Fully Quantized Distributed Gradient Descent

Frederik Kunstner

Supervisors: Prof. Martin Jaggi, Dr. Sebastian Stich

Abstract

In major distributed optimization system, the main bottleneck is often the communication between the different machines. To reduce the time dedicated to communications, some heuristics have been developed to reduce the precision of the messages sent and have been shown to produce good results in practice, and Alistarh et al. [2016] introduced the quantization framework to analyze theoretically the effects of lossy compression on the convergence rate of gradient descent algorithms. This works identifies an issue in one of the proofs in Alistarh et al. [2016] and provides a new approach to reduce the error introduced by low-precision updates.

Contents

1 Introduction

Many optimization problems can be cast as the minimization of a convex function $f : \mathbb{R}^D \to \mathbb{R}$ that can be written as a sum over convex functions $f_1, ..., f_N$, i.e.,

$$x^* = \arg\min_x \frac{1}{N} \sum_{n=1}^{N} f_n(x).$$

In Machine Learning, each function f_n is the error corresponding to the nth sample in a training set, and a classical algorithm to solve this optimization problem is Gradient Descent. Due to the increasing amount of data, the whole training may not fit on a single machine and has to be distributed across multiple nodes to be solved. This creates additional communication between the nodes, which is often the major bottleneck of those systems. This issue has recently been getting more attention, especially from industry, with projects to build systems better adapted to big scale problem Chilimbi et al. [2014], Seide et al. [2014], Strom [2015].

The most closely related line of work is 1-Bit SGD [Seide et al., 2014], which showed experimentally that sending only the sign of each coordinate instead of the full gradients still allowed convergence in some settings. To provide theoretical guarantees on those type of heuristics, Alistarh et al. [2016] introduced the quantization framework which enables the study of the lossy compression of gradient exchanges.

This works identifies an issue in one of the proofs in Alistarh et al. [2016] and provides a new approach to reduce the error introduced by low-precision updates. Our main contribution is the introduction and analysis of a feedback mechanism to correct the error made by the previous quantization, solving the issue and enabling fully quantized algorithms to achieve linear convergence.

This report is structured as follow:

Section 2 gives an overview of gradient descent algorithms for convex problems and the methods used to improve on the computation cost of those algorithms, at the loss of some precision.

Section 3 presents the distributed challenges of the problem and the quantization framework of Alistarh et al. [2016]. We discuss here the issues found in their proof and how it affects the results.

Section 4 introduces our main contribution and its analysis.

Section 5 summarizes our results and discuss possible research directions to improve on our early results.

Furthermore, Appendix 1 presents an idea on how to balance the computation and communication cost of gradient descent methods to achieve a desired accuracy and Appendix 2 list some useful properties of convex functions used throughout the proofs in this report.

2 Background on Convex optimization

Before diving in our main contribution, this section defines the basic optimization problem in a non distributed setting, introduces the notation and gives a quick overview of useful properties and classical methods to solve this problem. A more complete review of the convex optimization methods used here as building blocks can be found in Bubeck [2015], especially sections 4 & 6.

Readers familiar with convex optimization and randomness based methods such as Stochastic Gradient Descent (SGD) and Stochastic Variance Reduced Gradient (SVRG) can feel free to skip this section once past the notation introduction.

2.1 Problem definition

Problem Definition Many optimization problems in Machine Learning can be cast as the minimization of a convex function $f : \mathbb{R}^D \to \mathbb{R}$ that can be written as a sum over convex functions $f_1, ..., f_N$, i.e.,

$$x^* = \arg\min_x \frac{1}{N} \sum_{n=1}^{N} f_n(x).$$

In Machine Learning terminology, each function f_n is the error corresponding to the nth sample in a collection of N samples used to train a model. Some examples of this very general formulation include the Generalized Linear Model family, such as Linear and Logistic Regression, and SVMs.

While convexity is needed to ensure convergence to the global minimum, methods developed for those simple models can also work for non-convex models such as neural networks. As most models in Supervised Learning can be cast as a sum of a cost function over samples, improvements on those basic methods can have a big impact for practitioners.

First order Methods The main idea behind gradient based is to iteratively update an estimate of the parameters by following the negative of the gradient. Given an estimate of parameters at step k, x_k and an estimate of the gradient at x_k, g_k, the update is

$$x_{k+1} = x_k - \gamma_k g_k,$$

where γ_k is a step-size controlling how much we can move per iteration. The step-size depends on the quality of the gradient estimate and on the quality of linear approximation of the gradient at that point.

Function Properties In order to derive convergence rates for optimization methods on this problem, it is useful to make additional assumptions on f such as β-smoothness and α-strong convexity.

Definition 1 (β-smoothness)**.**
We say that a function $f : \mathbb{R}^D \to \mathbb{R}$ is β-smooth if its gradient is β-Lipschitz continuous;

$$\|\nabla f(x) - \nabla f(y)\|^2 \leq \beta \|x - y\|^2, \ \forall x, y \in \mathbb{R}^D.$$

An equivalent condition on f is that

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\beta}{2} \|x - y\|^2, \ \forall x, y \in \mathbb{R}^D.$$

Definition 2 (α-strong convexity)**.**
We say that a function $f : \mathbb{R}^D \to \mathbb{R}$ is α-strongly convex if

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\alpha}{2} \|x - y\|^2, \ \forall x, y \in \mathbb{R}^D.$$

A perhaps more helpful way to view those properties is that they are bounds on the rate of change of a twice differentiable function f. If f is α-strongly convex and β-smooth, then the eigenvalues λ_d, $d \in [D]$ of the Hessian of f, $\nabla^2 f$, are bounded between $0 \leq \alpha \leq \lambda_d \leq \beta$.

Thinking about those properties in one dimension is helpful to get a grasp on their effect. If the second derivative is always smaller than β, this gives a way to control the step-size; if β is small, the second derivative is small and thus the gradient is not changing quickly, meaning we can trust the direction of the gradient and take big steps. On the other hand, if the second derivative is always bigger than α, this ensures that the function is curving fast enough so that we are actually making progress.

As some algorithms we build upon, such as Stochastic Variance Reduced Gradient (SVRG, Johnson and Zhang [2013]), require α-strong convexity, we consider both assumptions for our developments to allow for an easier comparison. Our main contribution, however, can work under only the β-smooth condition with minimal changes to the proof, and is thus more general than presented here.

2.2 Gradient methods

Gradient Descent The simplest algorithm of this family uses the full, exact gradient as the gradient estimate; $g_k = \nabla f(x_k)$. The convergence rate of Gradient Descent (GD) is a classical result in the literature, so the details of the proof will not be presented here, but Section 3 of Bubeck [2015] can be of use to readers unfamiliar with proofs using β-smoothness and α-strong convexity as a building block.

Theorem 3 (Convergence rate of GD (Bubeck [2015], Thm 3.10))**.**
Let $f : \mathbb{R}^D \to \mathbb{R}$ be a α-strongly convex and β-smooth function, and $\kappa = \frac{\beta}{\alpha}$ be the condition number

of the problem. Gradient Descent with a constant step-size of $\gamma = \frac{1}{L}$ satisfies

$$f(y_k) - f^* \leq (1 - \kappa^{-1})(f(y_0) - f^*).$$

Stochastic Gradient Descent One of the main issues of GD is that it is expensive to compute the whole gradient, especially when the number of samples grows. The idea behind stochastic gradient descent is to use a single sample selected uniformly at random to compute the gradient estimate instead of computing the whole gradient. This method still makes progress in expectation, as the expected stochastic gradient is still the full gradient, and is much faster to compute; $O(1)$ in terms of the size of the training set, compared to $O(N)$ for GD.

However, this procedure introduces noise in the gradient estimate, which makes convergence more difficult. For traditional gradient descent, the gradient estimate at the global minimum is 0 and the estimated parameters stop changing, allowing convergence. For stochastic gradient descent, while the expectation of the gradient estimate at the global minimum is 0, the actual stochastic gradients might not be. Therefore, if a constant step size is used, SGD does not converge, unless the global minimum for f is also a minimum for all individual samples $f_n, n \in [N]$. It is possible to enable convergence by using a decreasing step-size γ_k of order $O(1/k)$ such that $\gamma_k g_k$ goes to zero over time, but this change hurts the convergence rate. For strongly convex functions, SGD has a convergence rate of $O(1/k)$ instead of $O(e^{-k})$ for traditional GD (See Bubeck [2015], Thm 6.2).

In practice however, it is often observed that SGD with a constant step-size is very competitive with GD in the early iterations and can exhibit close to linear convergence rate before reaching a plateau. This phenomenon is made explicit in Nedic and Bertsekas [2000] (Proposition 2.4) which shows that with a constant step-size, the error at step k can be decomposed in two terms; one depending on the distance between the initial estimate of the parameters x_0 to the optimal point x^* and decreasing linearly with k, and another depending on a bound on the second moment of the stochastic gradient $\mathbb{E}\left[\|g_k\|^2\right]$ and which does not decrease with k.

Hence, if the stochastic gradients exhibit low variance compared to the current distance to the optimum, it is possible to get good approximate solutions using constant step-size SGD at a fraction of the cost of GD. However, when we get closer to the optimum and the second moment of the stochastic variance is on a order of magnitude similar to the distance to the optimal point, progress becomes more difficult. We call this phenomenon the *Noise Barrier* of SGD, and it is a concept that we will encounter later on when studying quantization based methods.

A common optimization of SGD interesting to our case is the use of mini-batches to reduce the variance of the gradient estimate. Instead of using a single sample to estimate the gradient, an average of B samples is used. This increases the computation cost and reduces the second moment of the stochastic gradient by a factor of B, allowing a trade-off between computational complexity and accuracy of the gradient estimate. A similar trade-off will be shown between the communication cost and the accuracy of the gradient estimate in quantization base methods.

Variance Reduction Techniques Multiple solutions to the Noise Barrier have been developed over the years to combine the linear convergence rate of GD to the cheap cost of stochastic gradient updates. Stochastic Averaged Gradient (SAG) [Le Roux et al., 2017] is one of the earliest technique, which relies on remembering the gradient for every samples and averaging them to reduce the variance while still only computing the gradient for one sample per iteration. The approach used as a building block in this work is Stochastic Variance Reduced Gradient (SVRG) [Johnson and Zhang, 2013], which instead of storing $O(N)$ gradients requires the computation of the full gradient periodically to use as a weighting term for the stochastic gradients to reduce the variance. For the sake of completeness, it is worth mentioning SAGA [Defazio and Bach, 2014], based on the ideas of SAG and SVRG, and Stochastic Dual Coordinate Ascent (SDCA) [Shalev-Shwarts and Zhang, 2013], based on a primal-dual approach.

2.3 Stochastic Variance Reduced Gradient

The SVRG algorithm works in epochs; big iterations consisting of lots of smaller iterations. Contrary to the deep learning literature, epoch in this context does not mean one full pass through the data but simply some specified numbers of iterations. Letting y_k be the parameter estimate at the start of epoch k, SVRG

stores the gradient at that starting point to weight the stochastic gradient. During an epoch, consisting of T iterations, an intermediate estimate of the parameters, $x_t, t \in [T]$, are used. x_0 is initialized at y_k, and the gradient estimate used to update x_t is based on a stochastic part, using the gradient for a sample n selected uniformly at random from $[N]$, and a deterministic part, based on the full gradient at y_k;

$$g_t = \nabla f_n(x_t) - \nabla f_n(y_k) + \nabla f(y_k).$$

After T iterations, the parameter estimate for the next epoch is set at $y_{k+1} = \frac{1}{T}\sum_{t=1}^{T} x_t$. The full procedure is described in Algorithm 1.

Like SGD, the gradient estimate g_t is unbiased; in expectation over the sample selection, $\mathbb{E}[g_t]$ is equal to the gradient at x_t. However, unlike SGD, if the epoch starts at the global minimum x^*, then the gradient estimate is not only 0 in expectation but also for every choice of n, meaning that the procedure is stable at the minimum.

