New Bounds For Distributed Mean Estimation and Variance Reduction

Abstract

We consider the problem of *distributed mean estimation (DME)*, in which n machines are each given a local d-dimensional vector $\boldsymbol{x}_v \in \mathbb{R}^d$, and must cooperate to estimate the mean of their inputs $\boldsymbol{\mu} = \sum_{v=1}^{n} \boldsymbol{x}_v$, while minimizing total communication cost. DME is a fundamental construct in distributed machine learning, and there has been considerable work on variants of this problem, especially in the context of *distributed variance reduction* for stochastic gradients in parallel SGD. Previous work typically assumes an upper bound on the norm of the input vectors, and achieves an error bound in terms of this norm. However, in many real applications, the input vectors are concentrated around the correct output $\boldsymbol{\mu}$, but $\boldsymbol{\mu}$ itself has large norm. In such cases, previous output error bounds perform poorly.

In this paper, we show that output error bounds need not depend on input norm. We provide a method of quantization which allows distributed mean estimation to be performed with solution quality dependent only on the *distance between inputs*, not on input norm, and show an analogous result for distributed variance reduction. The technique is based on a new connection with lattice theory. We also provide lower bounds showing that the *communication to error trade-off* of our algorithms is asymptotically optimal. As the lattices achieving optimal bounds under ℓ_2-norm can be computationally impractical, we also present an extension which leverages easy-to-use cubic lattices, and is loose only up to a logarithmic factor in d. We show experimentally that our method yields practical improvements for common applications, relative to prior approaches.

1 Introduction

Several problems in distributed machine learning and optimization can be reduced to variants *distributed mean estimation* problem, in which n machines must cooperate to jointly estimate the mean of their d-dimensional inputs $\boldsymbol{\mu} = \sum_{v=1}^{n} \boldsymbol{x}_v$ as closely as possible, while minimizing communication. In particular, this construct is often used for *distributed variance reduction*: here, each machine receives as input an independent probabilistic estimate of a d-dimensional vector $\boldsymbol{\nabla}$, and the aim is for all machines to output a *common estimate* of $\boldsymbol{\nabla}$ with *lower variance* than the individual inputs, minimizing communication. Without any communication restrictions, the ideal output would be the mean of all machines' inputs.

While variants of these fundamental problems have been considered since seminal work by Tsitsiklis & Luo (1987), the task has seen renewed attention recently in the context of distributed machine learning. In particular, variance reduction is a key component in data-parallel distributed stochastic gradient descent (SGD), the standard way to parallelize the training of deep neural networks, e.g. Bottou (2010); Abadi et al. (2016), where it is used to estimate the average of gradient updates obtained in parallel at the nodes. Thus, several prior works proposed efficient compression schemes to solve variance reduction or mean estimation, see e.g. Suresh et al. (2017); Alistarh et al. (2017); Ramezani-Kebrya et al. (2019); Gandikota et al. (2019), and Ben-Nun & Hoefler (2019) for a general survey of practical distribution schemes. These schemes seek to *quantize* nodes' inputs coordinate-wise to one of a limited collection of values, in order to then efficiently encode and transmit these quantized values. A trade-off then arises between the *number of bits sent*, and the *added variance* due of quantization.

Since the measure of *output* quality is variance, it appears most natural to evaluate this with respect to *input* variance, in order to show that *variance reduction* is indeed achieved. Surprisingly, however, we are aware of no previous works which do so; all existing methods give bounds on output variance in terms of the *squared input norm*. This is clearly suboptimal when the squared norm is higher than the variance, i.e., when inputs are not centered

around the origin. In some practical scenarios this causes output variance to be *higher* than input variance, as we demonstrate in Section 4.

Contributions. In this paper, we provide the first bounds for distributed mean estimation and variance reduction which are still tight when inputs are not centered around the origin. Our results are based on new lattice-based quantization techniques, which may be of independent interest, and come with matching lower bounds, and practical extensions. More precisely, our contributions are as follows:

- For *distributed mean estimation*, we show that, to achieve a reduction of a factor q in the input 'variance' (which we define to be the maximum squared distance between inputs), it is necessary and sufficient for machines to communicate $\Theta(d \log q)$ bits.
- For *variance reduction*, we show tight $\Theta(d \log n)$ bounds on the worst-case communication bits required to achieve optimal $\Theta(n)$-factor variance reduction by n nodes over d-dimensional input, and indeed to achieve any variance reduction at all. We then show how incorporating *error detection* into our quantization scheme, we can also obtain tight bounds on the bits required *in expectation*.
- We show how to efficiently instantiate our lattice-based quantization framework in practice, with guarantees. In particular, we devise a variant of the scheme which ensures close-to-optimal communication-variance bounds even for the standard *cubic lattice*, and use it to obtain improvements relative to the best known previous methods for distributed mean estimation, both on synthetic and real-world tasks.

1.1 Problem Definitions and Discussion

MeanEstimation is defined as follows: we have n machines v, and each receives as input a vector $\boldsymbol{x}_v \in \mathbb{R}^d$. We also assume that all machines receive a common value y, with the guarantee that for any machines u, v, $\|\boldsymbol{x}_u - \boldsymbol{x}_v\| \leq y$. Our goal is for all machines to output the same value $\boldsymbol{EST} \in \mathbb{R}^d$, which is an unbiased estimator of the mean $\boldsymbol{\mu} = \frac{1}{n}\sum_{v \in M} \boldsymbol{x}_v$, i.e. $\mathbf{E}[\boldsymbol{EST}] = \boldsymbol{\mu}$, with variance as low as possible. Notice that the input specification is entirely deterministic; any randomness in the output arises only from the algorithm used.

