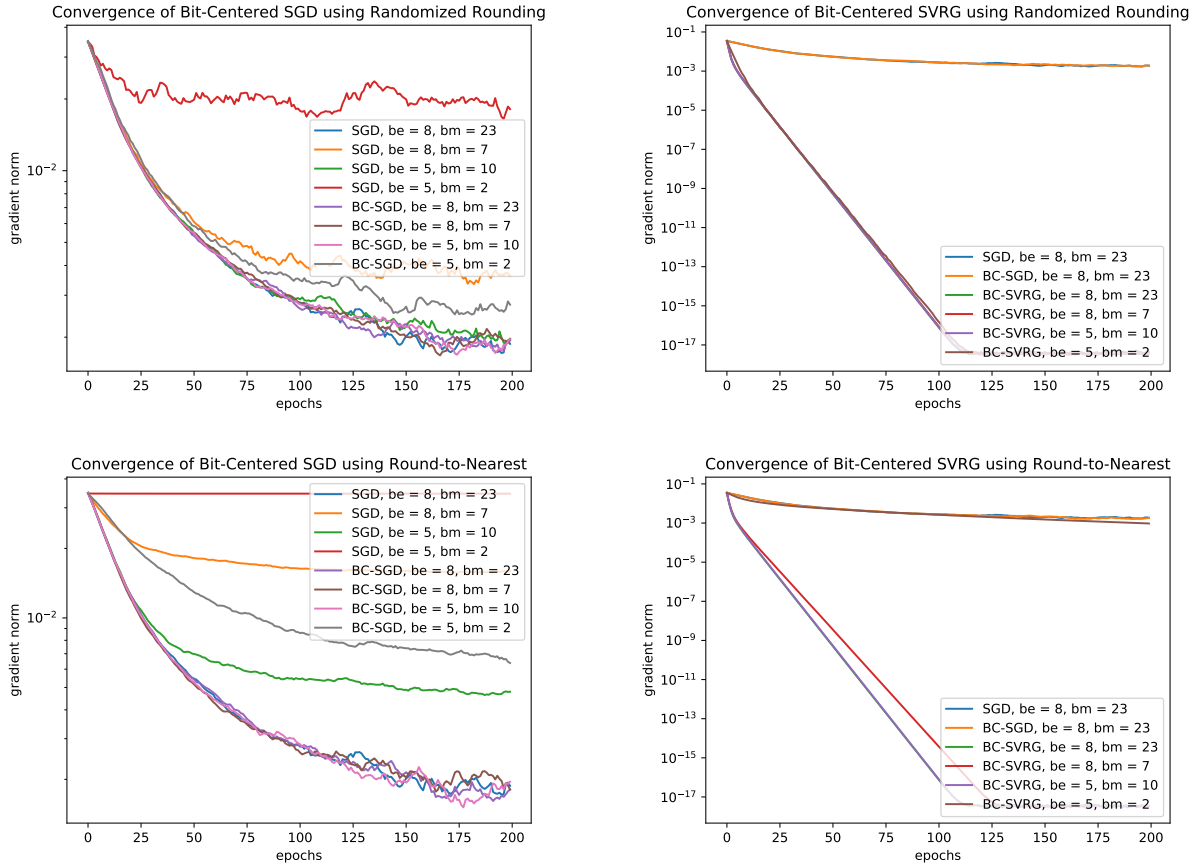

Figure 1: Convergence of BC-SGD (left) and BC-SVRG (right) compared with ordinary quantize-first low-precision SGD on a toy synthetic logistic regression problem (with model dimension $d = 128$, number of training examples $n = 1024$, condition number $\kappa = 1150$ near the global optimum, and no regularization). Single-precision floating point numbers (8-bit exponent, 23-bit mantissa) were used for the high-precision numbers in all experiments, and for low-precision numbers the number of exponent bits be and the number of mantissa bits bm are reported (the number of exponent bits was fixed at 11, as is the case for double-precision floats). Randomized rounding was used for all quantization in the top row, while nearest-neighbor rounding was used for all quantization in the bottom row.

this is by dynamically adjusting the *exponent bias*. Exponent bias is used in floating point numbers to control the range of the exponents that are supported. For example, a single precision number with exponent e (an unsigned 8-bit number) and mantissa m (a 23-bit digit string) has value

$$(-1)^{\text{sign}} \cdot 2^{e-127} \cdot (1.m);$$

this 127 number is the exponent bias. Instead of using a fixed bias for representing δ for bit-centering, we can use a bias that decreases over time proportional to the magnitude of the δ values we know we are going to compute. That is, we define a new low-precision floating point representation in which the value is

$$(-1)^{\text{sign}} \cdot 2^{e-127+\textbf{extra bias}} \cdot (1.m);$$

We let $\text{fl}_{\text{lo}-\text{bias}}()$ denote this bias-adjuested low-precision representation. Note that we would still want to use standard-bias low-precision numbers for storing intermediate values that do not decrease in magnitude with δ, such as the $\text{fl}_{\text{lo}}(o_x)$ used in a bit-centered multiply. Doing this results in the following algorithm.

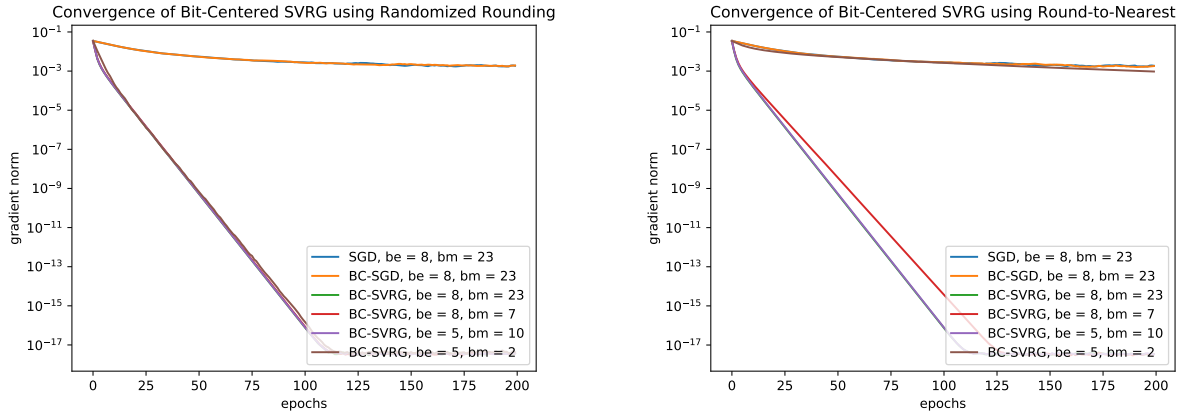

Figure 2: Convergence of dynamic-bias-adjusted BC-SVRG with randomized rounding (left) and nearest-neighbor rounding (right) on the same toy synthetic logistic regression problem used in Figure 1. For the bias control parameter, we used $\chi = 0.01$.

Algorithm 4 Dynamic-Bias-Adjusted BC-SVRG for Linear Models

given: N loss functions l_i and training examples x_i in high precision, number of epochs K, epoch length T, step size α
given: initial iterate $w_1 = o_{w,1}$ in high precision
given: bias control parameter χ
for $k = 1$ **to** K **do**
  for $i = 1$ **to** N **do**
    $\phi_{i,k} \leftarrow \mathsf{fl}_{\mathrm{hi}}(x_i)^T \mathsf{fl}_{\mathrm{hi}}(o_{w,k})$
    define: $h_i(\psi) = \alpha l_i'(\phi_i + \psi) - \alpha l_i'(\phi_i)$
    compute approximation: $\tilde{h}_i \approx h_i$
    quantize: $\tilde{x}_i \leftarrow \mathsf{fl}_{\mathrm{lo}}(x_i)$
  end for
  $g_k \leftarrow \dfrac{\mathsf{fl}_{\mathrm{hi}}(\alpha)}{N} \sum_{i=1}^{N} \cdot l_i'(\mathsf{fl}_{\mathrm{hi}}(x_i)^T \mathsf{fl}_{\mathrm{hi}}(o_{w,k})) \cdot \mathsf{fl}_{\mathrm{hi}}(x_i)$
  extra bias $\leftarrow \lfloor \log_2 (\chi \cdot \|g_k\|_\infty) \rfloor$
  quantize: $\tilde{g}_k \leftarrow \mathsf{fl}_{\mathrm{lo-bias}}(g_k)$
  $\delta_{w,k,0} \leftarrow \mathsf{fl}_{\mathrm{lo-bias}}(0)$
  for $t = 1$ **to** T **do**
    sample i uniformly from $\{1, \ldots, N\}$
    $\psi_{k,t} \leftarrow \mathsf{fl}_{\mathrm{lo}}(\tilde{x}_i)^T \mathsf{fl}_{\mathrm{lo-bias}}(\delta_{w,k,t-1})$
    $u_{k,t} \leftarrow \tilde{h}_i(\mathsf{fl}_{\mathrm{lo-bias}}(\psi_{k,t})) \cdot \mathsf{fl}_{\mathrm{lo}}(\tilde{x}_i)$
    update model: $\delta_{w,k,t} \leftarrow \mathsf{fl}_{\mathrm{lo-bias}}(\delta_{w,k,t-1}) - \mathsf{fl}_{\mathrm{lo-bias}}(\mathbf{u}_{k,t}) - \mathsf{fl}_{\mathrm{lo-bias}}(\tilde{g}_k)$
  end for
  $o_{w,k+1} \leftarrow \mathsf{fl}_{\mathrm{hi}}(o_{w,k}) + \mathsf{fl}_{\mathrm{hi}}(\delta_{w,k,T})$
end for
return $o_{w,K+1}$

Note that the extra bias value is global across all numbers, and so little extra memory is used to compute and store it. This technique can have a dramatic effect on the convergence of BC-SVRG, as shown in Figure 2.