Algorithm 1: One epoch of SVRG

Parameters: a step size γ and number of iterations per epoch T
Requires: the epoch parameter estimate y_k
Result: the next epoch parameter estimate y_{k+1}
Let $G_k = \nabla f(y_k)$
and $x_0 = y_k$
for $t \in [1, T]$ **do**
  Compute the gradient estimate $g_t = \nabla f_{i_t}(x_t) - \nabla f_{i_t}(y_k) + G_k$, where $i_t \sim U[1, N]$
  Update the iteration parameter estimate $x_{t+1} = x_t - \gamma g_t$
end
Update the next epoch parameter estimate $y_{k+1} = \frac{1}{T}\sum_{t=1}^{T} x_t$

As the proof of convergence of SVRG will be used as a building block for the quantized version, it is worth going through it now as the proof are not that contrived. The following derivation is a slight generalization of Theorem 6.5 in Bubeck [2015] to make some of the result easily reusable for the quantization framework. The first step is to see that using the full gradient at the start of the epoch makes it possible to bound the second moment of g_t with the optimality gap of x_t, $f(x_t) - f^*$, and the optimality gap of y_k, $f(y_k) - f^*$.

Lemma 4 (SVRG - Second moment bound of the gradient estimate (Bubeck [2015], First part of Thm 6.5))**.**
Let f be the average of N β-smooth functions $f_1, ..., f_N$. The gradient estimate used in SVRG, $g_t = \nabla f_n(x_t) - \nabla f_n(y_k) + \nabla f(y_k)$ with n selected uniformly at random, has a second moment bounded by

$$\mathbb{E}_n\left[\|g_t\|^2\right] \leq 4\beta(f(x_t) - f^* + f(y_k) - f^*).$$

Proof: The following steps suffice to show the result.

$$\begin{aligned}
\mathbb{E}_n\left[\|g_t\|^2\right] &= \mathbb{E}_n\left[\|\nabla f_n(x_t) - \nabla f_n(y_k) + \nabla f(y_k)\|^2\right], \\
&\overset{(1)}{=} \mathbb{E}_n\left[\|\nabla f_n(x_t) - \nabla f_n(x^*) + \nabla f_n(x^*) - \nabla f_n(y_k) + \nabla f(y_k)\|^2\right], \\
&\overset{(2)}{\leq} 2\mathbb{E}_n\left[\|\nabla f_n(x_t) - \nabla f_n(x^*)\|^2\right] + 2\mathbb{E}_n\left[\|\nabla f_n(y_k) - \nabla f_n(x^*) - \nabla f(y_k)\|^2\right], \\
&\overset{(3)}{\leq} 2\mathbb{E}_n\left[\|\nabla f_n(x_t) - \nabla f_n(x^*)\|^2\right] + 2\mathbb{E}_n\left[\|\nabla f_n(y_k) - \nabla f_n(x^*)\|^2\right], \\
&\overset{(4)}{\leq} 4\beta(f(x_t) - f^* + f(y_k) - f^*).
\end{aligned}$$

- In (1), we simply add and subtract $\nabla f_n(x^*)$.

- In (2), we use the fact that $\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$.
- In (3), we use the fact that $\mathbb{E}_n[\nabla f_n(y_k) - \nabla f_n(x^*)] = \nabla f(y_k) - \nabla f(x^*) = \nabla f(y_k)$ and the standard fact that $\mathbb{E}\left[\|x - \mathbb{E}[x]\|^2\right] \leq \mathbb{E}\left[\|x\|^2\right]$.
- In (4), we use the fact that $\mathbb{E}_n\left[\|\nabla f_n(x) - \nabla f_n(x^*)\|^2\right] \leq 2\beta(f(x) - f^*)$. This is a more general version of the fact that $\|\nabla f(x)\|^2 \leq 2\beta(f(x) - f^*)$ (See Trick 20), which can be found in Bubeck [2015], Lemma 6.4.

□

The next step is to show that it is sufficient for the second moment of the gradient estimate to be bounded by such an expression to enable linear convergence if we set the next epoch parameter estimate to be $y_{k+1} = \frac{1}{T}\sum_{t=1}^{T} x_t$.

Theorem 5 (SVRG - Progress per epoch (Bubeck [2015], Second part of Thm 6.5))**.**
Let f be be a α-strongly convex function and g_t an unbiased gradient estimate with second moment bound $\mathbb{E}\left[\|g_t\|^2\right] \leq C\beta(f(x_t) - f^* + f(y_k) - f^*)$ for some constant C. Then, SVRG with step size γ and epoch length T satisfies

$$\mathbb{E}_{y_{k+1}}[f(y_{k+1}) - f^*] \leq \left(\frac{2}{\alpha T\gamma(2 - \gamma C\beta)} + \frac{\gamma C\beta}{2 - \gamma C\beta}\right)(f(y_k) - f^*).$$

Proof: We will analyze how the distance from the optimum point evolves through iterations within a fixed epoch. We have that

$$\|x_{t+1} - x^*\|^2 = \|x_t - \gamma g_t - x^*\|^2 = \|x_t - x^*\|^2 - 2\gamma\langle g_t, x_t - x^*\rangle + \gamma^2\|g_t\|^2.$$

We can now use our assumptions on g_t to simplify the expression;

$$\begin{aligned}\mathbb{E}_{g_t}\left[\|x_{t+1} - x^*\|^2\right] &\overset{(1)}{\leq} \|x_t - x^*\|^2 - 2\gamma(f(x_t) - f^*) + \gamma^2\mathbb{E}_{g_t}\left[\|g_t\|^2\right], \\ &\overset{(2)}{\leq} \|x_t - x^*\|^2 - 2\gamma(f(x_t) - f^*) + \gamma^2 C\beta(f(x_t) - f^* + f(y_k) - f^*),\end{aligned}$$

Where (1) comes from the unbiasedness of g_t, as $\langle \mathbb{E}_{g_t}[g_t], x_t - x^*\rangle = \langle \nabla f(x_t), x_t - x^*\rangle$ which, from convexity, is lower bounded by $f(x_t) - f^*$, and (2) is the simple application of the second moment bound of g_t. We can apply this inequality recursively on $\|x_t - x^*\|^2$ to get

$$\begin{aligned}\mathbb{E}_{g_0,\ldots,g_{T-1}}\left[\|x_T - x^*\|^2\right] \leq \|x_0 - x^*\|^2 &\\ + (\gamma^2 C\beta - 2\gamma)\sum_{t=0}^{T-1}\mathbb{E}_{g_0,\ldots,g_{T-1}}[f(x_t) - f^*] &\\ + T\gamma^2 C\beta(f(y_k) - f^*).&\end{aligned}$$

As the left hand side is non negative, this inequality implies that

$$(2\gamma - \gamma^2 C\beta)\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}_{g_0,\ldots,g_{T-1}}[f(x_t) - f^*] \leq \frac{1}{T}\|x_0 - x^*\|^2 + \gamma^2 C\beta(f(y_k) - f^*).$$

Using the convexity of f and the definitions of x_0 and y_{k+1}, we have the following relations; that ,

leading to

$$
\begin{aligned}
(2\gamma - \gamma^2 C\beta)\mathbb{E}_{y_{k+1}|y_k}\left[f(y_{k+1}) - f^*\right] &\overset{(1)}{=} (2\gamma - \gamma^2 C\beta)\mathbb{E}_{g_0,\dots,g_{T-1}}\left[f(\frac{1}{T}\sum_{t=0}^{T-1} x_t) - f^*\right] \\
&\overset{(2)}{\leq} (2\gamma - \gamma^2 C\beta)\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}_{g_0,\dots,g_{T-1}}\left[f(x_t) - f^*\right], \\
&\leq \frac{1}{T}\left\|x_0 - x^*\right\|^2 + \gamma^2 C\beta(f(y_k) - f^*), \\
&\overset{(3)}{\leq} \frac{1}{T}\left\|x_0 - x^*\right\|^2 + \gamma^2 C\beta(f(y_k) - f^*), \\
&\overset{(4)}{\leq} \left(\frac{2}{T\alpha} + \gamma^2 C\beta\right)(f(y_k) - f^*).
\end{aligned}
$$

Where

- (1) follows from $y_{k+1} = \frac{1}{T}\sum_{t=0}^{T-1} x_t$,
- (2) uses the convexity of f to have $f(\frac{1}{T}\sum_{t=0}^{T-1} x_t) \leq \frac{1}{T}\sum_{t=0}^{T-1} f(x_t)$,
- (3) uses the definition of $x_0 := y_k$,
- (4) uses the α-strong convexity of f to have $\left\|y_k - x^*\right\|^2 \leq \frac{2}{\alpha}(f(y_k) - f^*)$.

Moving the constants to the right hand side finally leads to

$$
\mathbb{E}_{g_0,\dots,g_{T-1}}\left[f(y_{k+1}) - f^*\right] \leq \left(\frac{2}{T\alpha\gamma(2 - \gamma C\beta)} + \frac{\gamma C\beta}{2 - \gamma C\beta}\right)(f(y_k) - f^*).
$$

□

The combination of Lemma 4 with Theorem 5 directly shows that SVRG (Algorithm 1) with a correct choice of step-size and epoch length can achieve linear convergence. To make the expression found in Theorem 5 independent of α, β and C, it is useful to set $\gamma = O(1/C\beta)$ and $T = O(1/\alpha\gamma)$. Introducing a new parameter η to control the scaling of the step-size and epoch length, γ and T can be rewritten as

$$
\gamma = \frac{\eta}{C\beta} \text{ and } T = \frac{2}{\gamma\alpha} = \frac{2C}{\eta}\frac{\beta}{\alpha},
$$

leading to the following theorem.

Theorem 6 (SVRG - Choice of step-size)**.**
Let $f_1, ..., f_M$ be β-smooth convex functions and $f = \frac{1}{M}\sum_{m=1}^{M} f_m(x)$ be α-stronlgy convex. Then, SVRG (Algorithm 1) with $\gamma = \frac{\eta}{C\beta}, T = \frac{2C}{\eta}\frac{\beta}{\alpha}$ for some scaling parameter η satisfies

$$
\mathbb{E}\left[f(y_K) - f^*\right] \leq \left(\frac{1+\eta}{2-\eta}\right)^K (f(y_0) - f^*).
$$

Hence, $\eta \leq \frac{1}{5}$ leads to a decrease of at least $\frac{2}{3}$ in optimality gap at each epoch.

3 Previous Work on Quantization

This section introduces the problem in its distributed setting and the quantization approach introduced in Alistarh et al. [2016] and Zhang et al. [2016]. As our main contribution builds closely upon the work of Alistarh et al. [2016], we restate here some of their results for clarity.

When the data required to compute f is too large, or simply to be able to compute f in parallel, it has to be distributed among multiple machines such that a machine m can compute f_n for some set of samples, but not others. The traditional GD procedure can be trivially parallelized, with each machine computing the gradient for its samples and sharing it with the other machines. The issue with this approach is that the time required to communicate the gradients, especially for high-dimension problems, is often much

greater than the time required to compute them, making the communication a major bottleneck. This issue has recently been getting more attention, especially from industry, with projects to build systems better adapted to big scale problem [Chilimbi et al., 2014, Seide et al., 2014, Strom, 2015].

The most closely related line of work is 1-Bit SGD [Seide et al., 2014], which showed experimentally that sending only the sign of each coordinate instead of the full gradients still allowed convergence in some settings. The quantization framework by Alistarh et al. [2016] enables the study of algorithms that can use lossy compression of gradients for communication efficiency and still benefit from theoretically proven convergence rates.

The distributed system under consideration is synchronous and distributed, meaning that the machines wait on each other before updating the parameters, and that the update of the parameters is done by all machine in parallel instead of centralized on a single machine. The study of asynchronous system is another interesting possibility to speed up training on large datasets, as it enables machine to continuously work, without having to stop and wait for the other machines to have transmitted their updates. However, asynchronous updates introduce a problem; between the time gradient is computed and applied, the estimate of the parameters might have changed and the gradient no longer match the current estimate. This does not prevent such algorithms to work in practice (see Dean et al. [2012]), but makes the derivation of convergence rate more difficult - see for example Tsitsiklis et al. [1986]. To keep the focus of this work on the analysis of quantization, only synchronized systems are considered here.