In the variant of VarianceReduction, we again have a set of n machines, and now an unknown *true* vector $\boldsymbol{\nabla}$. Each machine v receives as input an independent unbiased estimator $\boldsymbol{x}_v$ of $\boldsymbol{\nabla}$ (i.e., $\mathbf{E}[\boldsymbol{x}_v] = \boldsymbol{\nabla}$) with variance $\mathbf{E}\left[\|\boldsymbol{x}_v - \boldsymbol{\nabla}\|^2\right] \leq \sigma^2$. Machines are assumed to have knowledge of σ. Our goal is for all machines to output the same value $\boldsymbol{EST} \in \mathbb{R}^d$, which is an unbiased estimator of $\boldsymbol{\nabla}$, i.e., $\mathbf{E}[\boldsymbol{EST}] = \boldsymbol{\nabla}$, with low variance. Since the input is random, output randomness now stems from this input randomness as well as any randomness in the algorithm.

VarianceReduction is common for instance in the context of gradient-based optimization of machine learning models, where we assume that each machine v processes local samples in order to obtain a *stochastic gradient* $\tilde{g}_v$, which is an unbiased estimator of the true gradient $\boldsymbol{\nabla}$, with variance bound σ^2. If we directly averaged the local stochastic gradients $\tilde{g}_v$, we could obtain an unbiased estimator of the true gradient G with variance bound σ^2/n, which can lead to faster convergence.

Input Variance Assumption The parameter y replaces the usual MeanEstimation assumption of a known bound $\mathbb{M}$ on the *norms of input vectors*. Note that, in the worst case, we can always set $y = 2\mathbb{M}$ and obtain the same asymptotic upper bounds as in e.g. Suresh et al. (2017); our results are therefore *at least* as good as previous approaches in all cases, but, as we will show, provide significant improvement when inputs are not centered around the origin.

The reason for this change is to allow stronger bounds in scenarios where we expect inputs to be closer to each other thanto the origin. In particular, it allows our MeanEstimation problem to more effectively generalize VarianceReduction. Parameter y is a deterministic analogue of the parameter σ for VarianceReduction; both y and σ provide a bound on the distance of inputs from their *mean*, rather than from the origin. Accordingly, *input variance* σ^2 for a VarianceReduction instance corresponds (up to constant factors) to y^2 for a MeanEstimation instance. For consistency of terminology, we therefore refer to y^2 as the *input variance* of the instance (despite such inputs being deterministic).

It is common in machine learning applications of VarianceReduction to assume that an estimate of the variance σ^2 is known (Alistarh et al., 2017; Gandikota et al., 2019). To study both problems in a common framework, we make the analogous assumption about MeanEstimation, and assume knowledge of the *input variance* y^2. Even if the relevant bounds y or σ are not known *a priori*, they can usually be estimated in practice. We discuss how we obtain estimates of input variance for our applications in Appendix G.

Relationship Between Problems If one allows unrestricted communication, the best solution is to average the inputs. This is an exact solution to MeanEstimation with variance 0, and is also the best possible solution to VarianceReduction of variance at most $\frac{\sigma^2}{n}$. However, doing so would require the exchange of infinite precision real numbers. So, it is common to instead communicate *quantized* values of bounded bit-length (Alistarh et al., 2017), which will engender additional variance caused by random choices within the quantization method. The resulting estimates will therefore have variance $\boldsymbol{Var}_{quant}$ for MeanEstimation, and $\frac{\sigma^2}{n} + \boldsymbol{Var}_{quant}$ for VarianceReduction. We will show a trade-off between bits of communication and output variance for both problems; in the case of VarianceReduction, though, there is an 'upper limit' to this trade-off, since we cannot go below $\Omega(\frac{\sigma^2}{n})$ total output variance.

The other major difference between the two problems is that in MeanEstimation, distances between inputs are bounded by y with certainty, whereas in VarianceReduction they are instead bounded by $O(\sigma)$ only in *expectation*. This causes extra complications for quantization, and, as we will see, introduces a gap between average and worst-case communication cost.

Vector Norms When dealing with vectors in $\mathbb{R}^d$, we will use names in bold, e.g. $\boldsymbol{x}$, $\boldsymbol{y}$. We will state most of our results in such a way that they will apply to any of the three most commonly-used norms on $\mathbb{R}^d$ in applications: ℓ_1 norm $\|\boldsymbol{x}\|_1 := \sum_{i=1}^d x_i$, ℓ_2 norm $\|\boldsymbol{x}\|_2 := \sqrt{\sum_{i=1}^d x_i^2}$, and ℓ_∞ norm $\|\boldsymbol{x}\|_\infty := \max_{i=1}^d x_i$. Throughout the paper we will therefore use the general notation $\|\cdot\|$, which should be considered to be fixed as one of these norms, other than for statements specific to particular norms. Definitions which depend on norms, such as variance $\mathbf{Var}[\boldsymbol{x}] := \mathbf{E}\left[\|\boldsymbol{x} - \mathbf{E}[\boldsymbol{x}]\|^2\right]$, are therefore assumed to also be under the appropriate norm.