4 Bit Centering for General Loss Functions

In the previous section, we described how to do bit centering for linear loss functions, which had a particular form. We noticed that we were able to produce a bound on the error of the delta of the gradients of the form

$$\text{error}(\delta_g) \leq C \cdot \|\delta_w\| \cdot \varepsilon_{\text{machine-lo}}$$

for a constant C that was a function of the problem parameters. Next, we will show how we can get a bound of the same form on the delta error of the gradients for *arbitrary* loss functions which can be computed via a computation graph. We will do this with a particular focus on deep neural networks as an application.

To start, recall an interesting property of our error analysis for the linear models case. When we did the analysis, we derived bounds on o_g using a particular approach, and then repeated *almost the same calculation* to get a bound on δ_g. This suggests that a more sophisticated analysis of a bit centering computation can get a bound on the error of computing the offset *and* the error of computing the delta at the same time.

Let $\mathsf{bc}(x)$ denote the value x represented using bit centering. That is, we represent it as $x = o_x + \delta_x$ using a high-precision-floating-point offset o_x and a low-precision-floating-point delta δ_x. Then we can compute the error of bit-centered operations just like we would compute the error of floating point operations. We will start this section by doing this. But first, we present a table summarizing our results. For matrix operands, we assume dimension of $m \times n$, and for matrix product the first operand is $m \times n$ and the second is $n \times p$.

Operation	**# LP Ops**	**# HP Ops**	**Result error ($\cdot\varepsilon_{\text{machine-hi}}$)**
Scalar sum	1 add	1 add	$\lvert x+y\rvert + \lvert\delta_x+\delta_y\rvert \cdot (k+1)$
Scalar difference	1 sub	1 sub	$\lvert x-y\rvert + \lvert\delta_x-\delta_y\rvert \cdot (k+1)$
Scalar product	2 extra loads 2 multiplies 2 adds	1 multiply 2 extra stores	$\lvert x\cdot y\rvert + (\lvert o_x\cdot\delta_y\rvert + \lvert\delta_x\cdot o_y\rvert + \lvert\delta_x\cdot\delta_y\rvert)\cdot(4k+1)$
Scalar function eval	$O(1)$ extra loads 1 eval of $\tilde{h}$	1 eval of f 1 function approx $O(1)$ extra stores	$\lvert f(x)\rvert + (k\cdot\zeta+1)\cdot L_f\cdot\lvert\delta_x\rvert$
Matrix sum	mn adds	mn adds	$\|X+Y\| + \|\delta_X+\delta_Y\|\cdot(k+1)$
Matrix difference	mn subs	mn subs	$\|X-Y\| + \|\delta_X-\delta_Y\|\cdot(k+1)$
Hadamard product	$2mn$ extra loads $2mn$ multiplies $2mn$ adds	mn multiplies $2mn$ extra stores	$\|X\cdot Y\|+$ $\quad\|\lvert o_X\cdot\delta_Y\rvert + \lvert\delta_X\cdot o_Y\rvert + \lvert\delta_X\cdot\delta_Y\rvert\|\cdot(4k+1)$
Entrywise f eval	$O(mn)$ extra loads mn evals of $\tilde{h}$	mn evals of f mn function approx $O(mn)$ extra stores	$\|f(x)\| + (k\cdot\zeta+1)\cdot L_f\cdot\|\delta_x\|$
Matrix product	$mn+np$ extra loads $2mnp$ multiplies $2mnp+mp+np$ adds	mnp multiplies mnp adds $mn+np$ extra stores	$\|\lvert X\rvert\cdot\lvert Y\rvert\|\cdot n+$ $\quad\|\lvert o_X\rvert\cdot\lvert\delta_Y\rvert + \lvert\delta_X\rvert\cdot\lvert o_Y\rvert + \lvert\delta_X\rvert\cdot\lvert\delta_Y\rvert\|\cdot$ $\quad(nk+3k+n)$
Const matrix prod	np extra loads mnp multiplies mnp adds	mnp multiplies mnp adds np extra stores	$\|\lvert X\rvert\cdot\lvert o_Y\rvert\|\cdot n + \|\lvert\delta_X\rvert\cdot\lvert o_Y\rvert\|\cdot(nk+k+1)$

4.1 Computing the Error of Bit-Centered Operations

Addition Suppose that we want to do addition using bit centering, to compute $\mathsf{bc}(x) + \mathsf{bc}(y)$ where $x = o_x + \delta_x$ and $y = o_y + \delta_y$. The natural way to do this is to let

$$\mathsf{bc}(x) + \mathsf{bc}(y) \stackrel{\text{def}}{=} (\mathsf{fl}_{\text{hi}}(o_x) + \mathsf{fl}_{\text{hi}}(o_y)) + (\mathsf{fl}_{\text{lo}}(\delta_x) + \mathsf{fl}_{\text{lo}}(\delta_y)),$$

where more explicitly the offset is the first term $\mathsf{fl}_{\text{hi}}(o_x) + \mathsf{fl}_{\text{hi}}(o_y)$ and the delta is the second term $\mathsf{fl}_{\text{lo}}(\delta_x) + \mathsf{fl}_{\text{lo}}(\delta_y)$. This will have error

$$\begin{aligned}
|\mathsf{bc}(x) + \mathsf{bc}(y) - (x + y)| &= |(\mathsf{fl}_{\text{hi}}(o_x) + \mathsf{fl}_{\text{hi}}(o_y)) + (\mathsf{fl}_{\text{lo}}(\delta_x) + \mathsf{fl}_{\text{lo}}(\delta_y)) - (o_x + o_y + \delta_x + \delta_y)| \\
&\leq |(\mathsf{fl}_{\text{hi}}(o_x) + \mathsf{fl}_{\text{hi}}(o_y)) - (o_x + o_y)| + |(\mathsf{fl}_{\text{lo}}(\delta_x) + \mathsf{fl}_{\text{lo}}(\delta_y)) - (\delta_x + \delta_y)| \\
&\leq |o_x + o_y| \cdot \varepsilon_{\text{machine-hi}} + |\delta_x + \delta_y| \cdot \varepsilon_{\text{machine-lo}} \\
&\leq |x + y| \cdot \varepsilon_{\text{machine-hi}} + |\delta_x + \delta_y| \cdot (k+1) \cdot \varepsilon_{\text{machine-hi}}.
\end{aligned}$$

Subtraction The analysis is the same. We will have

$$|\mathsf{bc}(x) - \mathsf{bc}(y) - (x - y)| \leq |x - y| \cdot \varepsilon_{\text{machine-hi}} + |\delta_x - \delta_y| \cdot (k+1) \cdot \varepsilon_{\text{machine-hi}}.$$

Multiplication Suppose that we want to do multiplication using bit centering, to compute $\mathsf{bc}(x) \cdot \mathsf{bc}(y)$ where $x = o_x + \delta_x$ and $y = o_y + \delta_y$. The natural way to do this is to let the offset of the result be computed as

$$o_{\mathsf{bc}(x) \cdot \mathsf{bc}(y)} \stackrel{\text{def}}{=} \mathsf{fl}_{\text{hi}}(o_x) \cdot \mathsf{fl}_{\text{hi}}(o_y)$$

and the delta be computed as

$$\delta_{\mathsf{bc}(x) \cdot \mathsf{bc}(y)} \stackrel{\text{def}}{=} \mathsf{fl}_{\text{lo}}(o_x) \cdot \mathsf{fl}_{\text{lo}}(\delta_y) + \mathsf{fl}_{\text{lo}}(\delta_x) \cdot (\mathsf{fl}_{\text{lo}}(o_y) + \mathsf{fl}_{\text{lo}}(\delta_y)).$$

Note that in order to compute this delta, we will need to additionally quantize and load low-precision version of o_x and o_y; this will require additional loads on the accelerator. Using the $1 + \varepsilon$ model, this will look like

$$\begin{aligned}
\mathsf{bc}(x) \cdot \mathsf{bc}(y) &= o_{\mathsf{bc}(x) \cdot \mathsf{bc}(y)} + \delta_{\mathsf{bc}(x) \cdot \mathsf{bc}(y)} \\
&= \mathsf{fl}_{\text{hi}}(o_x) \cdot \mathsf{fl}_{\text{hi}}(o_y) + \mathsf{fl}_{\text{lo}}(o_x) \cdot \mathsf{fl}_{\text{lo}}(\delta_y) + \mathsf{fl}_{\text{lo}}(\delta_x) \cdot (\mathsf{fl}_{\text{lo}}(o_y) + \mathsf{fl}_{\text{lo}}(\delta_y)) \\
&= (o_x \cdot o_y) \cdot (1 + \varepsilon_{\text{hi}}) \\
&\quad + \Big(((o_x \cdot (1 + \varepsilon_{\text{lo}})) \cdot \delta_y) \cdot (1 + \varepsilon_{\text{lo}}) \\
&\qquad + (\delta_x \cdot ((o_y \cdot (1 + \varepsilon_{\text{lo}})) + \delta_y) \cdot (1 + \varepsilon_{\text{lo}})) \cdot (1 + \varepsilon_{\text{lo}}) \Big) \cdot (1 + \varepsilon_{\text{lo}})
\end{aligned}$$

where as usual in the $1+\varepsilon$ model, all the ε in this expression can take on different values and are bounded as appropriate by the machine epsilon. By dropping terms in ε^2, we can simplify this to

$$\begin{aligned}
\mathsf{bc}(x) \cdot \mathsf{bc}(y) &= (o_x \cdot o_y) \cdot (1 + \varepsilon_{\text{hi}}) \\
&\quad + (o_x \cdot \delta_y) \cdot 2\varepsilon_{\text{lo}} \\
&\quad + (\delta_x \cdot o_y) \cdot \varepsilon_{\text{lo}} \\
&\quad + (\delta_x \cdot (o_y + \delta_y)) \cdot 2\varepsilon_{\text{lo}} \\
&\quad + (o_x \cdot \delta_y + \delta_y \cdot (o_y + \delta_y)) \cdot (1 + \varepsilon_{\text{lo}}) \\
&= (x \cdot y) \cdot (1 + \varepsilon_{\text{hi}}) \\
&\quad + (o_x \cdot \delta_y) \cdot 2\varepsilon_{\text{lo}} \\
&\quad + (\delta_x \cdot o_y) \cdot \varepsilon_{\text{lo}} \\
&\quad + (\delta_x \cdot (o_y + \delta_y)) \cdot 2\varepsilon_{\text{lo}} \\
&\quad + (o_x \cdot \delta_y + \delta_y \cdot (o_y + \delta_y)) \cdot (\varepsilon_{\text{lo}} + \varepsilon_{\text{hi}}) \\
&= (x \cdot y) \cdot (1 + \varepsilon_{\text{hi}}) \\
&\quad + (o_x \cdot \delta_y) \cdot (3k + 1) \cdot \varepsilon_{\text{hi}} \\
&\quad + (\delta_x \cdot o_y) \cdot (4k + 1) \cdot \varepsilon_{\text{hi}} \\
&\quad + (\delta_x \cdot \delta_y) \cdot (3k + 1) \cdot \varepsilon_{\text{hi}}.
\end{aligned}$$