3.1 Distributed Problem Definition

The data required to compute the cost function for the samples, $f_n, n \in [N]$ is now distributed across M machines. For simplicity, assume that $[N]$ can be partitioned in M sets, $\mathcal{M}_1, ..., \mathcal{M}_M$, of equal cardinality and consider the following formulation of the optimization problem,

$$f(x) = \frac{1}{M} \sum_{m=1}^{M} f_m(x), \text{ where } f_m(x) = \frac{M}{N} \sum_{n \in \mathcal{M}_m} f_n(x).$$

A single machine m can no longer compute a gradient estimate for f, but only for the samples it has access to, f_m. In order to compute an overall gradient estimate, each machine m computes a local gradient estimate $g_k^{(m)}$ and sends it to every other machine. After this communication step, each machine can compute the gradient estimate $g_k = \frac{1}{M} \sum_{m=1}^{M} g_k^{(m)}$ and apply the gradient step, $x_{k+1} = x_k - \gamma_k x_k$.

3.2 Quantization Paradigm

The main bottleneck in this procedure is the transfer of the local gradient estimates. The idea behind quantization is to transfer a low precision version of these local gradient estimates through a random, lossy compression scheme, thus greatly reducing the communication cost.

To still enable convergence, a quantization operator $Q : \mathbb{R}^D \to \mathbb{R}^D$ is required to be unbiased, $\mathbb{E}_Q [Q(x)] = x$, and have a *bounded variance blowup*, $\mathbb{E}_Q \left[\|Q(x) - x\|^2 \right] \leq b \|x\|^2$, for some value b depending on the lossiness of the compression scheme and perhaps other properties, such as the dimensionality of x, but independent of the selection of x. To reduce communication costs, those quantized gradients should have other properties such as sparsity or nice encoding schemes allowing their communication in few bits. Alistarh et al. [2016] give an example of such a quantization operator (Section 3.2) which, paired with a compression scheme (Appendix A.3), satisfies the necessary convergence properties and has short representations.

Definition 7 (Quantization operator with s levels, Q_s (Alistarh et al. [2016], Section 3.2))**.** *Let $Q_s(x) : \mathbb{R}^D \to \mathbb{R}^D$ be a quantization operator defined as*

$$[Q_s(x)]_d = \|x\| \cdot \text{sign}(x_d) \cdot \xi_d(x, s),$$

where

$$\xi_d(x, s) = \begin{bmatrix} \frac{\lfloor s|x_d|/\|x\| \rfloor}{s} & \text{with probability } 1 - s\frac{|x_d|}{\|x\|} + \left\lfloor s\frac{|x_d|}{\|x\|} \right\rfloor, \\ \frac{\lfloor s|x_d|/\|x\| \rfloor + 1}{s} & \text{otherwise.} \end{bmatrix}$$

For $s = 1$, this quantization scheme reduces to probabilistically rounding up or down each coordinate to the endpoints of the interval $[0, \|g\|]$, with probabilities proportional to the magnitude of the coordinate, and adding the sign information. For $s > 1$, the interval $[0, \|g\|]$ is first partitioned into s equal parts, and for each coordinate a similar rounding is made as for the case when $s = 1$, but in a smaller interval. The following properties have been shown to hold in Alistarh et al. [2016] for s-levels quantization operator.

Lemma 8 (Variance blowup bound of the s-levels quantization operator Q_s (Alistarh et al. [2016], Lemma 3.4))**.**
The variance of the quantization operator with s levels is bounded by

$$\mathbb{E}_{Q_s}\left[\|Q_s(x) - x\|^2\right] \leq \min(D/s^2, \sqrt{D}/s)\,\|x\|^2$$

Lemma 9 (Communication cost of the s-levels quantization operator Q_s (Alistarh et al. [2016], Theorem 3.5))**.**
There exists an encoding scheme so that, in expectation, the number of bits needed to communicate $Q_s(x)$, $x \in \mathbb{R}^D$, is upper bounded by

$$F + \left(\frac{1 + o(1)}{2}\left(\log\left(1 + \frac{s^2(1 + \min(D/s^2, \sqrt{D}/s)}{D}\right) + 1\right) + 2\right) D,$$

where F is the number of bits required to transfer the norm of x. .

In the theory, the convergence proofs for Gradient Descent related algorithms make the implicit assumption that we can transmit numbers with infinite precision, whereas in practice we only transmit $F = 32$ or $F = 64$ bits float approximations. It is however sufficient to enable a convergence up to a precision of ϵ if $F = O(log(1/\epsilon)$, such that the optimality gap guaranteed by theory hold by a multiplicative error of $1 + \text{poly}(\epsilon)$.

When not using quantization however, the number of bits per message is of the order of $O(FD)$, where F is the precision of the floating points and D the number of dimensions. Quantization has the added benefit of separating the error made by the floating point approximation and the compression. Taking $s = \sqrt{D}$ levels, we can transmit only $O(F + D)$ bits per message at the cost of a variance blowup of 2. This makes it possible to achieve the information theoretical lower bound on the communication complexity of convex optimization within ϵ-precision of $\Omega(D(\log D + \log(1/\epsilon)))$ [Tsitsiklis and Luo, 1987]. A more precise development of this argument is made in Alistarh et al. [2016].

3.3 Earlier results

Given Lemma 8 and the previous discussion on SGD, it is easy to see that SGD coupled with quantization still works, although the step-size should to be reduced to account for the reduction in accuracy due to quantization. The more interesting use of quantization is in conjunction with SVRG to achieve linear rate.

As a first step, we can focus on quantizing only the stochastic gradients part of the communication. The synchronization step at the start of each epoch is still required to compute the gradient at y_k to anchor the stochastic gradients, but the exchange of the stochastic gradient, which represent the vast majority of the communication of SVRG, can now be quantized. This procedure is shown in more details in Algorithm 2.

Given Theorem 5, we only need to show that the second moment of the gradient estimate can be bounded in a similar fashion as in Lemma 4. To make the analysis slightly easier notation wise, the following derivation considers a simplified version where only one machine sends a quantized stochastic gradient in each iteration. The analysis is still valid when each machine contributes a sample, as the result is only a reduction in the variance of the gradient estimate.

Lemma 10 (Second moment bound of g_t in QSVRG, (Alistarh et al. [2016], Thm XX))**.**
Let f be the average of N β-smooth function $f_1, ..., f_N$. The gradient estimate for quantized SVRG

with s levels, $g_t = Q_s(\nabla f_n(x_t) - \nabla f_n(y_k) + \nabla f(y_k))$ with n selected uniformly at random, has a second moment bounded by

$$\mathbb{E}\left[\|g_t\|^2\right] \leq \left(1 + \frac{\sqrt{D}}{s}\right) 4\beta(f(x_t) - f^* + f(y_k) - f^*).$$

Proof: The proof is straightforward given Lemmas 4 and 8;

$$\begin{aligned}
\mathbb{E}\left[\|g_t\|^2\right] &= \mathbb{E}\left[\|Q_s(\nabla f_{i_t}(x_t) - \nabla f_{i_t}(y_k) + \nabla f(y_k))\|^2\right], \\
&\overset{(1)}{\leq} \left(1 + \frac{\sqrt{D}}{s}\right) \mathbb{E}\left[\|\nabla f_{i_t}(x_t) - \nabla f_{i_t}(y_k) + \nabla f(y_k)\|^2\right], \\
&\overset{(2)}{\leq} \left(1 + \frac{\sqrt{D}}{s}\right) 4\beta(f(x_t) - f^* + f(y_k) - f^*).
\end{aligned}$$

Where (1) is direct application of Lemma 8 and (2) a direct application of Lemma 4 □

Algorithm 2: One epoch of QSVRG, Alistarh et al. [2016]

Parameters: A step size $\gamma \in \mathbb{R}_+$ and an epoch length $T \in \mathbb{N}$.
Requires: estimate of the parameter at the start of the current epoch y_k.
Result: estimate of the parameter at the end of the current epoch y_{k+1}.

Compute the gradient at y_k:
  Each machine m compute $G_k^{(m)} = \nabla f_m(y_k)$ and send it to every other machine.
  Once each message has been received, compute $G_k = \frac{1}{M}\sum_{m=1}^{M} G_k^{(m)} \equiv \nabla f(y_k)$.

Let $x_0 = y_k$.

for $t \in [1, T]$ **do**
  Compute the gradient estimate, g_t:
    Each machine m chooses a sample i_m uniformly at random from the samples it holds.
    Send $g_t^{(m)} = Q_s(\nabla f_{i_m}(x_t) - \nabla f_{i_m}(y_k) + G_k)$ to every other machine.
    Once each message has been received, compute $g_t = \frac{1}{M}\sum_{m=1}^{M} g_t^{(m)}$.

  Apply the gradient step:
    Let $x_{t+1} = x_t - \gamma g_t$.
end

Compute the epoch estimate y_k:
  Let $y_{k+1} = \frac{1}{T}\sum_{t=1}^{T} x_t$.

3.4 Issue for fully quantized method in distributed setting

The previous sections shows that quantization can be used on the stochastic gradients and introduces a controllable increase in the second moment of the gradient estimates. However, the setup described still requires the communication of the full precision at the start of the epoch to weight the stochastic gradients down. Under the assumption that QSVRG is only performed on one machine at a time, for the same reason as we analyze only one sample contribution for Lemma 10, and thus assuming that the gradient estimate is of the form

$$Q_s\left(\nabla f_n(x_t) - \nabla f_n(y_k) + \underbrace{Q_s(\nabla f(y_k))}_{\text{Epoch Gradient (one machine)}}\right),$$

Alistarh et al. [2016], [Version 3 on Arxiv], concluded that fully quantized SVRG (with quantized epoch gradients) converges linearly. Their proof works[1] for this setting but oversees the fact that in a distributed setting, the computation of the epoch gradient is also distributed and the gradient estimate is of the form

$$Q_s\left(\nabla f_n(x_t) - \nabla f_n(y_k) + \underbrace{\frac{1}{M}\sum_{m=1}^{M} Q_s(\nabla f_m(y_k))}_{\text{Epoch Gradient } (M \text{ machines})}\right).$$

In this form, the analysis provided in Alistarh et al. [2016] can no longer be applied.

To see why this is an issue, consider the epoch gradient on one machine when the epoch starts at the solution, such that $\nabla f(y_k) = 0$. In this setting, the quantization of the gradient will also output 0, and the quantized epoch gradient will be correct. In the multiple machine setting, however, this is no longer guaranteed to be the case. Consider this example in two dimensions with two machines where the gradient at the solution for the first machine and second machine are

$$\nabla f_1(x^*) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \nabla f_2(x^*) = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

The gradient of f is obviously 0, but using a one-level quantization operator on these gradient, the following quantization is possible

$$Q_1(\nabla f_1(x^*)) = \sqrt{2}\begin{bmatrix} 1 \\ 0 \end{bmatrix}, Q_1(\nabla f_2(x^*)) = \sqrt{2}\begin{bmatrix} 0 \\ -1 \end{bmatrix},$$

leading to an non-zero epoch gradient estimate. To deal with this issue and get a fully quantized distributed algorithm with linear convergence rate, we introduce new techniques to control the variance introduced by the quantization in the next section.

4 Main Contribution

In order to solve the Noise Barrier introduced by quantization, we present here a way to reduce the variance of the gradient estimate using only quantized communications. To introduce our contribution, consider the simpler problem of trying to communicate a vector x from one machine to another using only the exchange of quantized vectors. For notation purposes, we will denote by Q_b a quantization operator with variance blowup bound b (see Lemma 8). The simplest methods would be to send K independent quantizations of x, $x_1, ..., x_K$, and use their average as an estimate of x, leading to a sublinear convergence rate of $O(1/K)$,

$$\hat{x} = \frac{1}{K}\sum_{k=1}^{K} x_k,\ \mathbb{E}\left[\|\hat{x} - x\|^2\right] \leq \frac{1}{K}\mathbb{E}\left[\|Q_b(x) - x\|^2\right] \leq \frac{b}{K}\mathbb{E}\left[\|x\|^2\right].$$

Given a precise enough quantization, say $b < 1$, we could however come up with a better scheme giving more progress per bit. Consider the following iterative scheme where, starting at some x_0, we send the quantized difference of the current iterate estimate and x instead, and the iterative is updated with this difference such that

$$x_{t+1} = x_t + Q_b(x - x_t).$$

Following Lemma 8, this scheme allows linear convergence if $b < 1$ as

$$\mathbb{E}\left[\|x_T - x\|^2\right] = \mathbb{E}\left[\|x_{T-1} + Q_b(x - x_{T-1}) - x\|^2\right] \leq b\mathbb{E}\left[\|x_{T-1} - x\|^2\right] \leq b^T\mathbb{E}\left[\|x_0 - x\|^2\right].$$

While the quantization method introduced in Alistarh et al. [2016] could possibly be improved in terms of the constants involved, their scheme is asymptotically optimal in that if we were able to make

[1] For the sake of completeness, a smaller issue with the analysis is that the epoch gradient are transmitted only once at the start of the epoch. The errors made during each iteration are therefore not independent since they all rely on the same slightly wrong approximation of the epoch gradient. This problem can however be easily fixed by sending a fresh quantization of the epoch gradient at each iteration, which only changes the communication cost by a factor of 2.

more progress than mentioned above, up to multiplicative constants, this would violate the lower bound on the communication complexity of convex optimization [Tsitsiklis and Luo, 1987]. As a sanity check, observe that running a more precise quantization scheme is more efficient than running multiple less precise quantization schemes. In terms of precision, running one $Q_{\frac{1}{16}}$ update is equivalent to running two $Q_{\frac{1}{4}}$ update. However, in terms of communication cost, the more precise option wins. Using Lemma 8, we see that $b = 1/16$ is equivalent to $s = 4\sqrt{D}$ and $b = 1/4$ is equivalent to $s = 2\sqrt{D}$. Using 9, we see that the precise update has a cost of $F + \log(\sqrt{18})D$, and the two low-precision updates have a cost of $2F + \log(6)D > F + \log(\sqrt{18})D$.