1.2 Related Work

Several recent works consider efficient compression schemes for stochastic gradients, e.g. Seide et al. (2014b); Wang et al. (2018); Alistarh et al. (2017; 2018); Stich et al. (2018); Wen et al. (2017); Wangni et al. (2018); Lu & Sa (2020). We emphasize that these works consider a related, but different problem: they usually rely on assumptions on the input structure—such as second-moment bounds on the gradients—and are evaluated primarily on the practical performance of SGD, rather than isolating the variance-reduction step. (In some cases, these schemes also rely on history/error-correction (Aji & Heafield, 2017; Dryden et al., 2016; Alistarh et al., 2018; Stich et al., 2018).) As a result, they do not provide theoretical bounds on the problems we consider. In this sense, our work is closer to Suresh et al. (2017); Konečný & Richtárik (2018); Gandikota et al. (2019), which focus primarily on the *distributed estimation* problem, and only use SGD as one of many potential applications.

For example, QSGD (Alistarh et al., 2017) considers a similar problem to VarianceReduction; the major difference is that coordinates of the input vectors are assumed to be specified by 32-bit floats, rather than arbitrary real values. Hence, transmitting input vectors exactly already requires only $O(d)$ bits. They therefore focus on reducing the constant factor (and thereby improving practical performance for SGD), rather than providing asymptotic results on communication cost. They show that the expected number of bits per entry can be reduced from 32 to 2.8, at the expense of having an output variance bound in terms of *input norm* rather than *input variance*.

This is a common issue with existing quantization schemes, which leads to non-trivial complications when applying quantization to gradient descent and variance-reduced SGD (Künstner, 2017) or to model-averaging SGD (Lu & Sa, 2020), since in this case the inputs are clearly not centered around the origin. The standard way to circumvent this issue, adopted by the latter two references, but also by other work on quantization Mishchenko et al. (2019),

is to carefully adapt the quantization scheme and the algorithm to remove this issue, for instance by quantizing *differences* with respect to the last quantization point. These approaches, however, do not provide improvement as 'one-shot' quantization methods, and instead rely on historical information and properties of SGD or the function to optimize (such as smoothness). They are therefore inherently application-specific. Our method, by contrast, does not require "manual" centering of the iterates, and does not require storage of previous iterates, or any properties thereof.

Konečný & Richtárik (2018) study MeanEstimation under similar assumptions, and are the only prior work to use quantization centered around points *other* than the origin. However, again prior knowledge about the input distribution must be assumed for their scheme to provide any improvements.

Suresh et al. (2017) study the MeanEstimation problem defined on real-valued input vectors. They present a series of quantization methods, providing an $O(\frac{1}{n^2}\sum_{v\leq n}\|\boldsymbol{x}_v\|_2^2)$ upper bound, and corresponding lower bounds. Recent work by Gandikota et al. (2019) studies VarianceReduction, and uses multi-dimensional quantization techniques. However, their focus is on protocols using $o(d)$-bit messages per machine (which we show *cannot* reduce input variance). They do give two quantization methods using $\Theta(d)$-bit messages. Of these, one gives an $O(\frac{1}{n}\max_{v\leq n}\|\boldsymbol{x}_v\|_2^2)$ bound on output variance, similar to the bound of Suresh et al. (2017) for MeanEstimation (the other is much less efficient since it is designed to achieve a privacy guarantee). Mayekar & Tyagi (2020) obtain a similar error bound but with slightly longer $\Theta(d\log\log\log(\log^* d))$-bit messages.

All of the above works provide output error bounds based on the norms of input vectors. This is only optimal under the implicit assumption that inputs are centered around the origin. In Section G we provide evidence that this assumption does not hold in some practical scenarios, where the input (gradient) variance can be much lower than the input (gradient) norm: intuitively, for SGD, input variance is only close to squared norm when true gradients are close to $\mathbf{0}$, i.e., the optimization process is already almost complete.

2 Our Results

In this work, we argue that it is both stronger and more natural to bound output variance in terms of input variance, rather than squared norm. We devise optimal quantization schemes for MeanEstimation and VarianceReduction, and prove matching lower bounds, regardless of input norms. We summarize the main ideas that lead to these results.

2.1 Lattice-based Quantization

The reason that all prior works obtain output variance bounds in terms of input norm rather than variance is that they employ sets of quantization points which are *centered* around the origin $\mathbf{0}$. We instead cover the entire space $\mathbb{R}^d$ with quantization points that are in some sense uniformly spaced, using *lattices*.

Lattices are subgroups of $\mathbb{R}^d$ consisting of the integer combinations of a set of basis vectors. It is well-known (Minkowski, 1911) that certain lattices have desirable properties for covering and packing Euclidean space, and lattices have been previously used for some other applications of quantization (see, e.g., Gibson & Sayood (1988)), though mostly only in low dimension. By choosing an appropriate of lattices, we show that any vector in $\mathbb{R}^d$ can be rounded (in a randomized, unbiased fashion) to a nearby lattice point, but also that there are not too many nearby lattice points, so the correct one can be specified using few bits.