It follows that the error will be bounded by

$$|\mathsf{bc}(x) \cdot \mathsf{bc}(y) - (x \cdot y)| \leq |x \cdot y| \cdot \varepsilon_{\text{hi}} + (|o_x \cdot \delta_y| + |\delta_x \cdot o_y| + |\delta_x \cdot \delta_y|) \cdot (4k + 1) \cdot \varepsilon_{\text{hi}}.$$

Function application Suppose that we want to apply a function to a value stored using bit centering, to compute $f(\mathsf{bc}(x))$ where $x = o_x + \delta_x$. Assume that, as we did above, we are able to compute f to machine precision in high-precision arithmetic, and assume that f is Lipschitz continuous with parameter L_f. The natural way to do bit centering for this, following our approach in the linear model case, is to let the offset of the result be computed as

$$o_{f(\mathsf{bc}(x))} \stackrel{\text{def}}{=} f(\mathsf{fl}_{\text{hi}}(o_x)).$$

Next, define the function

$$h(\psi) = f(o_x + \psi) - f(o_x),$$

assume that it is sufficiently nice to approximate, and using our function approximation capabilities, compute a low-precision approximation $\tilde{h} \approx h$ over an appropriate range (either one computed by looking at δ_x or one computed from some *a priori* bound on δ_x). Recall from before that we assumed that given any "sufficiently nice" scalar function $h : \mathbb{R} \to \mathbb{R}$ with $h(0) = 0$ and any range $[-a, a]$ about zero, we can (via some amount of high-precision computation) produce a low-precision-computable routine $\tilde{h}$ that can compute $h(x)$ for any low-precision number $x \in [-a, a]$ to within relative accuracy

$$\left|\tilde{h}(x) - h(x)\right| \leq \zeta \cdot |x| \cdot \max_{u \in [-a,a]} \left|\frac{h(u)}{u}\right| \cdot \varepsilon_{\text{machine-lo}}$$

Once we've computed the approximation $\tilde{h}$, we can use it to compute the delta

$$\delta_{f(\mathsf{bc}(x))} \stackrel{\text{def}}{=} \tilde{h}(\mathsf{fl}_{\text{lo}}(\delta_x)).$$

Note that in order to compute this, we will need to load the value of $\tilde{h}$. This will require additional loads on the accelerator. The resulting computed bit centered value will have error

$$\begin{aligned}
|f(\mathsf{bc}(x)) - f(x)| &= \left|f(\mathsf{fl}_{\text{hi}}(o_x)) + \tilde{h}(\mathsf{fl}_{\text{lo}}(\delta_x)) - (f(o_x) + h(\delta_x))\right| \\
&\leq |f(\mathsf{fl}_{\text{hi}}(o_x)) - f(o_x)| + \left|\tilde{h}(\mathsf{fl}_{\text{lo}}(\delta_x)) - h(\delta_x)\right| \\
&\leq |f(o_x)| \cdot \varepsilon_{\text{machine-hi}} + \zeta \cdot |\delta_x| \cdot \max_{u \in [-a,a]} \left|\frac{h(u)}{u}\right| \cdot \varepsilon_{\text{machine-lo}} \\
&\leq f(o_x) \cdot \varepsilon_{\text{machine-hi}} + \zeta \cdot |\delta_x| \cdot \max_{u \in [-a,a]} \left|\frac{f(o_x + u) - f(o_x)}{u}\right| \cdot \varepsilon_{\text{machine-lo}} \\
&\leq |f(x) + (f(o_x) - f(x)| \cdot \varepsilon_{\text{machine-hi}} + \zeta \cdot |\delta_x| \cdot L_f \cdot \varepsilon_{\text{machine-lo}} \\
&\leq |f(x)| \cdot \varepsilon_{\text{machine-hi}} + L_f \cdot |\delta_x| \cdot \varepsilon_{\text{machine-hi}} + \zeta \cdot |\delta_x| \cdot L_f \cdot \varepsilon_{\text{machine-lo}} \\
&\leq |f(x)| \cdot \varepsilon_{\text{machine-hi}} + (k \cdot \zeta + 1) \cdot L_f \cdot |\delta_x| \cdot \varepsilon_{\text{machine-hi}}.
\end{aligned}$$

The fundamental axiom of bit centered arithmetic Now that we've done this, we can state an analogue of the fundamental axiom of floating point arithmetic, but applied to bit centering. Specifically, for any primitive operator $\star$ (except division), and any bit-centered values x and y

$$|\mathsf{bc}(x) \star \mathsf{bc}(y) - x| \leq |x \star y| \cdot \varepsilon_{\text{machine-hi}} + (O(|\delta_x|) + O(|\delta_y|) + O(|\delta_x \delta_y|)) \cdot \varepsilon_{\text{machine-lo}}$$

where the big-O terms here hide parameters that depend on the operator and possibly on the offsets o_x and o_y. But the important thing to notice here is that the low-precision error *is proportional to the input delta*. So as long as we keep these deltas small, we can get very accurate computations, for *arbitrary* compute graphs.

4.2 Vector and Matrix Operations

While the above analysis does give us a way of bounding the error of any deep neural network gradient computed with bit centering, this bound may be annoying to compute in practice because so far, we have only bounded the error of *scalar* computations, and neural networks in both their forward and backward passes are written in terms of vector and matrix computations. Here, we derive bounds on the error of common vector and matrix computations done using bit centering.

Addition As a consequence of our scalar bounds, the following will hold for any norm for which $|X_{i,j}| \leq Y_{i,j}$ for all indices i and j implies $\|X\| \leq \|Y\|$. Note that this condition will hold for all induced norms, as well as the Frobenius norm.

$$\|\mathsf{bc}(x) + \mathsf{bc}(y) - (x + y)\| \leq \|x + y\| \cdot \varepsilon_{\text{machine-hi}} + \|\delta_x + \delta_y\| \cdot (k + 1) \cdot \varepsilon_{\text{machine-hi}}.$$

Subtraction An analogous result will also hold for subtraction.

$$\|\mathsf{bc}(x) - \mathsf{bc}(y) - (x - y)\| \leq \|x - y\| \cdot \varepsilon_{\text{machine-hi}} + \|\delta_x - \delta_y\| \cdot (k + 1) \cdot \varepsilon_{\text{machine-hi}}.$$

Hadamard product An analogous result will also hold for the Hadamard product (or entrywise product).

$$\|\mathsf{bc}(x) \odot \mathsf{bc}(y) - (x \odot y)\| \leq \|x \odot y\| \cdot \varepsilon_{\text{machine-hi}} + \||o_x \odot \delta_y| + |\delta_x \odot o_y| + |\delta_x \odot \delta_y|\| \cdot (4k + 1) \cdot \varepsilon_{\text{machine-hi}}.$$

Entrywise function application An analogous result will also hold for entrywise function application.

$$\|f(\mathsf{bc}(x)) - f(x)\| \leq \|f(x)\| \cdot \varepsilon_{\text{machine-hi}} + (k \cdot \zeta + 1) \cdot L_f \cdot \|\delta_x\| \cdot \varepsilon_{\text{machine-hi}}.$$

Matrix product So far all our bounds have been trivial applications of our previous scalar bounds in matrix form. The bound for the matrix product is somewhat more complicated. Suppose that we want to do matrix multiplication using bit centering, to compute $\mathsf{bc}(X) \cdot \mathsf{bc}(Y)$ where $X \in R^{m \times n}$ and $Y \in \mathbb{R}^{n \times p}$ and $X = o_X + \delta_X$ and $Y = o_Y + \delta_Y$. The natural way to do this is to let the offset of the result be computed as

$$o_{\mathsf{bc}(X) \cdot \mathsf{bc}(Y)} \stackrel{\text{def}}{=} \mathsf{fl}_{\text{hi}}(o_X) \cdot \mathsf{fl}_{\text{hi}}(o_Y)$$

and the delta be computed as

$$\delta_{\mathsf{bc}(X) \cdot \mathsf{bc}(Y)} \stackrel{\text{def}}{=} \mathsf{fl}_{\text{lo}}(o_X) \cdot \mathsf{fl}_{\text{lo}}(\delta_Y) + \mathsf{fl}_{\text{lo}}(\delta_X) \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \mathsf{fl}_{\text{lo}}(\delta_Y)) .$$