This scheme is illustrated here with a fixed x, but the goal is to use it to approximate the gradient of each machine in distributed gradient descent. Instead of transmitting a quantization of the gradient at machine m, only the difference between the current estimate of the gradient at machine m and the true gradient is quantized and sent. Hand-waving a bit; assuming the method stabilizes at some point, the difference between the gradient estimate of machine m and its true gradient will shrink, and the variance introduced by quantization will also go down, allowing convergence. We make this idea more rigorous in the next section.

4.1 Distributed Quantized GD Algorithm

Our main contribution is a Distributed Gradient Descent algorithm which only transfers quantized vectors and still achieves linear convergence rate. We start with a GD based method to avoid dealing with stochasticity as a start, but our method can be extended to work with SVRG, as shown in [Appendix XX].

For simplicity, assume that, at the start of the algorithm, we are given an initial point y_0 and each machine knows the exact value of the gradient of each other machine, stored in a variable $G(0, m) = \nabla f_m(y_0)$. At the start of iteration k, all machines know y_k and have an estimate of the gradient of every machine at iteration $k-1$ stored in $G_{k-1}^{(m)}$. The gradient estimates for each machine will be updated by sending a quantization of bounded variance blowup b, Q_b, of the difference between the gradient at step y_k and the gradient estimate at iteration $k-1$;

$$G_k^{(m)} = G_{k-1}^{(m)} + Q_b(\nabla f_m(y_k) - G_{k-1}^{(m)}).$$

The gradient estimate for each machine are then averaged to produce the gradient estimate at step k;

$$g_k = \frac{1}{M}\sum_{m=1}^{M} G_k^{(m)} = \frac{1}{M}\sum_{m=1}^{M} G_{k-1}^{(m)} + Q_b(\nabla f_m(y_k) - G_{k-1}^{(m)}).$$

The procedure is explained in details in Algorithm 3.

The convergence of this algorithm is not obvious from the previous example, as the target was a fixed point whereas we are now trying to approximate the gradient, which is changing at each iteration. We will show that if the quantization is sufficiently precise, i.e., b is sufficiently small, then the estimation of the gradient for each machine "catches up" with the changing gradient fast enough to still enable linear convergence. We call this variant Quantized Distributed Gradient Descent with Feedback (QDGD-F) and refer to the case where only the quantized gradients are sent as QDGD.

4.2 Convergence proof

The key insight of the per-machine update rule defined above is that it is possible to bound the second moment of the gradient estimate by a term depending on the true gradient at the current point and isolate the error related to quantization. The error of the previous iterations carry over to the current one, but those get slightly corrected over time such that the error made in the early iterations vanish at a linear rate depending on the quantization precision b.

Lemma 11 (Second moment bound for the gradient estimate in QDGD-F)**.**
Let f be the average of M β-smooth convex functions, $f_1, ..., f_M$, and Q_b a quantization operator with a variance blowup bound of b. Then, the gradient estimate used in QDGD-F,

$$g_k = \frac{1}{M}\sum_{m=1}^{M} G_k^{(m)},\ G_k^{(m)} = G_{k-1}^{(m)} + Q_b(\nabla f_m(x_k) - G_{k-1}^{(m)}),$$

Algorithm 3: One iteration of Quantized Distributed Gradient Descent with Feedback (QDGD-F)

Parameters: A step size $\gamma \in \mathbb{R}_+$.
Requires:

- An estimate of the parameter at the start of the current step x_k,
- An estimate of the gradient of each machine at the previous step, $G_{k-1}^{(m)}$, $m \in [M]$.

Result:

- An estimate of the parameter at the end of the next step x_{k+1},
- An estimate of the gradient of each machine at the current step, $G_k^{(m)}$, $m \in [M]$.

Estimate the gradients for every machine at y_k:
Each machine m computes
$\Delta G_k^{(m)} = Q_b(\nabla f_m(x_k) - G_{k-1}^{(m)})$
and send it to every other machine.
Once each message has been received, compute the machines gradient estimate,
$G_k^{(m)} = G_{k-1}^{(m)} + \Delta G_k^{(m)}$, $m \in [M]$.
Apply the gradient step:
Let $x_{k+1} = x_k - \gamma \frac{1}{M} \sum_{m=1}^{M} G_k^{(m)}$.

where $G_0^{(m)}$ is initialized with $\nabla f_m(x_0)$, has a second moment bounded by

$$\mathbb{E}\left[\|g_k\|^2\right] \leq \|\nabla f(x_k)\|^2 + 4\beta \left[\sum_{j=0}^{k-1} b^{k-j}(f(y_j) - f^*) + b\sum_{j=1}^{k} b^{k-j}(f(y_j) - f^*)\right].$$

The proof is based on variance blowup bound of the quantization and uses a technique similar as the SVRG proof to bound the difference between gradients by the optimality gap at those points.

Proof:

Separating $\mathbb{E}\left[\|g_k\|^2\right]$ into the norm of the gradient and the approximation error
As a first step, we can separate the second moment of g_k a term depending on the magnitude of the true gradient at x_k and the approximation error of the quantization procedure,

$$\begin{aligned}\mathbb{E}\left[\|g_k\|^2\right] &\overset{(1)}{=} \mathbb{E}\left[\|g_k - \nabla f(x_k) + \nabla f(x_k)\|^2\right], \\ &\overset{(2)}{=} \|\nabla f(x_k)\|^2 + \mathbb{E}\left[\|g_k - \nabla f(x_k)\|^2\right],\end{aligned}$$

where (1) simply adds and remove the gradient at x_k and (2) uses Trick 19. To make the analysis easier, we can on the second moment of the approximation of each machine individually using the following rough upper bound

$$\begin{aligned}\mathbb{E}\left[\|g_k - \nabla f(x_k)\|^2\right] &= \mathbb{E}\left[\left\|\frac{1}{M}\sum_{m=1}^{M} G_k^{(m)} - \frac{1}{M}\sum_{m=1}^{M} \nabla f_m(x_k)\right\|^2\right], \\ &\overset{(3)}{\leq} \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}\left[\left\|G_k^{(m)} - \nabla f_m(x_k)\right\|^2\right],\end{aligned}$$

where (3) uses Trick 22. Using the second moment bound of the quantization operator, we get

$$\begin{aligned}\mathbb{E}\left[\left\|G_k^{(m)} - \nabla f_m(x_k)\right\|^2\right] &\overset{(4)}{=} \mathbb{E}\left[\left\|G_{k-1}^{(m)} + Q_b(\nabla f_m(x_k) - G_{k-1}^{(m)}) - \nabla f_m(x_k)\right\|^2\right], \\ &\overset{(5)}{\leq} b\mathbb{E}\left[\left\|G_{k-1}^{(m)} - \nabla f_m(x_k)\right\|^2\right],\end{aligned}$$

where (4) simply expands the definition of $G_k^{(m)}$ and (5) uses Lemma 8. We can also make explicit the variance depending on the approximation done during the previous iteration as

$$\begin{aligned}\mathbb{E}\left[\left\|G_{k-1}^{(m)} - \nabla f_m(x_k)\right\|^2\right] &\overset{(6)}{=} \mathbb{E}\left[\left\|G_{k-1}^{(m)} - \nabla f_m(x_{k-1}) + \nabla f_m(x_{k-1}) - \nabla f_m(x_k)\right\|^2\right], \\ &\overset{(7)}{\leq} \mathbb{E}\left[\left\|G_{k-1}^{(m)} - \nabla f_m(x_{k-1})\right\|^2\right] \\ &\quad + \left\|\nabla f_m(x_{k-1}) - \nabla f_m(x_k)\right\|^2,\end{aligned}$$

where (6) adds and remove the gradient of machine m at x_{k-1} and (7) uses Trick 19.

Full expression
Applying the previous steps recursively, we get the following formulation for the second moment;

$$\mathbb{E}\left[\|g_k\|^2\right] \leq \|\nabla f(x_k)\|^2 + \frac{1}{M}\sum_{m=1}^{M}\sum_{k=0}^{k-1} b^{k+1}\left\|\nabla f_m(x_{k-k}) - \nabla f_m(x_{k-k-1})\right\|^2.$$

Relation to the optimality gap
We can relate the difference in gradients to the value of the function at those points;

$$\begin{aligned}\frac{1}{M}\sum_{m=1}^{M}\left\|\nabla f_m(y_k) - \nabla f_m(y_{k-1})\right\|^2 &\overset{(8)}{=} \mathbb{E}_{m\sim\text{Unif[M]}}\left[\left\|\nabla f_m(y_k) - \nabla f_m(y_{k-1})\right\|^2\right], \\ &\overset{(9)}{=} \mathbb{E}_m\left[\left\|\nabla f_m(y_k) - \nabla f_m(x^*) + \nabla f_m(x^*) - \nabla f_m(y_{k-1}\right\|^2\right], \\ &\overset{(10)}{\leq} 2\mathbb{E}_m\left[\left\|\nabla f_m(y_k) - \nabla f_m(x^*)\right\|^2\right] \\ &\quad + 2\mathbb{E}_m\left[\left\|\nabla f_m(y_{k-1}) - \nabla f_m(x^*)\right\|^2\right], \\ &\overset{(11)}{\leq} 4\beta\left[f(y_k) - f^* + f(y_{k-1}) - f^*\right]\end{aligned}$$

(8) interprets the average over the M machines as an expectation over m, taking uniformly at random. (9) then adds and remove the gradient of f_m at the optimal point of f, (10) separates the norm using Trick 22 and (11) uses Trick 21 to relate

Final expression
Plugging this back in the bound we had, we get

$$\mathbb{E}\left[\|g_k\|^2\right] \leq \|\nabla f(x_k)\|^2 + 4\beta\left[\sum_{j=0}^{k-1} b^{k-j}(f(y_j) - f^*) + b\sum_{j=1}^{k} b^{k-j}(f(y_j) - f^*)\right].$$

□

Given this bound, we can show that under some conditions on the precision b, the convergence rate of QDGD deteriotates, when compared to GD, but stays linear.

Theorem 12 (Convergence of QDGD)**.**
Let f, the average of M β-smooth convex functions, $f_1, ..., f_M$, be an α-strongly convex function and Q_b be a quantization operator with a variance blowup bound of b. Using the following gradient

estimate

$$g_k = \frac{1}{M}\sum_{m=1}^{M} G_k^{(m)},\ G_k^{(m)} = G_{k-1}^{(m)} + Q_b(\nabla f_m(x_k) - G_{k-1}^{(m)},$$

where $G_0^{(m)}$ *is initialized with* $\nabla f_m(x_0)$ *and applying iteratively the update*

$$x_{k+1} = x_k - \gamma g_k$$

with a step-size $\gamma = \frac{1}{\beta}$ *lead to a linear convergence rate if* $b < \frac{4}{13\kappa+1}$*, such that*

$$\mathbb{E}\left[f(x_k) - f^*\right] \leq c^k (f(x_0) - f^*),$$

Where $c = \frac{1}{2}(\sqrt{(1-\kappa^{-1}+3b)^2 + b(1+\kappa^{-1})} + (1-\kappa^{-1}+3b))$.