Lattices contain an infinite number of points, and therefore any encoding using a finite number of bits must use bit-strings to refer to an infinite amount of lattice points. To allow the receiver in our quantization method to correctly decode the intended point, we utilize the fact that we have a bound on the distance between any two machines' inputs (y for MeanEstimation, and $O(\sigma\sqrt{n})$ (probabilistically, by Chebyshev's inequality) for VarianceReduction). Therefore, if all points that map to the same bit-string are sufficiently far apart, a machine can correctly decode based on proximity to its own input.

The simplest version of our lattice quantization algorithm can be described as follows

- To encode $\boldsymbol{x}_u$, map to one of a set of nearby lattice points forming a convex hull around $\boldsymbol{x}_u$. Denote this point by $\boldsymbol{z}$.
- Send $\boldsymbol{z} \bmod q$ under the lattice basis: q is the quantization precision parameter.
- To decode with respect to $\boldsymbol{x}_v$, output the closest lattice point to $\boldsymbol{x}_v$ matching $\boldsymbol{z} \bmod q$.

By showing that $\boldsymbol{x}_u$ is contained within a convex hull of nearby lattice points, we can round to one of these points randomly to obtain $\boldsymbol{z}$ such that the expectation of $\boldsymbol{z}$ is $\boldsymbol{x}_u$ itself, thereby ensuring *unbiasedness*. This is because a point within a convex hull can be expressed as a linear combination of its vertices with coefficients in $[0, 1]$, which we can use as rounding probabilities.

Our reason for using $\bmod q$ with respect to the lattice basis in order to encode lattice points into bit-strings is that by exploiting properties of the particular lattices we employ, we can show a lower bound on the distance between points encoded with the same bit-string, while also controlling the number of bits we use. Then, since points encoded with the same string are sufficiently far apart, our proximity-based decoding procedure can determine the correct point. We also have a parameter ϵ which controls the granularity of the lattice used. This method of lattice-based quantization gives the following guarantee for communicating a vector between two parties:

Theorem 1. *For any $q = \Omega(1)$, any $\epsilon > 0$, and any two parties u, v holding input vectors $\boldsymbol{x}_u, \boldsymbol{x}_v \in \mathbb{R}^d$ respectively, there is a quantization method in which u sends $O(d \log q)$ bits to v, and if $\|\boldsymbol{x}_u - \boldsymbol{x}_v\| = O(q\epsilon)$, v can recover an unbiased estimate $\boldsymbol{z}$ of $\boldsymbol{x}_u$ with $\|\boldsymbol{z} - \boldsymbol{x}_u\| = O(\epsilon)$.*

Details are given in Appendix A. This result is very general, and can be applied not only to DME but to any application in which high-dimensional vectors are communicated, in order to reduce communication.

2.2 Upper Bounds

Next, we show how to apply our quantization procedure to MeanEstimation and VarianceReduction. We wish to gather (quantized estimates of) machines' inputs to a single machine, which computes the average, and broadcasts a quantized estimate of this average to all other machines. For simplicity of analysis we do so using a star and a binary tree as our communications structures, but any connected communication topology would admit such an approach. The basic star topology algorithm can be described as follows:

- All machines u send $\boldsymbol{x}_u$, quantized with $\epsilon = \Theta(y)$ and precision parameter q, to the leader machine v.
- Machine v decodes all received vectors, averages them, and broadcasts the result, using the same quantization parameters.

By choosing the leader randomly we can obtain tight bounds in expectation on the number of communication bits used per machine, and by using a more balanced communication structure such as a binary tree we can extend these bounds in expectation to hold with certainty. Further details of these algorithmic results can be found in Appendix B.

Theorem 2. *For any $q = \Omega(1)$,* MeanEstimation *can be performed with each machine using strictly $O(d \log q)$ communication bits, with $O(\frac{y^2}{q})$ output variance.*

Theorem 3. VarianceReduction *can be performed using strictly $O(d \log n)$ bits, with $O(\frac{\sigma^2}{n})$ output variance, succeeding with high probability.*

These results give optimal communication-variance bounds for these problems. However, to make this bounds practical, we address two main challenges.

Challenge 1: Input Variance. One difficulty with our approach is that we assume a known estimate of input variance (y^2 for MeanEstimation, σ^2 for VarianceReduction). Furthermore, in VarianceReduction, even if our input variance estimate is correct, some pairs of inputs can be further apart, since the bound is probabilistic.

To address this problem, we develop a mechanism for *error correction*, which allows the receiver to detect if the encode and decode vectors ($\boldsymbol{x}_u$ and $\boldsymbol{x}_v$ respectively) are too far

apart for successful decoding. In this way, if our estimate of input variance proved too low, we can increase either it or the number of bits used for quantization until we succeed. Details are presented in Appendix C; the main idea is that rather than using $\bmod q$ to encode lattice points as bit-strings, we use a more sophisticated *coloring* of the lattice and a new encoding procedure to ensure if $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$ are far apart, with high probability the encoder v chooses a color which is not used by any nearby point to $\boldsymbol{x}_u$, and therefore u can tell that $\boldsymbol{x}_v$ was not nearby.