Note that as in the scalar product case, in order to compute this delta, we will need to additionally quantize and load low-precision versions of o_X and o_Y; this will require additional loads on the accelerator. In index i, j, this will have error

$$\begin{aligned}
&|(\mathsf{bc}(X) \cdot \mathsf{bc}(Y))_{i,j} - (X \cdot Y)_{i,j}| \\
&= \left|(\mathsf{fl}_{\text{hi}}(o_X) \cdot \mathsf{fl}_{\text{hi}}(o_Y))_{i,j} + (\mathsf{fl}_{\text{lo}}(o_X) \cdot \mathsf{fl}_{\text{lo}}(\delta_Y))_{i,j} + (\mathsf{fl}_{\text{lo}}(\delta_X) \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \mathsf{fl}_{\text{lo}}(\delta_Y)))_{i,j} - ((o_X + \delta_X) \cdot (o_Y + \delta_Y))_{i,j}\right| \\
&\leq \left|(\mathsf{fl}_{\text{hi}}(o_X) \cdot \mathsf{fl}_{\text{hi}}(o_Y))_{i,j} - (o_X \cdot o_Y)_{i,j}\right| + \left|(\mathsf{fl}_{\text{lo}}(o_X) \cdot \mathsf{fl}_{\text{lo}}(\delta_Y))_{i,j} - (o_X \cdot \delta_Y)_{i,j}\right| \\
&\quad + \left|(\mathsf{fl}_{\text{lo}}(\delta_X) \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \mathsf{fl}_{\text{lo}}(\delta_Y)))_{i,j} - (\delta_X \cdot (o_Y + \delta_Y))_{i,j}\right| \\
&\quad + |(\mathsf{fl}_{\text{lo}}(o_X) \cdot \mathsf{fl}_{\text{lo}}(\delta_Y))_{i,j} + (\mathsf{fl}_{\text{lo}}(\delta_X) \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \mathsf{fl}_{\text{lo}}(\delta_Y)))_{i,j}| \cdot \varepsilon_{\text{machine-lo}}
\end{aligned}$$

where this last term comes from bounding the error of the low-precision sum used to add $(\mathsf{fl}_{\text{lo}}(o_X) \cdot \mathsf{fl}_{\text{lo}}(\delta_Y))_{i,j}$ and $(\mathsf{fl}_{\text{lo}}(\delta_x) \cdot (\mathsf{fl}_{\text{lo}}(o_y) + \mathsf{fl}_{\text{lo}}(\delta_y)))_{i,j}$ when computing the delta term. We can further bound this with

$$\begin{aligned}
&|(\mathsf{bc}(X) \cdot \mathsf{bc}(Y))_{i,j} - (X \cdot Y)_{i,j}| \\
&\quad \leq \left|(\mathsf{fl}_{\text{hi}}(o_X) \cdot \mathsf{fl}_{\text{hi}}(o_Y))_{i,j} - (o_X \cdot o_Y)_{i,j}\right| + \left|(\mathsf{fl}_{\text{lo}}(o_X) \cdot \mathsf{fl}_{\text{lo}}(\delta_Y))_{i,j} - (o_X \cdot \delta_Y)_{i,j}\right| \\
&\qquad + \left|(\mathsf{fl}_{\text{lo}}(\delta_X) \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \mathsf{fl}_{\text{lo}}(\delta_Y)))_{i,j} - (\delta_X \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \mathsf{fl}_{\text{lo}}(\delta_Y)))_{i,j}\right| \\
&\qquad + \left|(\delta_X \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \mathsf{fl}_{\text{lo}}(\delta_Y)))_{i,j} - (\delta_X \cdot (o_Y + \delta_Y))_{i,j}\right| \\
&\qquad + |(\mathsf{fl}_{\text{lo}}(o_X) \cdot \mathsf{fl}_{\text{lo}}(\delta_Y))_{i,j} + (\mathsf{fl}_{\text{lo}}(\delta_X) \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \mathsf{fl}_{\text{lo}}(\delta_Y)))_{i,j}| \cdot \varepsilon_{\text{machine-lo}} \\
&\quad \leq \left|(\mathsf{fl}_{\text{hi}}(o_X) \cdot \mathsf{fl}_{\text{hi}}(o_Y))_{i,j} - (o_X \cdot o_Y)_{i,j}\right| + \left|(\mathsf{fl}_{\text{lo}}(o_X) \cdot \mathsf{fl}_{\text{lo}}(\delta_Y))_{i,j} - (\mathsf{fl}_{\text{lo}}(o_X) \cdot \delta_Y)_{i,j}\right| \\
&\qquad + \left|(\mathsf{fl}_{\text{lo}}(o_X) \cdot \delta_Y)_{i,j} - (o_X \cdot \delta_Y)_{i,j}\right| \\
&\qquad + \left|(\mathsf{fl}_{\text{lo}}(\delta_X) \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \mathsf{fl}_{\text{lo}}(\delta_Y)))_{i,j} - (\delta_X \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \mathsf{fl}_{\text{lo}}(\delta_Y)))_{i,j}\right| \\
&\qquad + \left|(\delta_X \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \mathsf{fl}_{\text{lo}}(\delta_Y)))_{i,j} - (\delta_X \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \delta_Y))_{i,j}\right| \\
&\qquad + \left|(\delta_X \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \delta_Y))_{i,j} - (\delta_X \cdot (o_Y + \delta_Y))_{i,j}\right| \\
&\qquad + |(\mathsf{fl}_{\text{lo}}(o_X) \cdot \mathsf{fl}_{\text{lo}}(\delta_Y))_{i,j} + (\mathsf{fl}_{\text{lo}}(\delta_X) \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \mathsf{fl}_{\text{lo}}(\delta_Y)))_{i,j}| \cdot \varepsilon_{\text{machine-lo}} \\
&\quad \leq \left|(\mathsf{fl}_{\text{hi}}(o_X) \cdot \mathsf{fl}_{\text{hi}}(o_Y))_{i,j} - (o_X \cdot o_Y)_{i,j}\right| + \left|(\mathsf{fl}_{\text{lo}}(o_X) \cdot \mathsf{fl}_{\text{lo}}(\delta_Y))_{i,j} - (\mathsf{fl}_{\text{lo}}(o_X) \cdot \delta_Y)_{i,j}\right| \\
&\qquad + (|o_X| \cdot |\delta_Y|)_{i,j} \cdot \varepsilon_{\text{machine-lo}} \\
&\qquad + \left|(\mathsf{fl}_{\text{lo}}(\delta_X) \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \mathsf{fl}_{\text{lo}}(\delta_Y)))_{i,j} - (\delta_X \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \mathsf{fl}_{\text{lo}}(\delta_Y)))_{i,j}\right| \\
&\qquad + (|\delta_X| \cdot |o_Y + \delta_Y|)_{i,j} \cdot \varepsilon_{\text{machine-lo}} \\
&\qquad + (|\delta_X| \cdot |o_Y|)_{i,j} \cdot \varepsilon_{\text{machine-lo}} \\
&\qquad + |(\mathsf{fl}_{\text{lo}}(o_X) \cdot \mathsf{fl}_{\text{lo}}(\delta_Y))_{i,j} + (\mathsf{fl}_{\text{lo}}(\delta_X) \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \mathsf{fl}_{\text{lo}}(\delta_Y)))_{i,j}| \cdot \varepsilon_{\text{machine-lo}}.
\end{aligned}$$

All the remaining terms that we need to bound are effectively dot products, since computing a single entry of a matrix product *is* a dot product. Recall that the error of a n-dimensional floating point dot product is

$$|\mathsf{fl}(x)^T \mathsf{fl}(y) - x^T y| \leq |x|^T |y| \cdot n \cdot \varepsilon_{\text{machine}},$$

so for a matrix product with inner dimension n, it follows that

$$|(\mathsf{fl}(X)\mathsf{fl}(Y) - XY)_{i,j}| \leq (|X||Y|)_{i.j} \cdot n \cdot \varepsilon_{\text{machine}}.$$

Using this, we can bound our error with

$$\begin{aligned}
&|(\mathsf{bc}(X) \cdot \mathsf{bc}(Y))_{i,j} - (X \cdot Y)_{i,j}| \\
&\quad \leq (|o_X| \cdot |o_Y|)_{i,j} \cdot n \cdot \varepsilon_{\text{machine-hi}} + (|\mathsf{fl}_{\text{lo}}(o_X)| \cdot |\delta_Y|)_{i,j} \cdot n \cdot \varepsilon_{\text{machine-lo}} \\
&\qquad + (|o_X| \cdot |\delta_Y|)_{i,j} \cdot \varepsilon_{\text{machine-lo}} \\
&\qquad + (|\delta_X| \cdot |\mathsf{fl}_{\text{lo}}(o_Y) + \mathsf{fl}_{\text{lo}}(\delta_Y)|)_{i,j} \cdot n \cdot \varepsilon_{\text{machine-lo}} \\
&\qquad + (|\delta_X| \cdot |o_Y + \delta_Y|)_{i,j} \cdot \varepsilon_{\text{machine-lo}} \\
&\qquad + (|\delta_X| \cdot |o_Y|)_{i,j} \cdot \varepsilon_{\text{machine-lo}} \\
&\qquad + |(\mathsf{fl}_{\text{lo}}(o_X) \cdot \mathsf{fl}_{\text{lo}}(\delta_Y))_{i,j} + (\mathsf{fl}_{\text{lo}}(\delta_X) \cdot (\mathsf{fl}_{\text{lo}}(o_Y) + \mathsf{fl}_{\text{lo}}(\delta_Y)))_{i,j}| \cdot \varepsilon_{\text{machine-lo}}.
\end{aligned}$$

Now dropping terms in ε^2 gives us

$$\begin{aligned}
&|(\mathsf{bc}(X) \cdot \mathsf{bc}(Y))_{i,j} - (X \cdot Y)_{i,j}| \\
&\quad \leq (|o_X| \cdot |o_Y|)_{i,j} \cdot n \cdot \varepsilon_{\text{machine-hi}} + (|o_X| \cdot |\delta_Y|)_{i,j} \cdot (n+1) \cdot \varepsilon_{\text{machine-lo}} \\
&\qquad + (|\delta_X| \cdot |o_Y + \delta_Y|)_{i,j} \cdot (n+1) \cdot \varepsilon_{\text{machine-lo}} + (|\delta_X| \cdot |o_Y|)_{i,j} \cdot \varepsilon_{\text{machine-lo}} \\
&\qquad + |(o_X \cdot \delta_Y)_{i,j} + (\delta_X \cdot (o_Y + \delta_Y))_{i,j}| \cdot \varepsilon_{\text{machine-lo}} \\
&\quad \leq (|o_X| \cdot |o_Y|)_{i,j} \cdot n \cdot \varepsilon_{\text{machine-hi}} + (|o_X| \cdot |\delta_Y|)_{i,j} \cdot (n+2) \cdot \varepsilon_{\text{machine-lo}} \\
&\qquad + (|\delta_X| \cdot |o_Y|)_{i,j} \cdot (n+3) \cdot \varepsilon_{\text{machine-lo}} + (|\delta_X| \cdot |\delta_Y|)_{i,j} \cdot (n+2) \cdot \varepsilon_{\text{machine-lo}}.
\end{aligned}$$