Proof:

Bounding the optimality gap

For notation simplicity, we introduce $h(x) = f(x) - f^*$ for the optimality gap at x. Using the β-smoothness of f, we can upper bound the error gap at x_k by

$$f(x_{k+1}) \leq f(x_k) - \gamma\langle \nabla f(x_k), g_k\rangle + \gamma^2 \frac{\beta}{2}\left\|g_k\right\|^2.$$

Taking the expectation with respect to all random variables and applying the previous inequality to the inequality gap, we get

$$\mathbb{E}\left[h(x_{k+1})\right] \leq \mathbb{E}\left[h(x_k) - \gamma\left\|\nabla f(x_k)\right\|^2 + \gamma^2\frac{\beta}{2}\left\|g_k\right\|^2\right].$$

Using Lemma 11, we can upper bound the second moment of g_k to get

$$\begin{aligned}\mathbb{E}\left[h(x_{k+1})\right] \leq\ & \mathbb{E}\left[h(x_k)\right] - \gamma\left\|\nabla f(x_k)\right\|^2 \\ & + \gamma^2\frac{\beta}{2}\left(\left\|\nabla f(x_k)\right\|^2 + 4\beta\left[\sum_{j=0}^{k-1} b^{k-j}h(x_j) + b\sum_{j=1}^{k} b^{k-j}h(x_j)\right]\right).\end{aligned}$$

Grouping the terms in $\left\|\nabla f(x_k)\right\|^2$ and assuming that $0 \leq \gamma \leq \frac{1}{\beta}$ (such that $\frac{\beta}{2}\gamma^2 - \gamma$ is negative), we can use the lower bound $\left\|\nabla f(x_k)\right\|^2 \geq 2\mu h(x_k)$ (see Trick 21) to get

$$\begin{aligned}\mathbb{E}\left[h(x_{k+1})\right] \leq\ & \mathbb{E}\left[h(x_k) + \left(\beta\gamma^2 - 2\gamma\right)\mu h(x_k)\right] \\ & + 2\gamma^2\beta^2\mathbb{E}\left[\left[\sum_{j=0}^{k-1} b^{k-j}h(x_j) + b\sum_{j=1}^{k} b^{k-j}h(x_j)\right]\right].\end{aligned}$$

To simplify the expression, set $\gamma = \frac{1}{\beta}$ and use $\kappa = \frac{\beta}{\mu}$ to denote the maximum condition number of f. This allows us to rewrite the previous expression as

$$\mathbb{E}\left[h(x_{k+1})\right] \leq \mathbb{E}\left[(1-\kappa^{-1})h(x_k) + 2\left[\sum_{j=0}^{k-1} b^{k-j}h(x_j) + b\sum_{j=1}^{k} b^{k-j}h(x_j)\right]\right].$$

Proving linear convergence

To show that the optimality gap decreases linearly with k, we will perform a proof by induction; assuming that the optimality decreases by a factor of $c < 1$ at each iteration $1, ..., k-1$, then it decreases by a factor of c again at iteration k. It is easy to see that the base case holds for $(1-\kappa^{-1}) \leq c \leq 1$; as we start with the correct gradient in memory, no quantization variance is introduced into the mix and the first step is simply a gradient descent step. We will now show that if it holds for $1, ..., k-1$, then it also holds for k.

Finding the constraints on c

Assuming that $\mathbb{E}\left[h(x_k)\right] \leq c\mathbb{E}\left[h(x_{k-1})\right]$, we can rewrite the bound as

$$\begin{aligned}
\mathbb{E}\left[h(x_{k+1})\right] &\leq \left((1-\kappa^{-1})c^k h(x_0) + 2\left[\sum_{j=0}^{k-1} b^{k-j}c^j h(x_0) + b\sum_{j=1}^{k} b^{k-j}c^j h(x_0)\right]\right), \\
&\overset{(1)}{=} \left((1-\kappa^{-1}) + \frac{2}{c^k}\left[\sum_{j=0}^{k-1} b^{k-j}c^j + b\sum_{j=1}^{k} b^{k-j}c^j\right]\right) c^k h(x_0), \\
&\overset{(2)}{=} \left((1-\kappa^{-1}) + 2\frac{b^k}{c^k}\left[\sum_{j=0}^{k-1} \left(\frac{c}{b}\right)^j + b\sum_{j=1}^{k} \left(\frac{c}{b}\right)^j\right]\right) c^k h(x_0),
\end{aligned}$$

where (1) puts $h(x_0)c^k$ in evidence and (2) puts b^k in evidence in the inner sum. We can simplify the summation terms as follow;

$$\begin{aligned}
\frac{b^k}{c^k}\left[\sum_{j=0}^{k-1} \left(\frac{c}{b}\right)^j + b\sum_{j=1}^{k} \left(\frac{c}{b}\right)^j\right] &\overset{(3)}{=} \frac{b^k}{c^k}\left[\frac{\left(\frac{c}{b}\right)^k - 1}{\frac{c}{b}-1} + b\frac{c}{b}\frac{\left(\frac{c}{b}\right)^k - 1}{\frac{c}{b}-1}\right], \\
&\overset{(4)}{=} \frac{b^k}{c^k}\left(\left(\frac{c}{b}\right)^k - 1\right)\frac{c+1}{\frac{c}{b}-1}, \\
&\overset{(5)}{=} \left(1 - \frac{b^k}{c^k}\right)\frac{1+c}{\frac{c}{b}-1},
\end{aligned}$$

where (3) uses $\sum_{j=0}^{k-1} a^j = \frac{1-a^k}{1-a}$ and $\sum_{j=1}^{k} a^j = a\frac{1-a^k}{1-a}$, (4) groups the terms in $\left(\left(\frac{c}{b}\right)^k - 1\right)$ and (5) simplifies the powers of k. Assuming that $0 < b < c$, $1 - \frac{b^k}{c^k}$ is upper-bounded and converges to 1 and we have that

$$\mathbb{E}\left[h(x_{k+1})\right] \leq \left((1-\kappa^{-1}) + 2\frac{1+c}{\frac{c}{b}-1}\right) c^k h(x_0).$$

Hence, as long as there exists $0 < b < (1-\kappa^{-1}) \leq c < 1$, such that $(1-\kappa^{-1}) + 2\frac{1+c}{\frac{c}{b}-1} \leq c$, we can have linear convergence with rate c. This constraint can be made more explicit when rewritten as a quadratic function,

$$-c^2 + c(1-\kappa^{-1}+3b) + b(1+\kappa^{-1}) \leq 0.$$

As the constraint is concave in c, it is satisfied when

$$c \leq \frac{1}{2}((1-\kappa^{-1}+3b) - \sqrt{(1-\kappa^{-1}+3b)^2 + b(1+\kappa^{-1})})$$

or

$$c \geq \frac{1}{2}((1-\kappa^{-1}+3b) + \sqrt{(1-\kappa^{-1}+3b)^2 + b(1+\kappa^{-1})}).$$

A small development shows that the upper bound on c is not possible as it conflicts with $1-\kappa^{-1} \leq c$, while the lower bound implies $1-\kappa^{-1} \leq c$. Taking $b = 0$, i.e., without quantization introducing variance and the machine gradients being transmitted exactly, the lower bound reduces to $c \geq 1-\kappa^{-1}$. To ensure convergence, we must have that

$$1 > c \geq \frac{1}{2}(\sqrt{(1-\kappa^{-1}+3b)^2 + b(1+\kappa^{-1})} + (1-\kappa^{-1}+3b)).$$

Shifting the focus on a bound on b

This means that linear convergence is possible if

$$\frac{1}{2}(\sqrt{(1-\kappa^{-1}+3b)^2 + b(1+\kappa^{-1})} + (1-\kappa^{-1}+3b)) < 1,$$

with a rate of $c = \frac{1}{2}((1-\kappa^{-1}+3b) + \sqrt{(1-\kappa^{-1}+3b)^2 + b(1+\kappa^{-1})})$. To conclude the proof, the more concise representation can be obtained through the following steps;

$$
\begin{aligned}
& \frac{1}{2}(\sqrt{(1-\kappa^{-1}+3b)^2 + b(1+\kappa^{-1})} + (1-\kappa^{-1}+3b)) < 1, \\
\longleftrightarrow \quad & \sqrt{(1-\kappa^{-1}+3b)^2 + b(1+\kappa^{-1})} < 1 + \kappa^{-1} - 3b, \\
\longleftrightarrow \quad & (1-\kappa^{-1}+3b)^2 + b(1+\kappa^{-1}) < (1+\kappa^{-1}-3b)^2, \\
\longleftrightarrow \quad & 4(3b-\kappa^{-1}) + b(1+\kappa^{-1}) < 0, \\
\longleftrightarrow \quad & b(13+\kappa^{-1}) < 4\kappa^{-1}, \\
\longleftrightarrow \quad & b(13\kappa+1) < 4, \\
\longleftrightarrow \quad & b < \frac{4}{13\kappa+1}.
\end{aligned}
$$

□

Assuming a constant condition number, this results makes it possible to match the information theoretical lower bound of $\Omega(D(\log D + \log(1/\epsilon)))$ bits of communication to achieve ϵ-accurate results.

Comparing quantized and vanilla Gradient Descent It is not trivial to compare directly the improvement per bit transmitted between traditional Gradient Descent and Quantized Gradient Descent as, for simplicity, the theoretical bounds assume that the gradients are transmitted exactly in GD and that the norm of the gradient is transmitted exactly if QDGD is implemented using the quantization and compression scheme of Alistarh et al. [2016]. However, assuming that the error induced by the floating point precision with $F = 64$ bits of precision is negligible, which should hold in the early iterations of the algorithm, Consider a problem with a condition number of $\kappa = 200$ in $D = 100$ dimensions. An update of GD needs to transmit $F \cdots D = 3200$ bits and leads to a multiplicative decrease of at least $1 - \kappa^{-1} = 0.995$. An update of QDGD with parameters $b = \eta\frac{4}{13\kappa+1}, \eta = 0.1$[2] using the quantization and compression scheme of Alistarh et al. [2016] with $s = \sqrt{D/b}$ (see Lemma 8) needs to transmits less than 800 bits (see Lemma 9) and the previous theorem shows that the multiplicative decrease is at most 0.996. As we can do 4 updates of QDGD for at most the same communication cost of one update of GD, at equal communication cost, QDGD is more efficient than GD as the multiplicative decrease of 4 steps of QDGD is $0.996^4 < 0.985 < 0.995$. For clarity, those results are summarized in Table 1.

Comparison of multiplicative error reduction of quantized vs. traditional GD.

	Error reduction	Communication Cost
DGD	≤ 0.995	3200
QDGD-F	≤ 0.996	<800
QDGD-F (4 steps)	$\leq 0.996^4 \leq 0.985$	<3200

Assumption on the problem: $\kappa = 200, D = 100, F = 32$.

Table 1: **Multiplicative error reduction of quantized vs. traditional GD.** Results for Distributed Gradient Descent (DGD) and Quantized Distributed Gradient Descent with Feedback (QDGD-F), under the assumption that the problem has a condition number of $\kappa = 200$ in $D = 100$ dimensions, that the error induced by the floating point precision with $F = 32$ is negligible and that the quantization and compression scheme of Alistarh et al. [2016] is used.

4.3 Experiment

To confirm the theoretical results, we try DGD, QDGD and QDGD-F using 2 machines on the RCV1 dataset Lewis et al. [2004]. We use the LYRL2004 split, using the first 23149 samples to train a linear regression to identify the largest class present in the dataset, `CCAT`, encoded as +1, where other classes are encoded as 0. We compute the smoothness constant of the problem, β, and add a $\lambda = \beta/99$-weighted L_2

[2] $\eta = 0.1$ was picked arbitrarily for the sake of this example and might not be optimal.