As an application, we obtain an algorithm for VarianceReduction which uses an optimal expected number of bits per machine (except for an additive $\log n$, which in our applications is assumed to be far smaller than $d \log q$):

Theorem 4. *For any $q = \Omega(1)$,* VarianceReduction *can be performed using $O(d \log q + \log n)$ communication bits per machine in expectation, with $O(\frac{\sigma^2}{q} + \frac{\sigma^2}{n})$ output variance, succeeding with high probability.*

Challenge 2: Computational Tractability. Another issue is that known lattices which are optimal for ℓ_1 and ℓ_2-norms can be computationally prohibitive to generate and use for problems in high dimension. This is discussed further in Appendix D. We show that if we instead use the standard *cubic lattice*, which is optimal under ℓ_∞-norm and admits straightforward $\tilde{O}(d)$-computation encoding and decoding algorithms, in combination with a structured random rotation using the Walsh-Hadamard transform as proposed by Suresh et al. (2017), we can come within a log-factor variance of the optimal bounds of Theorems 2, 3, and 4.

Theorem 5. *Using the cubic lattice with a random rotation, we achieve the following output variances under ℓ_2-norm (succeeding with high probability):*

- $O(\frac{y^2 \log nd}{q})$ *for* MeanEstimation*, using strictly $O(d \log q)$ bits;*
- $O(\frac{\sigma^2 \log d}{n})$ *for* VarianceReduction*, using strictly $O(d \log n)$ bits;*
- $O(\frac{\sigma^2 \log nd}{q} + \frac{\sigma^2}{n})$ *for* VarianceReduction*, using $O(d \log q + \log n)$ bits in expectation.*

Furthermore, each machine need perform only $\tilde{O}(d)$ computation in expectation.

3 Lower Bounds

We next show matching lower bounds on the communication required for MeanEstimation and VarianceReduction. These results bound the number of bits a machine must receive (from any source) to output an estimate of sufficient accuracy, via an information-theoretic argument, and therefore apply to almost any model of communication. Our proofs essentially argue that, if a machine receives only a small amount of bits during the course of an algorithm, it has only a small number of possible (expected) outputs. We can therefore find instances such that our desired output ($\boldsymbol{\mu}$ or $\boldsymbol{\nabla}$) is far from any of these possible outputs. This argument is complicated, however, by the probabilistic nature of outputs (and, in the case of the VarianceReduction problem, inputs).

Theorem 6. *For any* MeanEstimation *algorithm in which any machine receives at most b bits* in expectation*,*

$$\mathbf{E}\left[\|\boldsymbol{EST} - \mu\|^2\right] = \Omega(y^2 2^{-\frac{3b}{d}}) \ .$$

To achieve an output variance of $O(\frac{y^2}{q})$, we see that machines must receive $\Omega(d \log q)$ bits in expectation, matching the upper bound of Theorem 4. Similarly, we have the following tight bounds for VarianceReduction:

Theorem 7. *For any* VarianceReduction *algorithm in which all machines receive (strictly) at most b bits,*

$$\mathbf{E}\left[\|\boldsymbol{EST} - \boldsymbol{\nabla}\|^2\right] = \Omega(\sigma^2 n 2^{-\frac{2b}{d}}) \ .$$

This bound matches Theorem 3, since to reduce the variance expression to $O(\sigma^2)$ (i.e., to have *reduced* output variance compared to input variance), we require $b = \Omega(d \log n)$ bits.

Theorem 8. *For any* VarianceReduction *algorithm in which any machine receives at most b bits* in expectation*,*

$$\mathbf{E}\left[\|\boldsymbol{EST} - \boldsymbol{\nabla}\|^2\right] = \Omega(\sigma^2 2^{-\frac{3b}{d}}) \ .$$

Here we match the leading terms of Theorem 4: to reduce variance to $O(\frac{\sigma^2}{q})$, we require $\Omega(d \log q)$ bits in expectation. Note that it is well known (and implied by e.g. Braverman et al. (2016)) that output variance cannot be reduced below $O(\frac{\sigma^2}{n})$ even with unlimited communication, so Theorem 8 implies that the full variance expression in Theorem 4 is tight. These bounds are proven in Appendix E.

4 Experimental Validation

We investigate three distinct applications for our scheme: compressing gradients and models in variants of data-parallel SGD, and distributed power iteration for eigenvalue computation (Suresh et al., 2017). We implement the practical version of our algorithm (Section 2.2), which for simplicity we call RLQSGD (or LQSGD for the version without Hadamard rotation). The implemented variant ensures optimal variance, up to a $\log d$ factor (Theorem 5). We compare against QSGD (Alistarh et al., 2017), the Hadamard-based scheme of Suresh et al. (2017), as well as uncompressed baselines. Please see Appendix G for a full description of the implementation, a complete experimental report, and additional experiments.

Example 1: Compressing Gradients. We first apply compression to parallel solver for least-squares regression, which is given as input a matrix A, partitioned among the nodes, and a target vector $\boldsymbol{b}$, with the goal of finding $\boldsymbol{w}^* = \text{argmin}_{\boldsymbol{w}} ||A\boldsymbol{w} - \boldsymbol{b}||_2^2$. In this example, we generate $\boldsymbol{w}^* \in \mathbb{R}^d$ and entries of $A \in \mathbb{R}^{S \times d}$ by sampling from $\mathcal{N}(0, 1)$, and we set $\boldsymbol{b} = A\boldsymbol{w}^*$. Note that the input data (and gradients) are thus naturally normalized. We then run distributed gradient descent with our quantization scheme, using 3 bits per coordinate for all methods, and examine the characteristics of the gradients sent, as well as the convergence and variance of the overall process. Figure 1 (left) shows the gradient norms and distances for the case of two nodes executing gradient descent for $d = 100$ and 8K samples, while the center and right panels show the variance of the gradient estimate for each method and convergence, respectively.