And so,

$$\begin{aligned}
&|(\mathsf{bc}(X) \cdot \mathsf{bc}(Y))_{i,j} - (X \cdot Y)_{i,j}| \\
&\quad \leq (|X| \cdot |Y|)_{i,j} \cdot n \cdot \varepsilon_{\text{machine-hi}} + (|o_X| \cdot |\delta_Y|)_{i,j} \cdot (nk + 2k + n) \cdot \varepsilon_{\text{machine-hi}} \\
&\qquad + (|\delta_X| \cdot |o_Y|)_{i,j} \cdot (nk + 3k + n) \cdot \varepsilon_{\text{machine-hi}} + (|\delta_X| \cdot |\delta_Y|)_{i,j} \cdot (nk + 2k + n) \cdot \varepsilon_{\text{machine-hi}} \\
&\quad \leq (|X| \cdot |Y|)_{i,j} \cdot n \cdot \varepsilon_{\text{machine-hi}} + (|o_X| \cdot |\delta_Y| + |\delta_X| \cdot |o_Y| + |\delta_X| \cdot |\delta_Y|)_{i,j} \cdot (nk + 3k + n) \cdot \varepsilon_{\text{machine-hi}}.
\end{aligned}$$

Finally, for any norm of the type described above, we can bound the error with

$$\begin{aligned}
&\|\mathsf{bc}(X) \cdot \mathsf{bc}(Y) - X \cdot Y\| \\
&\quad \leq \||X| \cdot |Y|\| \cdot n \cdot \varepsilon_{\text{machine-hi}} + \||o_X| \cdot |\delta_Y| + |\delta_X| \cdot |o_Y| + |\delta_X| \cdot |\delta_Y|\| \cdot (nk + 3k + n) \cdot \varepsilon_{\text{machine-hi}}.
\end{aligned}$$

Special case: constant matrix product One simpler special case of the matrix product is a product where one of the terms (say, Y) is known *a priori*. In this case, we can set $o_Y = Y$ and $\delta_Y = 0$, and substantially simplify the formula for computing the matrix product in bit centering to

$$o_{\mathsf{bc}(X) \cdot \mathsf{bc}(Y)} \stackrel{\text{def}}{=} \mathsf{fl}_{\text{hi}}(o_X) \cdot \mathsf{fl}_{\text{hi}}(o_Y)$$

and the delta be computed as

$$\delta_{\mathsf{bc}(X) \cdot \mathsf{bc}(Y)} \stackrel{\text{def}}{=} \mathsf{fl}_{\text{lo}}(\delta_X) \cdot \mathsf{fl}_{\text{lo}}(o_Y).$$

In index i, j, this will have error

$$\begin{aligned}
&|(\mathsf{bc}(X) \cdot \mathsf{bc}(Y))_{i,j} - (X \cdot Y)_{i,j}| \\
&\quad = \left|(\mathsf{fl}_{\mathrm{hi}}(o_X) \cdot \mathsf{fl}_{\mathrm{hi}}(o_Y))_{i,j} + (\mathsf{fl}_{\mathrm{lo}}(\delta_X) \cdot \mathsf{fl}_{\mathrm{lo}}(o_Y))_{i,j} - ((o_X + \delta_X) \cdot o_Y)_{i,j}\right| \\
&\quad \leq \left|(\mathsf{fl}_{\mathrm{hi}}(o_X) \cdot \mathsf{fl}_{\mathrm{hi}}(o_Y))_{i,j} - (o_X \cdot o_Y)_{i,j}\right| + \left|(\mathsf{fl}_{\mathrm{lo}}(\delta_X) \cdot \mathsf{fl}_{\mathrm{lo}}(o_Y))_{i,j} - (\delta_X \cdot o_Y)_{i,j}\right| \\
&\quad \leq \left|(\mathsf{fl}_{\mathrm{hi}}(o_X) \cdot \mathsf{fl}_{\mathrm{hi}}(o_Y))_{i,j} - (o_X \cdot o_Y)_{i,j}\right| + \left|(\mathsf{fl}_{\mathrm{lo}}(\delta_X) \cdot \mathsf{fl}_{\mathrm{lo}}(o_Y))_{i,j} - (\delta_X \cdot \mathsf{fl}_{\mathrm{lo}}(o_Y))_{i,j}\right| \\
&\qquad + \left|(\delta_X \cdot \mathsf{fl}_{\mathrm{lo}}(o_Y))_{i,j} - (\delta_X \cdot o_Y)_{i,j}\right| \\
&\quad \leq \left|(\mathsf{fl}_{\mathrm{hi}}(o_X) \cdot \mathsf{fl}_{\mathrm{hi}}(o_Y))_{i,j} - (o_X \cdot o_Y)_{i,j}\right| + \left|(\mathsf{fl}_{\mathrm{lo}}(\delta_X) \cdot \mathsf{fl}_{\mathrm{lo}}(o_Y))_{i,j} - (\delta_X \cdot \mathsf{fl}_{\mathrm{lo}}(o_Y))_{i,j}\right| \\
&\qquad + (|\delta_X| \cdot |o_Y|)_{i,j} \cdot \varepsilon_{\text{machine-lo}}
\end{aligned}$$

Each of these terms is a dot product, so we can bound them with

$$\begin{aligned}
&|(\mathsf{bc}(X) \cdot \mathsf{bc}(Y))_{i,j} - (X \cdot Y)_{i,j}| \\
&\quad \leq (|o_X| \cdot |o_Y|)_{i,j} \cdot n \cdot \varepsilon_{\text{machine-hi}} + (|\delta_X| \cdot |o_Y|)_{i,j} \cdot n \cdot \varepsilon_{\text{machine-lo}} \\
&\qquad + (|\delta_X| \cdot |o_Y|)_{i,j} \cdot \varepsilon_{\text{machine-lo}} \\
&\quad \leq (|X| \cdot |o_Y|)_{i,j} \cdot n \cdot \varepsilon_{\text{machine-hi}} + (|\delta_X| \cdot |o_Y|)_{i,j} \cdot (nk + k + 1) \cdot \varepsilon_{\text{machine-hi}}.
\end{aligned}$$

And so, in terms of any norm of the type described above,

$$\begin{aligned}
&\|\mathsf{bc}(X) \cdot \mathsf{bc}(Y) - X \cdot Y\| \\
&\quad \leq \||X| \cdot |o_Y|\| \cdot n \cdot \varepsilon_{\text{machine-hi}} + \||\delta_X| \cdot |o_Y|\| \cdot (nk + k + 1) \cdot \varepsilon_{\text{machine-hi}}.
\end{aligned}$$

5 Proof of Convergence for BC-SVRG

In this section, we will prove convergence for one algorithm using bit centering: bit centered SVRG. Here, describe the general BC-SVRG algorithm, and prove its convergence for strongly convex objectives.

Algorithm 5 BC-SVRG: Bit-Centered Stochastic Variance-Reduced Gradient

given: N loss functions f_i, number of epochs K, epoch length T, step size α
given: initial iterate $w_1 = o_{w,1} \in \mathbb{R}^d$ in high precision
for $k = 1$ **to** K **do**
  for $i = 1$ **to** N **do**
    compute gradient in HP: $g_{k,i} \leftarrow \nabla f_i(\mathsf{fl}_{\mathrm{hi}}(o_{w,k}))$
    pre-compute and store all necessary data for computing bit-centered delta of ∇f_i later
  end for
  $\bar{g}_k \leftarrow \dfrac{\mathsf{fl}_{\mathrm{hi}}(\alpha)}{\mathsf{fl}_{\mathrm{hi}}(N)} \cdot \sum_{i=1}^{N} \mathsf{fl}_{\mathrm{hi}}(g_{k,i})$
  quantize and store: $\tilde{g}_k \leftarrow \mathsf{fl}_{\mathrm{lo}}(\bar{g}_k)$
  $\delta_{w,k,0} \leftarrow \mathsf{fl}_{\mathrm{lo}}(0) \in \mathbb{R}^d$
  for $t = 1$ **to** T **do**
    sample i uniformly from $\{1, \ldots, N\}$
    load data necessary for computing bit-centered delta of ∇f_i
    compute $u_{k,t} \leftarrow \delta_{\nabla f_i(w_{k,t-1})} \approx \nabla f_i(o_{w,k} + \delta_{w,k,t-1}) - \nabla f_i(o_{w,k})$ **in low precision**
    update model: $\delta_{w,k,t} \leftarrow \mathsf{fl}_{\mathrm{lo}}(\delta_{w,k,t-1}) - \mathsf{fl}_{\mathrm{lo}}(\alpha) \cdot \mathsf{fl}_{\mathrm{lo}}(u_{k,t}) - \mathsf{fl}_{\mathrm{lo}}(\tilde{g}_k)$
  end for
  sample t uniformly from $\{0, \ldots, T-1\}$, then set $o_{w,k+1} \leftarrow \mathsf{fl}_{\mathrm{hi}}(o_{w,k}) + \mathsf{fl}_{\mathrm{hi}}(\delta_{w,k,t})$
end for
return $o_{w,K+1}$

We will prove this algorithm converges at a linear rate, subject to the following assumptions. Note that as before, we are going to ignore terms in $O(\varepsilon^2)$ in this section. First, we assume strong convexity with parameter μ of the objective function $f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$, and Lipschitz continuity of the gradients of each component function $\nabla f_i(w)$ with parameter L. These are standard assumptions used in the analysis of SVRG. Second, we assume that the quantization error of the computation of the delta on Line 5 is always bounded by

$$\|u_{k,t} - \nabla f_i(o_{w,k} + \delta_{w,k,t-1}) - \nabla f_i(o_{w,k})\| \leq C \cdot \|\delta_{w,k,t-1}\| \cdot \varepsilon_{\text{machine-lo}}$$

for some constant parameter C. This will hold (over any bounded region, for some parameter C) for any function ∇f_i that is a composition of the operators we described in the above sections, as a consequence of the fundamental axiom of bit centered arithmetic. Third, we make the simplifying assumption that $\varepsilon_{\text{machine-hi}} = 0$: that the high-precision numbers used are effectively full-precision real-number arithmetic. This is equivalent to ignoring terms in $\varepsilon_{\text{machine-hi}} = 0$, and this is a reasonable assimption when we the high-precision format has enough precision that the accuracy of the output of Algorithm 5

Under these conditions, we can prove the following theorem.