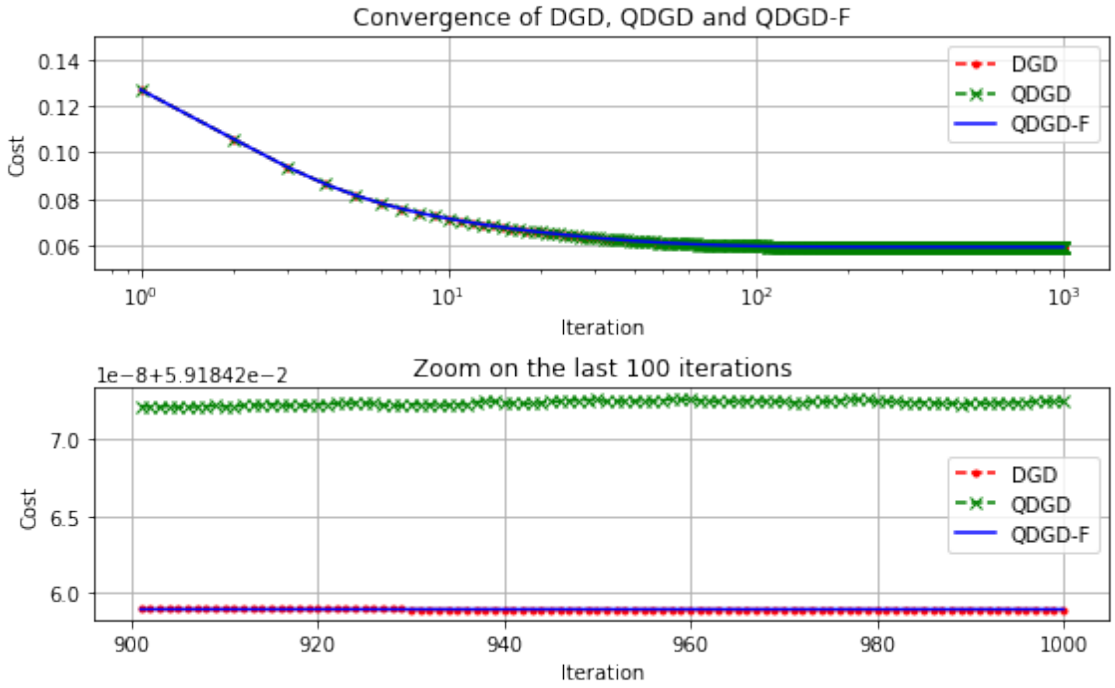

Figure 1: **Progress per iteration of DGD, QDGD and QDGD-F.** All three methods perform similarly, but QDGD suffers from the Noise Barrier problem and is not able to converge to the global optimum, as made clear by the zoom on the last 100 iterations.

regularization term in order to ensure that $\kappa < 200$ and use $b = \frac{1}{2}\frac{4}{13 \cdot 200+1}$ as the quantization precision for both QDGD and QDGD-F. This problem is too simple to be representative of a real-world machine learning pipeline, but it does illustrate the two main points discussed in this work:

- QDGD suffers from the Noise Barrier problem, and QDGD-F fixes that problem, as illustrated in Figure 1. As the problem is relatively simple, DGD, QDGD and QDGD-F all perform well, but QDGD can not converge to the optimum due to noise, as made clear by the zoom on the last 100 iterations.
- Quantized methods are more communication efficient than traditional gradient descent, as illustrated in Figure 2, where the quantized methods use an order of magnitude less bits than traditional gradient descent.

5 Conclusion

This work shows that quantization can be used to reduce the communication cost of gradients exchanges in distributed gradient descent systems and that the feedback mechanism can be used as a variance reduction technique for quantization approaches, making it possible to achieve linear convergence in this setting and asymptotically approach the information theoretic limit of communication optimal convex optimization.

An interesting question for future research raised by those results is the trade-off between the communication cost and the accuracy of the gradient update. As illustrated in Theorem 12, quantization can be applied to reduce the communication cost of the gradient update at the cost of some loss in convergence rate. As discussed in the earlier sections, stochastic gradient updates make it possible to decrease the computation cost of a gradient update at the cost of a reduction in the accuracy of the gradient updates. Given those communication-accuracy and computation-accuracy trade-offs, it should be possible to have an algorithm that can adapt to the relative cost of communication over computation in a distributed system while keeping a desired level of accuracy. As a first step towards this idea, we show in Appendix A that it is possible to adapt our quantization with feedback approach to SVRG

Many other questions are still open on this topic, such as the optimality of the quantization and compression scheme introduced in Alistarh et al. [2016]. While our results for the communication costs

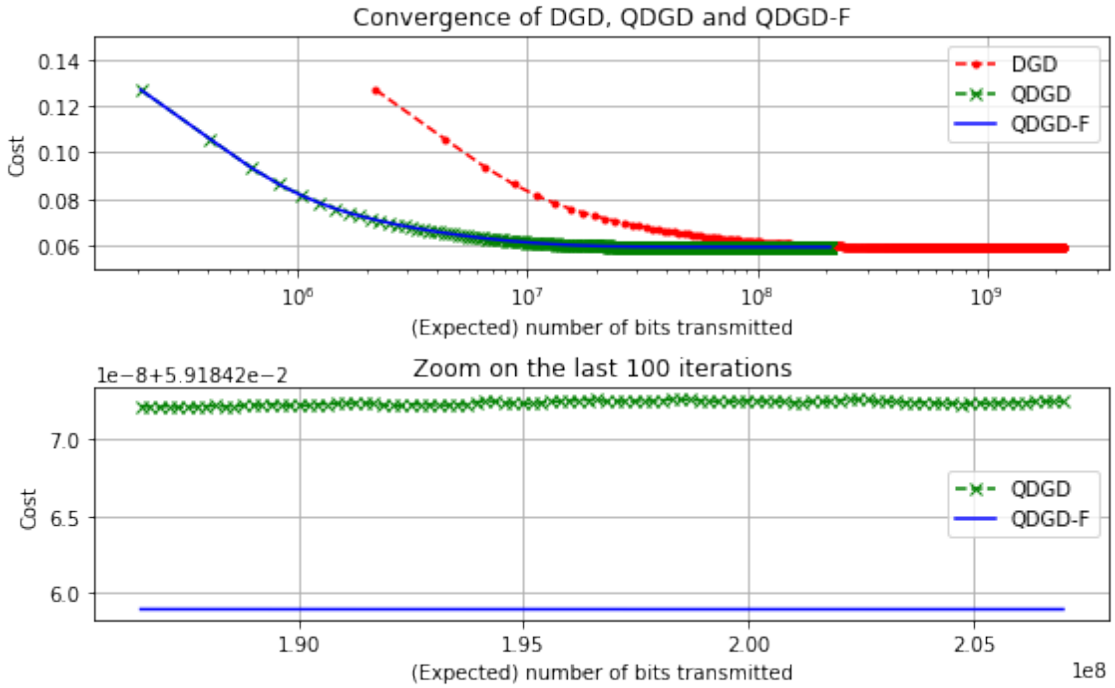

Figure 2: **Progress per (expected) number of transmitted bits.** The number of expected bits is computed using Lemmas 8 and 9. The obvious observation is that the quantized versions of gradient descent use an order of magnitude less communication to achieve similar results as the unquantized version.

rely on those schemes, our convergence results only assume a bounded variance blowup and can thus be applied to other quantization schemes with possibly other desirable properties such as sparsity.

Our convergence analysis assumes that the system starts with the knowledge of the correct gradients for each machines to make the analysis of the convergence rate easier. However, we observed that this might not be necessary in practice and whether this assumption is necessary in theory is an interesting question as the feedback mechanism might be able to cope with some initial error given enough precision.

The focus of this work has been on gradient descent methods, but those results could be adapted to work in coordinate-descent (CD) methods. In distributed CD settings, the data is typically distributed over multiple machines by features, such that each machine knows some part of every samples instead of everything about a subset of them. However, this makes it difficult to compute the gradient on one machine, as it depends on features unavailable on the current machine. In this setting, Mannari [2017] studied the effect of quantization on the transfer of the information required to compute the gradient on one machine, but the analysis must be tailored to a specific cost function, and it is not yet known how to achieve a linear convergence rate with quantization in this setting. It is possible that our feedback mechanism and its analysis can be applied to this setting. It should also be possible to adapt our full gradient methods to work on blocks of coordinates instead to get parts of the benefits of coordinate descent methods.

References

D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic. Qsgd: Communication-optimal stochastic gradient descent, with applications to training neural networks (version 3 on arxiv). 2016.

S. Bubeck. Convex optimization: Algorithms and complexity. 2015.

T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman. Project Adam: Building an Efficient and Scalable Depp Learning Training System. 2014.

J. Dean, G. S. Corrado, R. Monda, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng. Large Scale Distributed Deep Networks. 2012.

A. Defazio and F. Bach. SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. 2014.

R. Johnson and T. Zhang. Accelerating Stochastic Gradient Descent using Predictive Variance Reduction. 2013.

N. Le Roux, M. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. 2017.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A New Benchmark Collection for Text Categorization Research. 2004.

M. Mannari. Faster Coordinate Descent for Machine Learning through Limited Precision Operations. 2017.

A. Nedic and D. Bertsekas. Convergence Rate of Incremental Subgradient Algorithms. 2000.

F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu. 1-Bit Stochastic Gradient Descent and its Application to Data-Parallel Distributed Training of Speech DNNs. 2014.

S. Shalev-Shwarts and T. Zhang. Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization. 2013.

N. Strom. Scalable Distributed DNN Training Using Commodity GPU Cloud Computing. 2015.

J. N. Tsitsiklis and Z.-Q. Luo. Communication Complexity of Convex Optimization. 1987.

J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans. Distributed Asynchronous Deterministic and Stochastic Gradient Optimization Algorithms. 1986.

H. Zhang, J. Li, K. Kara, D. Alistarh, H. Liu, and C. Zhang. The ZipML Framework for Training Models with End-to-End Low Precision: The Cans, the Cannots, and a Little Bit of Deep Learning. 2016.

Appendix

A QSVRG and the Computation-Communication Tradeoff

Using Quantized MiniBatch GD as an example, the noise barrier depends on the variance of the gradient estimate will depend on the variance introduced by the quantization operation and the variance introduced by the selection of the samples to compute the gradient from. This means that it is possible to decrease the precision of the quantization while increasing the size of the mini-batches to counteract the increase in variance, and still conserve the same bound on the variance of the gradient estimate. This should makes it possible to trade computation accuracy for computational accuracy and tune the parameters of the algorithm to adapt to the architecture of the system it is running on, depending on the relative cost in time of computation vs. communication.

In order to find an algorithm with such properties that still achieves linear convergence, we show here that it is possible to combine the feedback mechanism introduced to deal with the variance of the quantization with the variance reduction mechanism for SGD, SVRG. While we do not provide a full analysis of the computation-communication tradeoff, it should be possible to extend our results by simply considering different quantization precision and minibatch sizes for the inner iterations of the QSVRG algorithm.

A.1 Quantized SVRG algorithm

We assume that, at the start of the algorithm, we are given an initial point y_0 and each machine knows the exact value of the gradient of each other machine, stored in a variable $G(0,m) = \nabla f_m(y_0)$. At the start of iteration k, all machine know y_k and have an estimate of the gradient of every machine at iteration $k-1$ stored in $G(k-1,m)$. Starting with $x_0^{(k)} = y_k$, we will do a SVRG update with a gradient estimate of the form $g_t^{(k)} = G_s(t,k) + G_e(t,k)$, where G_s is the stochastic gradient estimate and G_e is the epoch gradient estimate. G_s is taken from a sample uniformly at random,

$$G_s(t,k) = Q_1(\nabla f_{i_t}(x_t^{(k)}) - \nabla f_{i_t}(y_k)), \qquad i_t \sim U[1,N].$$

In order to approximate the epoch gradient for y_k without without having to transmit it in full precision, we will only send a b-quantization of the difference from the previous gradient estimate $G(k-1,m)$, with $0 < b < 1$;

$$G(k,m) = G(k-1,m) + Q_b(\nabla f_m(y_k) - G(k-1,m)).$$

$b < 1$ is needed to ensure that we indeed make progress, and $b > 0$ simply means that we are not transmitting the gradient with full precision. As $G(k,m)$ is only an approximation of the epoch gradient, using it without modification for multiple iterations would bias the gradient estimate. To avoid this problem, we send a small correction term at each iteration to randomize the epoch gradient estimate

$$G_e(t,k) = \frac{1}{M}\sum_{m=1}^{M} G_e(t,k,m), \qquad G_e(t,k,m) = G(k,m) + Q_1(\nabla f_m(y_k) - G(k,m)).$$

This yields the following gradient estimate at step t;

$$g_t^{(k)} = \underbrace{Q_1(\nabla f_{i_t}(x_t^{(k)}) - \nabla f_{i_t}(y_k))}_{\text{Stochastic Gradient}} + \underbrace{\frac{1}{M}\sum_{m=1}^{M} G(k,m) + Q_1(\nabla f_m(y_k) - G(k,m))}_{\text{Randomized Quantized Epoch Gradient Estimate}}.$$

Sub-optimality of this procedure

The information sent at each iteration is an approximation of the difference between our epoch gradient estimate and the true gradient $\nabla f_m(y_k)$. While we need it to make sure the epoch gradient estimates are independent across iterations, we could also use them to refine our estimate of the epoch gradient by replacing $G(k,m)$ by $G(k,m) + Q_1(\nabla f_m(y_k) - G(k,m))$ after each iteration.