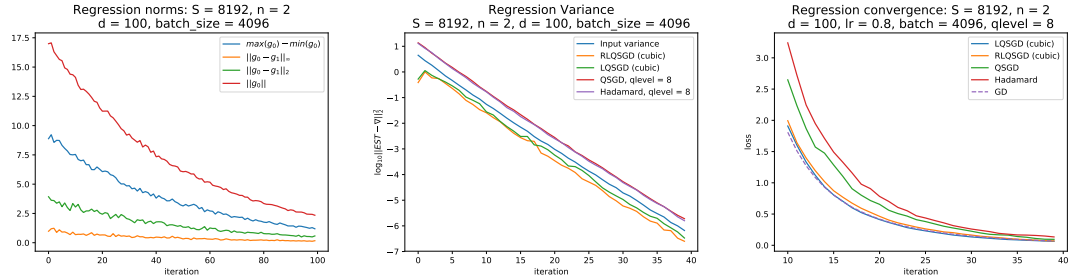

Figure 1: Gradient quantization results for the regression example.

Figure 1 (left) shows that, even in this simple normalized example, the distance between the two gradients throughout training is much smaller than the norm of the gradients themselves. Figure 1 (center) shows that both variants of our scheme (RLQSGD and LQSGD, with and without rotation) have significantly lower variance relative to Hadamard and standard QSGD, and in fact are the only schemes to get below input variance in this setting. Figure 1 (right) shows that this leads to better convergence using our method. The full report in Appendix G contains experiments on other datasets, and node counts, as well as a larger-scale application of our scheme to train neural networks. This application shows that our scheme matches or slightly improves the performance of specialized gradient compression schemes, at the same bit budget.

Example 2: Local SGD. A related example is that of compressing models in LocalSGD (Stich, 2018), where each node takes several SGD steps on its local model, followed

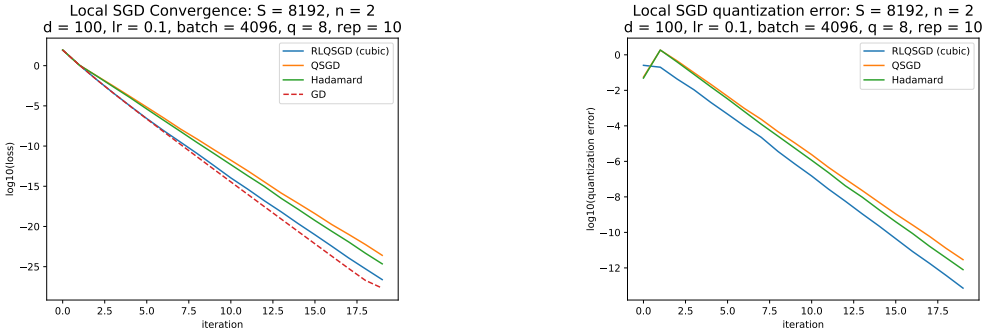

Figure 2: Local SGD: convergence for different quantizers (left) and quantization error (right).

by a global *model averaging* step, among all nodes. (Similar algorithms are popular in Federated Learning (Kairouz et al., 2019).) We use RLQSGD to quantize the models transmitted by each node as part of the averaging: to avoid artificially improving our performance, we compress the *model difference* Δ_i between averaging steps, at each node i. RQSGD is a good fit since neither the models nor the Δ_i are zero-centered. We consider the same setup as for the previous example, averaging every 10 local SGD iterations. We illustrate the convergence behavior and the quantization error in Figure 2, which shows better convergence and higher accuracy for lattice-based quantization.

Example 3: Distributed Power Iteration. The power iteration method estimates the principal eigenvector of an input matrix X, whose rows are partitioned across machines, to form input matrices X_i at the nodes. Each row of the input matrix X is generated from a multivariate gaussian with first two eigenvalues large and comparable. In each iteration, each machine updates their estimate relative to its input matrix as $\boldsymbol{u}_i = X_i^T X_i \boldsymbol{x}$, and nodes average these estimates. We apply 8-bit quantization to communicate these vectors $\boldsymbol{u}_i$, following Suresh et al. (2017). Notice that our method is a good candidate here since these estimates clearly need not be zero-centered, a fact that is evident in Figure 3 (left). Please see Appendix G.4 for a full description and additional results. Figure 3 (center) shows convergence under different quantization schemes, while Figure 3 (right) shows that our method provides significantly lower quantization error across iterations.

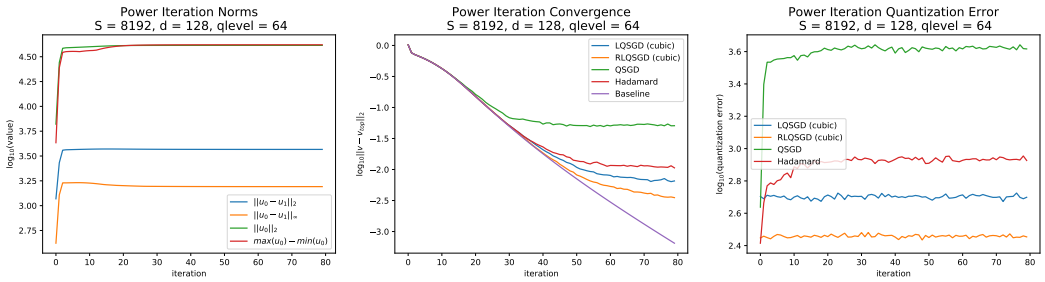

Figure 3: Power iteration: input norms (left), convergence (center) and quantization error (right).