Theorem 1. *Suppose that we run BC-SVRG under the above conditions. Then for any parameter $0 < \gamma < 1$, if we set our step size α and epoch length T to be*

$$\alpha = \frac{\gamma}{4L(1+\gamma)} \qquad T = \frac{8\kappa(1+\gamma)}{\gamma^2},$$

then BC-SVRG converges in expected value at a linear rate given by

$$\mathbf{E}\left[f(o_{w,k+1}) - f(w^*)\right] \leq \left(\gamma + 9 \cdot (23 + 2C) \cdot \kappa \cdot \sqrt{\frac{2}{\gamma}} \cdot \varepsilon_{\textit{machine-lo}} + O(\varepsilon^2_{\textit{machine-lo}})\right) \cdot \mathbf{E}\left[f(o_{w,k}) - f(w^*)\right].$$

Note that in order for this rate to be useful, we need the low-precision format to have epsilon small enough that

$$9 \cdot (23 + 2C) \cdot \kappa \cdot \sqrt{\frac{2}{\gamma}} \cdot \varepsilon_{\text{machine-lo}} < 1.$$

To do this, we need

$$\varepsilon_{\text{machine-lo}} = O(\kappa^{-1}).$$

This theory exposes a relationship between the amount of precision we need and the condition number of the problem. However, as long as this is satisfied, this theorem shows that BC-SVRG can produce solutions of arbitrarily small objective gap—that is, its performance is not bound by the accuracy of the low-precision representation.

Now we will present the proof of the theorem.

Proof of Theorem 1. First, we analyze the error caused by the low-precision updates in the inner loop. First, we notice that, using the $1 + \varepsilon$ model,

$$\begin{aligned}\delta_{w,k,t,j} &= \left(\left(\delta_{w,k,t-1,j} - \alpha \cdot u_{k,t,j} \cdot (1 + \varepsilon_{\text{lo}})\right)(1 + \varepsilon_{\text{lo}}) - \tilde{g}_{k,j}\right)(1 + \varepsilon_{\text{lo}}) \\ &= \delta_{w,k,t-1,j} \cdot (1 + 2\varepsilon_{\text{lo}}) - \alpha \cdot u_{k,t,j}(1 + 3\varepsilon_{\text{lo}}) - \tilde{g}_{k,j}(1 + \varepsilon_{\text{lo}})\end{aligned}$$

where the extra subscript j denotes the jth element of the vector. It follows that

$$\begin{aligned}&\|\delta_{w,k,t} - (\delta_{w,k,t-1} - \alpha \cdot u_{k,t} - \tilde{g}_k)\| \\ &\quad\leq \left(\left(\delta_{w,k,t-1} - \alpha \cdot u_{k,t} \cdot (1 + \varepsilon_{\text{lo}})\right)(1 + \varepsilon_{\text{machine-lo}}) - \tilde{g}_k\right)(1 + \varepsilon_{\text{machine-lo}}) \\ &\quad= 2\|\delta_{w,k,t-1}\| \cdot \varepsilon_{\text{machine-lo}} + 3\alpha \cdot \|u_{k,t}\| \cdot \varepsilon_{\text{machine-lo}} + 3\|\tilde{g}_k\|\varepsilon_{\text{machine-lo}} \\ &\quad= 2\|\delta_{w,k,t-1}\| \cdot \varepsilon_{\text{machine-lo}} + 3\alpha \cdot \|\nabla f_i(o_{w,k} + \delta_{w,k,t-1}) - \nabla f_i(o_{w,k})\| \cdot \varepsilon_{\text{machine-lo}} + 3\|\bar{g}_k\| \cdot \varepsilon_{\text{machine-lo}}\end{aligned}$$

where in the last line we dropped terms in ε^2, as usual. Next, by Lipschitz continuity,

$$\begin{aligned}&\|\delta_{w,k,t} - (\delta_{w,k,t-1} - \alpha \cdot u_{k,t} - \tilde{g}_k)\| \\ &\quad\leq 2\|\delta_{w,k,t-1}\| \cdot \varepsilon_{\text{machine-lo}} + 3\alpha L\|\delta_{w,k,t-1}\| \cdot \varepsilon_{\text{machine-lo}} + 3\|\bar{g}_k\| \cdot \varepsilon_{\text{machine-lo}} \\ &\quad= (2 + 3\alpha L) \cdot \|\delta_{w,k,t-1}\| \cdot \varepsilon_{\text{machine-lo}} + 3\|\bar{g}_k\| \cdot \varepsilon_{\text{machine-lo}}.\end{aligned}$$

Now, we also know that, since $\tilde{g}$ is just a quantization of $\bar{g}$,

$$\|\tilde{g}_k - \bar{g}_k\| \leq \|\bar{g}_k\| \cdot \varepsilon_{\text{machine-lo}},$$

and so adding this to our bound by the triangle inequality gives us

$$\begin{aligned}
&\|\delta_{w,k,t} - (\delta_{w,k,t-1} - \alpha \cdot u_{k,t} - \bar{g}_k)\| \\
&\quad \leq (2 + 3\alpha L) \cdot \|\delta_{w,k,t-1}\| \cdot \varepsilon_{\text{machine-lo}} + 4\|\bar{g}_k\| \cdot \varepsilon_{\text{machine-lo}}.
\end{aligned}$$

Finally, adding in our bound on the quantization error for the delta on Line 5 gives us

$$\begin{aligned}
&\|\delta_{w,k,t} - (\delta_{w,k,t-1} - \alpha \cdot \nabla f_i(o_{w,k} + \delta_{w,k,t-1}) + \alpha \nabla f_i(o_{w,k}) - \bar{g}_k)\| \\
&\quad \leq (2 + 3\alpha L) \cdot \|\delta_{w,k,t-1}\| \cdot \varepsilon_{\text{machine-lo}} + 4\|\bar{g}_k\| \cdot \varepsilon_{\text{machine-lo}} + C \cdot \|\delta_{w,k,t-1}\| \cdot \varepsilon_{\text{machine-lo}} \\
&\quad \leq (2 + 3\alpha L + C) \cdot \|\delta_{w,k,t-1}\| \cdot \varepsilon_{\text{machine-lo}} + 4\|\bar{g}_k\| \cdot \varepsilon_{\text{machine-lo}}.
\end{aligned}$$

Now we have this bound on the error of the update step, we will use it to analyze the expected distance to the optimum of the iterates as the algorithm progresses. At each step, define

$$w_{k,t} = o_{w,k} + \delta_{w,k,t}$$

and define

$$v_{k,t} = o_{w,k} + \delta_{w,k,t-1} - \alpha \nabla f_i(o_{w,k} + \delta_{w,k,t-1}) + \alpha \nabla f_i(o_{w,k}) - \bar{g}_k.$$

Notice that our bound above implies that

$$\|w_{k,t} - v_{k,t}\| \leq (2 + 3\alpha L + C) \cdot \|\delta_{w,k,t-1}\| \cdot \varepsilon_{\text{machine-lo}} + 4\|\bar{g}_k\| \cdot \varepsilon_{\text{machine-lo}}.$$

At the next timestep,

$$\begin{aligned}
\mathbf{E}\left[\|w_{k,t} - w^*\|^2\right] &= \mathbf{E}\left[\|v_{k,t} - w^* + (w_{k,t} - v_{k,t})\|^2\right] \\
&= \mathbf{E}\left[\|v_{k,t} - w^*\|^2 + 2\left(v_{k,t} - w^*\right)^T (w_{k,t} - v_{k,t}) + \|w_{k,t} - v_{k,t}\|^2\right] \\
&= \mathbf{E}\left[\|v_{k,t} - w^*\|^2 + 2\left(v_{k,t} - w^*\right)^T (w_{k,t} - v_{k,t})\right],
\end{aligned}$$

where we dropped the term $\|w_{k,t} - v_{k,t}\|^2$ because our inequality above shows that this is a $O(\varepsilon^2)$ term. Next, by Cauchy-Schwarz, we can further bound this with

$$\begin{aligned}
\mathbf{E}\left[\|w_{k,t} - w^*\|^2\right] &\leq \mathbf{E}\left[\|v_{k,t} - w^*\|^2 + 2\|v_{k,t} - w^*\|\|w_{k,t} - v_{k,t}\|\right] \\
&= \mathbf{E}\left[\|v_{k,t} - w^*\|^2 + 2\|w_{k,t} - w^*\|\|w_{k,t} - v_{k,t}\|\right]
\end{aligned}$$

where in the last line we again dropped $O(\varepsilon^2)$ terms, since $w_{k,t}$ differs from $v_{k,t}$ by only at most a factor of ε. Since $2xy \leq x^2 + y^2$, it follows that for any $\eta > 0$ (a parameter to be set later),

$$\mathbf{E}\left[\|w_{k,t} - w^*\|^2\right] \leq \mathbf{E}\left[\|v_{k,t} - w^*\|^2 + \eta\|w_{k,t} - w^*\|^2 + \frac{1}{\eta}\|w_{k,t} - v_{k,t}\|^2\right].$$

Note that we do not drop the $\|w_{k,t} - v_{k,t}\|^2$ term because later, we will choose η to be proportional to ε so that it cancels out one of the ε terms in that norm. Collecting terms and dividing gives us

$$\begin{aligned}
\mathbf{E}\left[\|w_{k,t} - w^*\|^2\right] &\leq \frac{1}{1-\eta}\mathbf{E}\left[\|v_{k,t} - w^*\|^2 + \frac{1}{\eta}\|w_{k,t} - v_{k,t}\|^2\right] \\
&= \mathbf{E}\left[(1+\eta)\|v_{k,t} - w^*\|^2 + \frac{1}{\eta}\|w_{k,t} - v_{k,t}\|^2\right],
\end{aligned}$$

where the transformation in the last line applies because we will set $\eta = O(\varepsilon)$, and so up to terms in $O(\varepsilon^2)$, it will hold that

$$\frac{1}{1-\eta} = 1 + \eta;$$

this factor of $1+\eta$ disappears from the last term above because that term was already $O(\varepsilon)$.