While this could be used to avoid having to transmit the b-quantization of the epoch gradient difference at the start of the epoch, or at least reduce the required precision, we chose not to analyze this scheme as its analysis is more cumbersome and notation heavy. It might be however an interesting optimization

to do in practice; assuming our scheme achieves linear convergence, using the additional information transmitted at each epoch should improve the constant in the convergence rate.

A.2 Convergence analysis

We will apply the following steps:

- Bound the variance of the gradient estimate, $\mathbb{E}\left[\left\|g_t^{(k)}\right\|^2\right]$ (Theorem 16)
  - Separate the stochastic gradient variance from the epoch gradient estimate variance (Lemma 13)
  - Apply the SVRG analysis (Lemma 4) to the stochastic gradient variance.
  - Relate the variance of the epoch gradient estimate at iteration t to the variance of the epoch gradient estimate at the start of the epoch (Lemma 14).
  - Relate the variance of the epoch gradient estimate at the start of the epoch to the variance of the epoch gradient estimate of previous epochs (Lemma 15).

A.2.1 Bound for the variance of the gradient estimate

Lemma 13 (Separation of stochastic and epoch variance)**.**

$$\mathbb{E}_{g_t^{(k)}}\left[\|g_t\|^2\right] \leq \mathbb{E}_{G_s(t,k)}\left[\|G_s(t,k)+\nabla f(y_k)\|^2\right] + \mathbb{E}_{G_e(t,k)}\left[\|\nabla f(y_k)+G_e(t,k)\|^2\right].$$

Proof: The following operations show how to separate the variance.

$$\begin{aligned}
\mathbb{E}_{g_t^{(k)}}\left[\|g_t\|^2\right] &= \mathbb{E}_{G_s(t,k),G_e(t,k)}\left[\|G_s(t,k)+G_e(t,k)\|^2\right], \\
&\overset{(1)}{=} \mathbb{E}_{G_s(t,k),G_e(t,k)}\left[\|G_s(t,k)+\nabla f(y_k)-\nabla f(y_k)+G_e(t,k)\|^2\right], \\
&\overset{(2)}{=} \mathbb{E}_{G_s(t,k)}\left[\|G_s(t,k)+\nabla f(y_k)\|^2\right] + \mathbb{E}_{G_e(t,k)}\left[\|-\nabla f(y_k)+G_e(t,k)\|^2\right], \\
&\quad - 2\mathbb{E}_{G_s(t,k),G_e(t,k)}\left[\langle G_s(t,k)+\nabla f(y_k), -\nabla f(y_k)+G_e(t,k)\rangle\right], \\
&\overset{(3)}{\leq} \mathbb{E}_{G_s(t,k)}\left[\|G_s(t,k)+\nabla f(y_k)\|^2\right] + \mathbb{E}_{G_e(t,k)}\left[\|\nabla f(y_k)+G_e(t,k)\|^2\right].
\end{aligned}$$

In (1), we add and remove the true gradient at y_k, (2) expands the norm and (3) uses the fact that the expectation of the $G_e(t,k)$ is $\nabla f(y_k)$ to eliminate the dot product. □

Lemma 14 (Relation of the epoch gradient estimate at iteration t to the gradient estimate at the start of the epoch)**.**

$$\mathbb{E}_{G_e(t,k)}\left[\|\nabla f(y_k)-G_e(t,k)\|^2\right] \leq \mathbb{E}_{m,G(k,m)}\left[\|\nabla f_m(y_k)-G(k,m)\|^2\right].$$

Proof: This relation is easy using Lemma 8 for Q_1;

$$\begin{aligned}
\mathbb{E}_{G_e(t,k)}\left[\|\nabla f(y_k) - G_e(t,k)\|^2\right] &\overset{(1)}{=} \mathbb{E}_{G_e(t,k,1),\dots,G_e(t,k,M)}\left[\left\|\frac{1}{M}\sum_{m=1}^{M}\nabla f_m(y_k) - G_e(t,k,m)\right\|^2\right], \\
&\overset{(2)}{\leq} \mathbb{E}_{m,G_e(t,k,m)}\left[\|\nabla f_m(y_k) - G_e(t,k,m)\|\,2\right], \\
&\overset{(3)}{=} \mathbb{E}_{m,G(t,k,m)}\left[\|\nabla f_m(y_k) - G(k,m) - Q_1(\nabla f_m(y_k) - G(k,m))\|^2\right], \\
&\overset{(4)}{\leq} \mathbb{E}_{m,G(k,m)}\left[\|\nabla f_m(y_k) - G(k,m)\|^2\right].
\end{aligned}$$

(1) expands the definition of $G(t,k)$, (2) uses Trick 22 to transform the sum into an expectation, (3) expands the definition of $G(t,k,m)$ and (4) applies Lemma 8. □

Lemma 15 (Relation of the epoch gradient estimate at current epoch to previous epochs)**.**

$$\mathbb{E}\left[\|\nabla f_m(y_k) - G(k,m)\|^2\right] \leq 4\beta\left[\sum_{j=0}^{k-1} b^{k-j} f(y_j) - f^* + b\sum_{j=1}^{k} b^{k-j} f(y_j) - f^*\right].$$

Proof:

$$\begin{aligned}
\mathbb{E}_{m,G(k,m)}\left[\|\nabla f_m(y_k) - G(k,m)\|^2\right] &\overset{(1)}{=} \mathbb{E}_{m,Q_b}\left[\|\nabla f_m(y_k) - g(k-1,m) - Q_b(\nabla f_m(y_k) - G(k-1,m)\|^2\right], \\
&\overset{(2)}{\leq} b\mathbb{E}_m\left[\|\nabla f_m(y_k) - G(k-1,m)\|\right], \\
&\overset{(3)}{=} b\mathbb{E}_m\left[\|\nabla f_m(y_k) - \nabla f_m(y_{k-1}) + \nabla f_m(y_{k-1}) - G(k-1,m)\|\right], \\
&\overset{(4)}{=} b\mathbb{E}_m\left[\|\nabla f_m(y_k) - \nabla f_m(y_{k-1}\|^2\right] \\
&\quad + b\mathbb{E}_m\left[\|\nabla f_m(y_{k-1}) - G(k-1,m)\|^2\right],
\end{aligned}$$

(1) expands the definition of $G(k,m)$, (2) applies Lemma 8, (3) adds and remove the true gradient at the previous epoch, (4) uses Trick 19.

We can relate the difference in gradients to the value of the function at those points;

$$\begin{aligned}
\mathbb{E}_m\left[\|\nabla f_m(y_k) - \nabla f_m(y_{k-1})\|^2\right] &\overset{(1)}{=} \mathbb{E}_m\left[\|\nabla f_m(y_k) - \nabla f_m(x^*) + \nabla f_m(x^*) - \nabla f_m(y_{k-1}\|^2\right], \\
&\overset{(2)}{=} 2\mathbb{E}_m\left[\|\nabla f_m(y_k) - \nabla f_m(x^*)\|^2\right] + 2\mathbb{E}_m\left[\|\nabla f_m(y_{k-1}) - \nabla f_m(x^*)\|^2\right], \\
&\overset{(3)}{=} 4\beta\left[f(y_k) - f^* + f(y_{k-1}) - f^*\right]
\end{aligned}$$

(1) adds and remove the gradient of f_m at the optimal point of f, (2) separates the norm using Trick 22 and (3) uses Trick 21.

Plugging this back in the bound we had, we get

$$\begin{aligned}
\mathbb{E}_{m,G(k,m)}\left[\|\nabla f_m(y_k) - G(k,m)\|^2\right] &\leq b4\beta\left[f(y_k) - f^* + f(y_{k-1}) - f^*\right], \\
&\quad + b\mathbb{E}_{m,G(k-1,m)}\left[\|\nabla f_m(y_{k-1}) - G(k-1,m)\|^2\right].
\end{aligned}$$

And applying the same steps recursively to reach $G(0,m)$ gives us

$$\mathbb{E}\left[\|\nabla f_m(y_k) - G(k,m)\|^2\right] \leq 4\beta\left[\sum_{j=0}^{k-1} b^{k-j}(f(y_j) - f^*) + b\sum_{j=1}^{k} b^{k-j}(f(y_j) - f^*)\right].$$

□

A rough upper bound on the sum on the right, ignoring constants, is $b^k(\sum_{j=0}^{k} b^{-j} f(y_j) - f^*$, hence if the sequence $f(y_k) - f^*$ decreases at a linear rate of b, having $f(y_k) - f^* \leq b^k(f(y_k) - f^*)$, the variance decreases at a rate of kb^k, which leads
Direct application the previous lemmas gives us

Theorem 16 (Bound of the variance of g_t for the current procedure)**.**

$$\mathbb{E}\left[\|g_t\|^2\right] \leq 12\beta(f(x_t) - f^* + f(y_k) - f^*) + 4\beta\left[\sum_{j=0}^{k-1} b^{k-j}(f(y_j) - f^*) + b\sum_{j=1}^{k} b^{k-j}(f(y_j) - f^*)\right].$$

The main implication of this theorem is that if the quantization precision $b < 1$ and the error of the sequence y_k decreases quickly enough, the variance of the gradient estimate converges to 0.

A.2.2 Recursive error formulation

We are now ready to do some heavy lifting on this formula and prove the following theorem.

Theorem 17.
Assuming that $\gamma = \frac{\eta}{\beta}$, $T = \frac{2}{\gamma\alpha}$ with $\eta \leq 1/6$, we have the following recursive relation

$$h(y_{k+1}) \leq \frac{1 + 12\eta}{2 - 12\eta} h(y_k) + \frac{4\eta}{2 - 12\eta}\left[\sum_{j=0}^{k-1} b^{k-j} h(y_j) + b\sum_{j=1}^{k} b^{k-j} h(y_j)\right].$$

Proof:

Step 1: Link $\|x_{t+1} - x^*\|^2$ to the variance of the gradient estimate
Starting from the definition of x_{T+1}, we can use the unbiasedness of the gradient estimate abd plug in our bound for the variance to get the following relation. For the sake of space we use $h(x)$ as a shortcut for $f(x) - f^*$.

$$\begin{aligned}
\mathbb{E}_{x_{T+1}^{(k)}|x_T^{(k)}}\left[\left\|x_{T+1}^{(k)} - x^*\right\|^2\right] &\overset{(1)}{=} \mathbb{E}_{g_T^{(k)}}\left[\left\|x_T^{(k)} - \gamma g_T^{(k)} - x^*\right\|^2\right], \\
&\overset{(2)}{=} \mathbb{E}_{g_T^{(k)}}\left[\left\|x_T^{(k)} - x^*\right\|^2 - 2\gamma\langle g_T^{(k)}, x_T^{(k)} - x^*\rangle + \gamma^2\left\|g_T^{(k)}\right\|^2\right], \\
&\overset{(3)}{\leq} \left\|x_T^{(k)} - x^*\right\|^2 - 2\gamma\langle \nabla f(x_T^{(k)}), x_T^{(k)} - x^*\rangle + 12\beta\gamma^2(h(x_t) + h(y_k)), \\
&\quad + 4\beta\gamma^2\left[\sum_{j=0}^{k-1} b^{k-j} h(y_j) + b\sum_{j=1}^{k} b^{k-j} h(y_j)\right], \\
&\overset{(4)}{\leq} \left\|x_T^{(k)} - x^*\right\|^2 - 2\gamma h(x_T^{(k)}) + 12\beta\gamma^2(h(x_t) + h(y_k)), \\
&\quad + 4\beta\gamma^2\left[\sum_{j=0}^{k-1} b^{k-j} h(y_j) + b\sum_{j=1}^{k} b^{k-j} h(y_j)\right], \\
&\overset{(5)}{\leq} \left\|x_T^{(k)} - x^*\right\|^2 + (12\beta\gamma^2 - 2\gamma)h(x_T^{(k)}) + 12\beta\gamma^2(h(y_k)), \\
&\quad + 4\beta\gamma^2\left[\sum_{j=0}^{k-1} b^{k-j} h(y_j) + b\sum_{j=1}^{k} b^{k-j} h(y_j)\right].
\end{aligned}$$

(1) expands the definition of $x_{T+1}^{(k)}$, (2) expands the norm using $\|a + b\|^2 = \|a\| + \|b\| + 2\langle a, b\rangle$, (3) uses the unbiasedness of the gradient estimate and the bound on the variance derived in Thm 16, (4)

uses $\langle \nabla f(x_T^{(k)}), x_T^{(k)} - x^* \rangle \geq h(x_T^{(k)})$ and (5) groups the terms in $h(x_T^{(k)})$.