5 Conclusions

We have argued in this work that for the problems of distributed mean estimation and variance reduction, one should measure the output variance in terms of the input variance, rather than the input norm as used by previous works. Through this change in perspective, we have shown truly optimal algorithms, and matching lower bounds, for both problems, independently of the norms of the input vectors. This improves significantly over previous work whenever the inputs are not known to be concentrated around the origin, both theoretically and in terms of practical performance. In future work, we plan to explore practical applications for variants of our schemes, for instance in the context of federated or decentralized distributed learning.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

Nir Ailon and Bernard Chazelle. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39(1):302–322, 2009. doi: 10.1137/060673096.

Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 440–445, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1045. URL `https://www.aclweb.org/anthology/D17-1045`.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pp. 1709–1720, 2017.

Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems*, pp. 5973–5983, 2018.

Tal Ben-Nun and Torsten Hoefler. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *ACM Computing Surveys (CSUR)*, 52(4):1–43, 2019.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

Mark Braverman, Ankit Garg, Tengyu Ma, Huy L Nguyen, and David P Woodruff. Communication lower bounds for statistical estimation problems via a distributed data processing inequality. In *Proceedings of the 48th Annual ACM symposium on Theory of Computing (STOC 2016)*, pp. 1011–1020, 2016.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Nikoli Dryden, Tim Moon, Sam Ade Jacobs, and Brian Van Essen. Communication quantization for data-parallel training of deep neural networks. In *2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC)*, pp. 1–8. IEEE, 2016.

Venkata Gandikota, Raj Kumar Maity, and Arya Mazumdar. vqsgd: Vector quantized stochastic gradient descent. *arXiv preprint arXiv:1911.07971*, 2019.

Jerry D. Gibson and Khalid Sayood. Lattice quantization. In *Advances in Electronics and Electron Physics*, volume 72, pp. 259 – 330. Academic Press, 1988. doi: https://doi.org/10.1016/S0065-2539(08)60560-0.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Martin Henk. A note on lattice packings via lattice refinements. *Experimental Mathematics*, 27:1–9, 09 2016. doi: 10.1080/10586458.2016.1208595.

K. J. Horadam. *Hadamard Matrices and Their Applications*. Princeton University Press, 2007. ISBN 9780691119212. URL `http://www.jstor.org/stable/j.ctt7t6pw`.

Jeremy Howard. imagenette. `https://github.com/fastai/imagenette/`. URL `https://github.com/fastai/imagenette/`.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. In *Proc. International Conference on Machine Learning (ICML)*, 2019.

Jakub Konečný and Peter Richtárik. Randomized distributed mean estimation: Accuracy vs. communication. *Frontiers in Applied Mathematics and Statistics*, 4:62, 2018. ISSN 2297-4687. doi: 10.3389/fams.2018.00062. URL `https://www.frontiersin.org/article/10.3389/fams.2018.00062`.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. `http://www.cs.toronto.edu/kriz/cifar.html`, 2014.

Frederik Künstner. Fully quantized distributed gradient descent. `http://infoscience.epfl.ch/record/234548`, 2017. URL `http://infoscience.epfl.ch/record/234548`.

Thijs Laarhoven. Sieving for closest lattice vectors (with preprocessing). In *International Conference on Selected Areas in Cryptography*, pp. 523–542. Springer, 2016.

Yucheng Lu and Christopher De Sa. Moniqua: Modulo quantized communication in decentralized SGD. In *Proc. International Conference on Machine Learning (ICML)*, 2020.

Prathamesh Mayekar and Himanshu Tyagi. Ratq: A universal fixed-length quantizer for stochastic optimization. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Hermann Minkowski. *Gesammelte Abhandlungen*. Teubner, 1911.

Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč, and Peter Richtárik. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*, 2019.

Ali Ramezani-Kebrya, Fartash Faghri, and Daniel M Roy. Nuqsgd: Improved communication efficiency for data-parallel sgd via nonuniform quantization. *arXiv preprint arXiv:1908.06077*, 2019.

C. A. Rogers. A note on coverings and packings. *Journal of the London Mathematical Society*, s1-25(4):327–331, 1950. doi: 10.1112/jlms/s1-25.4.327. URL `https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/jlms/s1-25.4.327`.

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. In *Proc. INTERSPEECH*, 2014a.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and application to data-parallel distributed training of speech dnns. In *Interspeech 2014*, September 2014b.

Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. In *Advances in Neural Information Processing Systems*, pp. 4447–4458, 2018.

Ananda Theertha Suresh, Felix X Yu, Sanjiv Kumar, and H Brendan McMahan. Distributed mean estimation with limited communication. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3329–3337, 2017.

John N Tsitsiklis and Zhi-Quan Luo. Communication complexity of convex optimization. *Journal of Complexity*, 3(3):231–243, 1987.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization. In *Advances in Neural Information Processing Systems*, pp. 14236–14245, 2019.