Now, the term $v_{k,t}$ is exactly what vanilla (non-quantized) SVRG would update its model to. As a result, we can bound $\|v_{k,t} - w^*\|^2$ using exactly the argument used on page 4 of the original SVRG paper Johnson and Zhang [2]. This straightforward derivation will give

$$\mathbf{E}\left[\|v_{k,t} - w^*\|^2 \middle| w_{k,t-1}\right] \leq \|w_{k,t-1} - w^*\|^2 - 2\alpha(1-2L\alpha)\left(f(w_{k,t-1}) - f(w^*)\right) + 4\alpha^2 L\left(f(o_{w,k}) - f(w^*)\right).$$

As a result,

$$\begin{aligned}
&(1+\eta)\mathbf{E}\left[\|v_{k,t} - w^*\|^2 \middle| w_{k,t-1}\right] \\
&\quad\leq (1+\eta)\|w_{k,t-1} - w^*\|^2 - 2(1+\eta)\alpha(1-2L\alpha)\left(f(w_{k,t-1}) - f(w^*)\right) \\
&\qquad + 4(1+\eta)\alpha^2 L\left(f(o_{w,k}) - f(w^*)\right) \\
&\quad\leq \|w_{k,t-1} - w^*\|^2 + \eta\|w_{k,t-1} - w^*\|^2 \\
&\qquad - 2(1+\eta)\alpha(1-2L\alpha)\left(f(w_{k,t-1}) - f(w^*)\right) + 4(1+\eta)\alpha^2 L\left(f(o_{w,k}) - f(w^*)\right) \\
&\quad\leq \|w_{k,t-1} - w^*\|^2 + \eta\frac{2}{\mu}\left(f(w_{k,t-1}) - f(w^*)\right) \\
&\qquad - 2(1+\eta)\alpha(1-2L\alpha)\left(f(w_{k,t-1}) - f(w^*)\right) + 4(1+\eta)\alpha^2 L\left(f(o_{w,k}) - f(w^*)\right) \\
&\quad= \|w_{k,t-1} - w^*\|^2 - 2\left((1+\eta)\alpha(1-2L\alpha) - \eta\mu^{-1}\right)\left(f(w_{k,t-1}) - f(w^*)\right) \\
&\qquad + 4(1+\eta)\alpha^2 L\left(f(o_{w,k}) - f(w^*)\right).
\end{aligned}$$

On the other hand, since $(x+y)^2 \leq 2x^2 + 2y^2$,

$$\begin{aligned}
\|w_{k,t} - v_{k,t}\|^2 &\leq \left((2+3\alpha L + C)\cdot\|\delta_{w,k,t-1}\|\cdot\varepsilon_{\text{machine-lo}} + 4\|\bar{g}_k\|\cdot\varepsilon_{\text{machine-lo}}\right)^2 \\
&\leq \left((2+3\alpha L + C)\cdot\|w_{k,t-1} - o_{w,k}\|\cdot\varepsilon_{\text{machine-lo}} + 4\alpha L\|o_{w,k} - w^*\|\cdot\varepsilon_{\text{machine-lo}}\right)^2 \\
&\leq \left((2+3\alpha L + C)\cdot(\|w_{k,t-1} - w^*\| + \|o_{w,k} - w^*\|)\cdot\varepsilon_{\text{machine-lo}} + 4\alpha L\|o_{w,k} - w^*\|\cdot\varepsilon_{\text{machine-lo}}\right)^2 \\
&\leq \left((2+3\alpha L + C)\cdot\|w_{k,t-1} - w^*\|\cdot\varepsilon_{\text{machine-lo}} + (2+7\alpha L + C)\cdot\|o_{w,k} - w^*\|\cdot\varepsilon_{\text{machine-lo}}\right)^2 \\
&\leq (2+7\alpha L + C)^2\cdot\varepsilon^2_{\text{machine-lo}}\cdot\left(2\|w_{k,t-1} - w^*\|^2 + 2\|o_{w,k} - w^*\|^2\right) \\
&\leq 4(2+7\alpha L + C)^2\mu^{-1}\cdot\varepsilon^2_{\text{machine-lo}}\cdot\left(f(w_{k,t-1}) - f(w^*) + f(o_{w,k}) - f(w^*)\right),
\end{aligned}$$

where the last line follows from strong convexity of f. Combining our bounds together gives us

$$\begin{aligned}
\mathbf{E}\left[\|w_{k,t} - w^*\|^2\right] &\leq \mathbf{E}\left[(1+\eta)\|v_{k,t} - w^*\|^2 + \frac{1}{\eta}\|w_{k,t} - v_{k,t}\|^2\right] \\
&\leq \mathbf{E}\Big[\|w_{k,t-1} - w^*\|^2 - 2\left((1+\eta)\alpha(1-2L\alpha) - \eta\mu^{-1}\right)\left(f(w_{k,t-1}) - f(w^*)\right) \\
&\qquad + 4(1+\eta)\alpha^2 L\left(f(o_{w,k}) - f(w^*)\right) \\
&\qquad + \frac{4(2+7\alpha L + C)^2}{\mu\eta}\cdot\varepsilon^2_{\text{machine-lo}}\cdot\left(f(w_{k,t-1}) - f(w^*) + f(o_{w,k}) - f(w^*)\right)\Big] \\
&= \mathbf{E}\left[\|w_{k,t-1} - w^*\|^2 - \Phi\cdot\left(f(w_{k,t-1}) - f(w^*)\right) + \Psi\cdot\left(f(o_{w,k}) - f(w^*)\right)\right]
\end{aligned}$$

where

$$\Phi = 2\alpha(1-2L\alpha) + 2\eta\left(\alpha(1-2L\alpha) - \mu^{-1}\right) - \frac{4(2+7\alpha L + C)^2}{\mu\eta}\cdot\varepsilon^2_{\text{machine-lo}}$$

and

$$\Psi = 4\alpha^2 L + 4\eta\alpha^2 L + \frac{4(2+7\alpha L + C)^2}{\mu\eta} \cdot \varepsilon^2_{\text{machine-lo}}.$$

Now summing up over all the iterations og an epoch gives us

$$\mathbf{E}\left[\|w_{k,T} - w^*\|^2\right] \leq \mathbf{E}\left[\|w_{k,0} - w^*\|^2 - \Phi \cdot \sum_{t=1}^{T} (f(w_{k,t-1}) - f(w^*)) + \Psi \cdot T \cdot (f(o_{w,k}) - f(w^*))\right],$$

and by the definition of $o_{w,k+1}$,

$$\mathbf{E}\left[\frac{1}{T}\sum_{t=1}^{T} (f(w_{k,t-1}) - f(w^*))\right] = \mathbf{E}\left[(f(o_{w,k+1}) - f(w^*))\right],$$

so

$$\mathbf{E}\left[\|w_{k,T} - w^*\|^2\right] \leq \mathbf{E}\left[\|w_{k,0} - w^*\|^2 - \Phi \cdot T \cdot (f(o_{w,k+1}) - f(w^*)) + \Psi \cdot T \cdot (f(o_{w,k}) - f(w^*))\right].$$

It follows that, since

$$\|w_{k,0} - w^*\|^2 = \|o_{w,k} - w^*\|^2 \leq 2\mu^{-1}(f(o_{w,k}) - f(w^*)),$$

we can conclude that

$$\Phi \cdot T \cdot \mathbf{E}\left[f(o_{w,k+1}) - f(w^*)\right] \leq \left(\Psi T + 2\mu^{-1}\right) \cdot \mathbf{E}\left[f(o_{w,k}) - f(w^*)\right],$$

and so

$$\mathbf{E}\left[f(o_{w,k+1}) - f(w^*)\right] \leq \left(\frac{\Psi}{\Phi} + \frac{2}{\mu\Phi T}\right) \cdot \mathbf{E}\left[f(o_{w,k}) - f(w^*)\right].$$