We can repeat the process to link $x_{T+1}^{(k)}$ to $x_1^{(k)}$;

$$\begin{aligned}
\mathbb{E}_{x_{T+1}^{(k)}|x_1^{(k)}}\left[\left\|x_{T+1}^{(k)} - x^*\right\|^2\right] &\overset{(1)}{\leq} \left\|x_1^{(k)} - x^*\right\|^2 + (12\beta\gamma^2 - 2\gamma)\sum_{t=1}^{T} h(x_T^{(k)}) + 12\beta T\gamma^2(h(y_k)), \\
&\quad + 4\beta T\gamma^2\left[\sum_{j=0}^{k-1} b^{k-j}h(y_j) + b\sum_{j=1}^{k} b^{k-j}h(y_j)\right], \\
&\overset{(2)}{\leq} \frac{2}{\alpha}h(x_1^{(k)}) + (12\beta\gamma^2 - 2\gamma)\sum_{t=1}^{T} h(x_T^{(k)}) + 12\beta T\gamma^2(h(y_k)), \\
&\quad + 4\beta T\gamma^2\left[\sum_{j=0}^{k-1} b^{k-j}h(y_j) + b\sum_{j=1}^{k} b^{k-j}h(y_j)\right], \\
0 &\overset{(3)}{\leq} \frac{2}{T\alpha}h(x_1^{(k)}) + (12\beta\gamma^2 - 2\gamma)\frac{1}{T}\sum_{t=1}^{T} h(x_T^{(k)}) + 12\beta\gamma^2(h(y_k)), \\
&\quad + 4\beta\gamma^2\left[\sum_{j=0}^{k-1} b^{k-j}h(y_j) + b\sum_{j=1}^{k} b^{k-j}h(y_j)\right],
\end{aligned}$$

(1) applies the previous steps repeatedly and (2) uses the fact that $\|x - x^*\|^2 \leq \frac{2}{\alpha}(f(x) - f^*)$. (3) uses the fact that the norm on the left hand side is necessarily positive to bound the right hand side as positive and divides by T.

To simplify the notation, we will assume that the parameters T and γ are linked; bigger iterations will need more averaged samples to get an accurate answer. Building from our knowledge of SVRG, choosing $T = \frac{2}{\gamma\alpha}$ and $\gamma = \frac{\eta}{\beta}$, where η controls the length of the step size and iterations per epoch, works nicely. Defining $x_1^{(k)} := y_k$ and moving the term dependent on the average error during epoch $k+1$, we get

$$(2 - 12\eta)\frac{1}{T}\sum_{t=1}^{T} h(x_T^{(k)}) \leq (1 + 12\eta)h(y_k) + 4\eta\left[\sum_{j=0}^{k-1} b^{k-j}h(y_j) + b\sum_{j=1}^{k} b^{k-j}h(y_j)\right].$$

Assuming that $\eta < 1/6$, the left hand side is still positive and we can use the convexity of h to get $\sum_{t=1}^{T} h(x_T^{(k)}) \geq h(\sum_{t=1}^{T} x_T^{(k)})$. Defining $y_{k+1} = \sum_{t=1}^{T} x_T^{(k)}$ as the average of the iterates during epoch $k+1$, we get

$$h(y_{k+1}) \leq \frac{1 + 12\eta}{2 - 12\eta}h(y_k) + \frac{4\eta}{2 - 12\eta}\left[\sum_{j=0}^{k-1} b^{k-j}h(y_j) + b\sum_{j=1}^{k} b^{k-j}h(y_j)\right].$$

□

A.2.3 Linear convergence proof for some parameters

The recursive expression obtained as an upper bound for the error at step y_{k+1}, depending on all previous error, might seem messy at first. However, we can still show linear convergence under some set of parameters. While the proof of the following theorem might not seem very illuminating on the true behavior of QSVRG with quantized gradients, it is a useful first step in understanding the relationship between parameters.

Theorem 18.
With $b = 1/2$ and $\eta = 1/48$, we have linear convergence of QSVRG with quantized gradients;

$$\mathbb{E}\left[f(y_k) - f^*\right] \leq 0.9^k(f(y_0) - f^*).$$

Proof: The expression we found in Theorem 17 for the error at step y_{k+1} is

$$h(y_{k+1}) \leq \frac{1}{2-12\eta}\left((1+12\eta)h(y_k) + 4\eta\left(\sum_{j=0}^{k-1} b^{k-j}h(y_j) + b\sum_{j=1}^{k} b^{k-j}h(y_j)\right)\right).$$

Replacing the constants $b = 1/2$ and $\eta = 1/48$,

$$h(y_{k+1}) \leq \frac{1}{7}\left(5h(y_k) + \frac{1}{3}\left(\sum_{j=0}^{k-1}\frac{1}{2^{k-j}}h(y_j) + \frac{1}{2}\sum_{j=1}^{k}\frac{1}{2v}h(y_j)\right)\right).$$

For the first step, it is easy to see that we have $h(y_1) \leq \frac{5}{7}h(y_0)$; as we assume we start with the exact epoch gradient in memory, there is no variance induced by the quantization of the epoch gradients yet, and we have the same convergence rate as QSVRG without quantization of epoch gradients.

For the subsequent steps however, the variance of the epoch gradients starts to play a role. For the second step, a bit of torturing gives us

$$\begin{aligned}
h(y_2) &\leq \frac{5}{7}h(y_1) + \frac{1}{21}\left(\frac{1}{2}h(y_0) + \frac{1}{2}h(y_1)\right), \\
&= \frac{5}{7}\left(\frac{5}{7} + \frac{1}{15}\left(\frac{1}{2} + \frac{1}{2}\frac{5}{7}\right)\right)h(y_0), \\
&= \frac{5}{7}\left(\frac{5}{7} + \frac{1}{15}\frac{6}{7}\right)h(y_0), \\
&= \frac{5}{7}\left(\frac{5}{7} + \frac{\frac{2}{5}}{7}\right)h(y_0), \\
&\leq \frac{5}{7}\frac{6}{7}h(y_0).
\end{aligned}$$

So we can see that the performance is degrading due to the added error. However, there is a limit to this degradation, which depends on the parameters b, η and the best convergence rate achievable. For our specific choice of parameters, this limit happens to be (slightly below) 6/7. To show this, we will proceed by induction.

Assume that for all $0 \leq j \leq k$, we have that $h(y_j) \leq (6/7)^j h(y_0)$. then, $h(y_{k+1}) \leq (6/7)^{k+1}h(y_0)$. As we have already shown the base cases to illustrate the degradation of the convergence rate, we only need to show that it holds for $k+1 > 2$ given the previous steps;

$$\begin{aligned}
h(y_{k+1}) &\overset{(1)}{\leq} \frac{5}{7}h(y_k) + \frac{1}{21}\frac{1}{2^k}\left(\sum_{j=0}^{k-1} 2^j h(y_j) + \frac{1}{2}\sum_{j=1}^{k} 2h(y_j)\right), \\
&\overset{(2)}{=} \frac{5}{7}\left(\frac{6}{7}\right)^k h(y_0) + \frac{1}{21}\frac{1}{2^k}\left(\sum_{j=0}^{k-1}\left(2\frac{6}{7}\right)^j + \frac{1}{2}\sum_{j=1}^{k}\left(2\frac{6}{7}\right)^j\right)h(y_0),
\end{aligned}$$

Where (1) just develops from Theorem 17 and (2) applies the assumption that $h(y_j) \leq \left(\frac{6}{7}\right)^j h(y_0)$. In order for the recurrence to hold for y_{k+1}, we need to have that

$$\frac{1}{21}\frac{1}{2^k}\left(\sum_{j=0}^{k-1}\left(2\frac{6}{7}\right)^j + \frac{1}{2}\sum_{j=1}^{k}\left(2\frac{6}{7}\right)^j\right) \leq \frac{1}{7}\left(\frac{6}{7}\right)^k.$$

To show this, take the following steps;

$$\begin{aligned}
\frac{1}{21}\frac{1}{2^k}\left(\sum_{j=0}^{k-1}\left(2\frac{6}{7}\right)^j+\frac{1}{2}\sum_{j=1}^{k}\left(2\frac{6}{7}\right)^j\right) &\overset{(1)}{=} \frac{1}{21}\frac{1}{2^k}\left(\frac{\left(2\frac{6}{7}\right)^k-1}{2\frac{6}{7}-1}+\frac{6}{7}\frac{\left(2\frac{6}{7}\right)^k-1}{2\frac{6}{7}-1}\right),\\
&\overset{(2)}{=} \frac{1}{21}\frac{1}{2^k}\left(\frac{\frac{6}{7}+1}{2\frac{6}{7}-1}\left(\left(2\frac{6}{7}\right)^k-1\right)\right),\\
&\overset{(3)}{=} \frac{1}{21}\frac{1}{2^k\left(\frac{6}{7}\right)^k}\left(\frac{\frac{6}{7}+1}{2\frac{6}{7}-1}\left(\left(2\frac{6}{7}\right)^k-1\right)\right)\left(\frac{6}{7}\right)^k,\\
&\overset{(4)}{=} \frac{1}{21}\left(\frac{\frac{6}{7}+1}{2\frac{6}{7}-1}\left(1-\frac{1}{2^k\left(\frac{6}{7}\right)^k}\right)\right)\left(\frac{6}{7}\right)^k,
\end{aligned}$$

Where (1) uses $\sum_{j=0}^{k-1}x^j=\frac{1-x^k}{1-x}$ to simplify, (2) groups terms, (3) puts $\left(\frac{6}{7}\right)^k$ in evidence and (4) simplifies the powers of k. It is now easy to see that the convergence holds if the fraction before the $\left(\frac{6}{7}\right)^k$ are lower than $\frac{1}{7}$. As k grows, this term will tend to, and is bounded by,

$$\frac{1}{21}\frac{\frac{6}{7}+1}{2\frac{6}{7}-1}=\frac{13}{105}\leq\frac{1}{7}.$$

Concluding the proof. □

While the previous proof does not shed much light on the behavior of QSVRG with quantized epoch gradients, it is sufficient to show that under the specified parameters, we achieve linear convergence.

B Tricks

This section shows some of the well known inequalities used in multiple proofs in this report.

Trick 19.
Let $\bar{x}=\mathbb{E}[x]$ and c be a constant.

$$\mathbb{E}\left[\|x-\bar{x}+c\|^2\right]=\mathbb{E}\left[\|x-\bar{x}\|^2\right]+\|c\|^2.$$

Trick 20.
Let f be a L-smooth, μ-strongly convex function and f^* be the value of f at the global minimum. We have that

$$2\mu(f(x)-f^*)\leq\|\nabla f(x)\|^2\leq 2L(f(x)-f^*).$$

Trick 21 ((Bubeck [2015], Lemma 6.4))**.**

$$\mathbb{E}_i\left[\|\nabla f_i(x)-\nabla f_i(x^*)\|^2\right]\leq 2L(f(x)-f^*)$$

Trick 22.

$$\left\|\sum_{n=1}^{N}x_n\right\|^2\leq N\sum_{n=1}^{N}\|x_n\|^2.$$

$$\left\|\frac{1}{N}\sum_{n=1}^{N}x_n\right\|^2\leq\frac{1}{N}\sum_{n=1}^{N}\|x_n\|^2\leq\mathbb{E}_{n\sim U[1,N]}\left[\|x_n\|^2\right].$$