Now, ignoring terms of ε^2, we can show that

$$\begin{aligned}
\frac{\Psi}{\Phi} &= \frac{4\alpha^2 L + 4\eta\alpha^2 L + \frac{4(2+7\alpha L+C)^2}{\mu\eta} \cdot \varepsilon^2_{\text{machine-lo}}}{2\alpha(1-2L\alpha) + 2\eta\left(\alpha(1-2L\alpha) - \mu^{-1}\right) - \frac{4(2+7\alpha L+C)^2}{\mu\eta} \cdot \varepsilon^2_{\text{machine-lo}}} \\
&= \frac{4\alpha^2 L}{2\alpha(1-2L\alpha)}\left(1 + \frac{4\eta\alpha^2 L + \frac{4(2+7\alpha L+C)^2}{\mu\eta} \cdot \varepsilon^2_{\text{machine-lo}}}{4\alpha^2 L} - \frac{2\eta\left(\alpha(1-2L\alpha) - \mu^{-1}\right) - \frac{4(2+7\alpha L+C)^2}{\mu\eta} \cdot \varepsilon^2_{\text{machine-lo}}}{2\alpha(1-2L\alpha)}\right) \\
&= \frac{4\alpha^2 L}{2\alpha(1-2L\alpha)}\left(1 + \frac{\frac{4(2+7\alpha L+C)^2}{\mu\eta} \cdot \varepsilon^2_{\text{machine-lo}}}{4\alpha^2 L} + \frac{2\eta\mu^{-1} + \frac{4(2+7\alpha L+C)^2}{\mu\eta} \cdot \varepsilon^2_{\text{machine-lo}}}{2\alpha(1-2L\alpha)}\right) \\
&= \frac{4\alpha^2 L}{2\alpha(1-2L\alpha)}\left(1 + \left(\frac{1}{2\alpha L} + \frac{1}{1-2L\alpha}\right) \cdot \frac{2(2+7\alpha L+C)^2}{\alpha\mu\eta} \cdot \varepsilon^2_{\text{machine-lo}} + \frac{\eta}{\alpha\mu(1-2L\alpha)}\right) \\
&= \frac{4\alpha^2 L}{2\alpha(1-2L\alpha)}\left(1 + \left(\frac{1}{2\alpha L(1-2L\alpha)}\right) \cdot \frac{2(2+7\alpha L+C)^2}{\alpha\mu\eta} \cdot \varepsilon^2_{\text{machine-lo}} + \frac{\eta}{\alpha\mu(1-2L\alpha)}\right) \\
&= \frac{2\alpha L}{1-2L\alpha}\left(1 + \frac{(2+7\alpha L+C)^2}{\alpha^2\mu L(1-2L\alpha)\eta} \cdot \varepsilon^2_{\text{machine-lo}} + \frac{\eta}{\alpha\mu(1-2L\alpha)}\right) \\
&= \frac{2\alpha L}{1-2L\alpha} + \frac{2\cdot(2+7\alpha L+C)^2}{\alpha\mu(1-2L\alpha)^2\eta} \cdot \varepsilon^2_{\text{machine-lo}} + \frac{2\eta L}{\mu(1-2L\alpha)^2}
\end{aligned}$$

Now, suppose we set

$$\eta = \frac{(2+7\alpha L + C)}{\sqrt{\alpha L}} \cdot \varepsilon_{\text{machine-lo}}.$$

Then,

$$\frac{\Psi}{\Phi} = \frac{2\alpha L}{1-2L\alpha} + \frac{4\cdot(2+7\alpha L+C)}{\mu(1-2L\alpha)^2}\sqrt{\frac{L}{\alpha}} \cdot \varepsilon_{\text{machine-lo}}.$$

Similarly,

$$\begin{aligned}
\frac{2}{\mu\Phi T} &= \frac{2}{\mu T\left(2\alpha(1-2L\alpha)+2\eta\left(\alpha(1-2L\alpha)-\mu^{-1}\right)-\frac{4(2+7\alpha L+C)^2}{\mu\eta}\cdot\varepsilon^2_{\text{machine-lo}}\right)} \\
&= \frac{1}{\mu\alpha(1-2L\alpha)T}\left(1-\frac{\eta\left(\alpha(1-2L\alpha)-\mu^{-1}\right)-\frac{2(2+7\alpha L+C)^2}{\mu\eta}\cdot\varepsilon^2_{\text{machine-lo}}}{\alpha(1-2L\alpha)}\right) \\
&= \frac{1}{\mu\alpha(1-2L\alpha)T}\left(1-\eta+\frac{\eta}{\alpha\mu(1-2L\alpha)}+\frac{2(2+7\alpha L+C)^2\cdot\varepsilon^2_{\text{machine-lo}}}{\alpha\mu\eta(1-2L\alpha)}\right) \\
&\leq \frac{1}{\mu\alpha(1-2L\alpha)T}+\frac{\eta}{\alpha^2\mu^2(1-2L\alpha)^2T}+\frac{2\cdot(2+7\alpha L+C)^2\cdot\varepsilon^2_{\text{machine-lo}}}{\alpha^2\mu^2\eta(1-2L\alpha)^2T} \\
&= \frac{1}{\mu\alpha(1-2L\alpha)T}+\frac{(2+7\alpha L+C)}{\alpha^2\mu^2(1-2L\alpha)^2T\sqrt{\alpha L}}\cdot\varepsilon_{\text{machine-lo}}\cdot+\frac{2\cdot(2+7\alpha L+C)}{\alpha\mu^2(1-2L\alpha)^2T}\sqrt{\frac{L}{\alpha}}\cdot\varepsilon_{\text{machine-lo}} \\
&= \frac{1}{\mu\alpha(1-2L\alpha)T}+\frac{2+7\alpha L+C}{\alpha\mu^2(1-2L\alpha)^2T}\sqrt{\frac{L}{\alpha}}\left(\frac{1}{\alpha L}+2\right)\cdot\varepsilon_{\text{machine-lo}}.
\end{aligned}$$

Putting these bounds together gives us

$$\begin{aligned}
\frac{\Psi}{\Phi}+\frac{2}{\mu\Phi T} &\leq \frac{2\alpha L}{1-2L\alpha}+\frac{1}{\mu\alpha(1-2L\alpha)T}+\frac{4\cdot(2+7\alpha L+C)}{\mu(1-2L\alpha)^2}\sqrt{\frac{L}{\alpha}}\cdot\varepsilon_{\text{machine-lo}} \\
&\quad+\frac{2+7\alpha L+C}{\alpha\mu^2(1-2L\alpha)^2T}\sqrt{\frac{L}{\alpha}}\left(\frac{1}{\alpha L}+2\right)\cdot\varepsilon_{\text{machine-lo}} \\
&\leq \frac{2\alpha L}{1-2L\alpha}+\frac{1}{\mu\alpha(1-2L\alpha)T}+\frac{2+7\alpha L+C}{\mu(1-2L\alpha)^2}\sqrt{\frac{L}{\alpha}}\left(4+\frac{1}{\alpha^2\mu LT}+\frac{2}{\alpha\mu T}\right)\cdot\varepsilon_{\text{machine-lo}}.
\end{aligned}$$

Next, we substitute the assigned values for α and T,

$$\alpha=\frac{\gamma}{4L(1+\gamma)}\qquad\qquad T=\frac{8\kappa(1+\gamma)}{\gamma^2}.$$

First, notice that

$$\begin{aligned}
\frac{2\alpha L}{1-2L\alpha}+\frac{1}{\mu\alpha(1-2L\alpha)T} &= \frac{2L\frac{\gamma}{4L(1+\gamma)}}{1-2L\frac{\gamma}{4L(1+\gamma)}}+\frac{1}{\mu\frac{\gamma}{4L(1+\gamma)}(1-2L\frac{\gamma}{4L(1+\gamma)})\cdot\frac{8\kappa(1+\gamma)}{\gamma^2}} \\
&= \frac{\gamma}{2+\gamma}+\frac{\gamma(1+\gamma)}{(2+\gamma} \\
&= \gamma.
\end{aligned}$$

For the first part of the error term,

$$\begin{aligned}
\frac{2+7\alpha L+C}{\mu(1-2L\alpha)^2}\sqrt{\frac{L}{\alpha}} &= \frac{2+7L\frac{\gamma}{4L(1+\gamma)}+C}{\mu(1-2L\frac{\gamma}{4L(1+\gamma)})^2}\sqrt{\frac{L}{\frac{\gamma}{4L(1+\gamma)}}} \\
&= \frac{2+7\frac{\gamma}{4(1+\gamma)}+C}{\mu(\frac{2+\gamma}{2(1+\gamma)})^2}\sqrt{\frac{4L^2(1+\gamma)}{\gamma}} \\
&= \frac{8(1+\gamma)^2+7\gamma(1+\gamma)+C(1+\gamma)^2}{\mu(2+\gamma)^2}\cdot L\sqrt{\frac{4(1+\gamma)}{\gamma}} \\
&\leq \frac{16+7+2C}{2}\cdot\kappa\sqrt{\frac{8}{\gamma}},
\end{aligned}$$

where the last line follows from the fact that $0 < \gamma < 1$. Simplifying further,

$$\frac{2 + 7\alpha L + C}{\mu(1 - 2L\alpha)^2}\sqrt{\frac{L}{\alpha}} \leq (23 + 2C) \cdot \kappa\sqrt{\frac{2}{\gamma}}.$$

For the second part of the error term,

$$\begin{aligned} 4 + \frac{1}{\alpha^2\mu LT} + \frac{2}{\alpha\mu T} &= 4 + \frac{1}{\left(\frac{\gamma}{4L(1+\gamma)}\right)^2 \mu L \frac{8\kappa(1+\gamma)}{\gamma^2}} + \frac{2}{\frac{\gamma}{4L(1+\gamma)}\mu\frac{8\kappa(1+\gamma)}{\gamma^2}} \\ &= 4 + 2(1+\gamma) + \gamma \\ &\leq 9, \end{aligned}$$

where again this last line follows from the fact that $0 < \gamma < 1$. So, our whole error term is bounded by

$$\frac{2 + 7\alpha L + C}{\mu(1 - 2L\alpha)^2}\sqrt{\frac{L}{\alpha}}\left(4 + \frac{1}{\alpha^2\mu LT} + \frac{2}{\alpha\mu T}\right) \cdot \varepsilon_{\text{machine-lo}} \leq 9 \cdot (23 + 2C) \cdot \kappa \cdot \sqrt{\frac{2}{\gamma}} \cdot \varepsilon_{\text{machine-lo}}.$$

Substituting this back into our original bound on the convergence of BC-SVRG,

$$\mathbf{E}\left[f(o_{w,k+1}) - f(w^*)\right] \leq \left(\gamma + 9 \cdot (23 + 2C) \cdot \kappa \cdot \sqrt{\frac{2}{\gamma}} \cdot \varepsilon_{\text{machine-lo}}\right) \cdot \mathbf{E}\left[f(o_{w,k}) - f(w^*)\right].$$

Which is what we wanted to prove. □

References

[1] David Bindel. CS 6210 course notes: Dot products in floating point. `http://www.cs.cornell.edu/~bindel/class/cs6210-f12/notes/lec05.pdf`, 2012.

[2] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

[3] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.