Hongyi Wang, Scott Sievert, Shengchao Liu, Zachary Charles, Dimitris Papailiopoulos, and Stephen Wright. Atomo: Communication-efficient learning via atomic sparsification. In *Advances in Neural Information Processing Systems*, pp. 9850–9861, 2018.

Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*, pp. 1299–1309, 2018.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 1508–1518, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

A Lattice-Based Quantization

Our quantization scheme will take the following course: to encode a vector $\boldsymbol{x}$, we will round it (in an unbiased fashion) to a nearby point in some pre-determined *lattice*. We then map lattice points to short bit-strings in such a way that points with the same bit-string are far apart and cannot be confused. The decoding procedure can then recover the intended lattice point using proximity to a *decoding vector*, in our case the decoding machine's own input. We require the following definitions:

Definition 9. *A* lattice Λ *in d dimensions is an additively-closed subgroup of $\mathbb{R}^d$, defined by a basis $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d \in \mathbb{R}^d$, and consisting of all integer combinations of the basis vectors. For any lattice $\Lambda \subseteq \mathbb{R}^d$, the* cover radius r_c *of Λ is the infimum distance r such that $\bigcup_{\boldsymbol{\lambda}\in\Lambda} B_r(\boldsymbol{\lambda}) = \mathbb{R}^d$. The* packing radius r_p *of Λ is the supremum distance r such that $\bigcap_{\boldsymbol{\lambda}\in\Lambda} B_r(\boldsymbol{\lambda}) = \emptyset$.*

Based on these notions, can bound the number of lattice points in any ball in $\mathbb{R}^d$.

Lemma 10. *Let $\Lambda \subset \mathbb{R}^d$ be a lattice, with cover radius r_c and packing radius r_p. Then, for any $\boldsymbol{x} \in \mathbb{R}^d$, $\delta > 0$, $\left(\frac{\delta - r_c}{r_c}\right)^d \leq |B_\delta(\boldsymbol{x}) \cap \Lambda| \leq \left(\frac{\delta + r_p}{r_p}\right)^d$.*

Proof. We fix a point $\boldsymbol{x} \in \mathbb{R}^d$ and upper-bound the number of points in Λ within distance δ of it: consider the ball $B_{\delta+r_p}(\boldsymbol{x})$. For any point $\boldsymbol{y}$ within distance δ of $\boldsymbol{x}$, $B_{r_p}(\boldsymbol{y}) \subset B_{\delta+r_p}(\boldsymbol{x})$. It is also the case that for any $\boldsymbol{z} \neq \boldsymbol{y} \in \Lambda$, $B_{r_p}(\boldsymbol{z}) \cap B_{r_p}(\boldsymbol{y}) = \emptyset$. So,

$$|B_\delta(\boldsymbol{x}) \cap \Lambda| \leq \frac{Vol(B_{\delta+r_p})}{Vol(B_{r_p})} \quad .$$

Under any norm, the ratio $\frac{Vol(B_{r_1})}{Vol(B_{r_2})}$ of the volumes of two balls is $(\frac{r_1}{r_2})^d$. So:

$$|B_\delta(\boldsymbol{x}) \cap \Lambda| \leq \left(\frac{\delta + r_p}{r_p}\right)^d \quad .$$

The lower bound follows similarly: $B_{\delta-r_c}(\boldsymbol{x}) \subset \bigcup_{y\in\Lambda\cap B_\delta(\boldsymbol{x})} B_{r_c}(\boldsymbol{y})$, so

$$|B_\delta(\boldsymbol{x}) \cap \Lambda| \geq \frac{Vol(B_{\delta-r_c})}{Vol(B_{r_c})} = \left(\frac{\delta - r_c}{r_c}\right)^d \quad .$$

□

For our quantization purposes, we want lattices which have low r_c (so that we can always find a close lattice point to quantize to), and high r_p (so that, by Lemma 10, there are not too many nearby lattice points, and therefore we can specify one with few bits). Ideally we want $r_c = O(r_p)$. Such lattices have long been known to exist:

Theorem 11. *For any d, under ℓ_1, ℓ_2, or ℓ_∞-norm distances, there exists a lattice $\Lambda \subset \mathbb{R}^d$ with $r_c \leq 3r_p$.*

Proof. For ℓ_1 or ℓ_2-norm (or indeed any ℓ_p norm, $p \geq 1$), see Rogers (1950), or Proposition 1.1 of Henk (2016). Under ℓ_∞ norm, the standard cubic lattice (i.e. with the standard basis as lattice basis) clearly has this property (in fact, $r_c = r_p$). □

We will call a lattice Λ an ϵ-lattice if $\epsilon = r_p \leq r_c \leq 3\epsilon$. Note that r_p, r_c scale with the lattice basis, and therefore Theorem 11 implies that for any of the three norms, an ϵ-lattice exists for any $\epsilon > 0$ by scaling appropriately. We will denote such an epsilon lattice Λ_ϵ.

A.1 Lattice Coloring

Rounding a vector $\boldsymbol{x} \in \mathbb{R}^d$ to a nearby lattice point is not sufficient for it to be communicated using a bounded number of bits, since there are still an infinite number of lattice points (though we have reduced from the uncountably infinite $\mathbb{R}^d$ to the countably infinite Λ). So, we must encode an infinite subset of lattice points using the same bit-string, and we want to ensure that the points in this subset are far apart, so that a receiver can identify which point the encoder intended. There is a natural way to do this: we simply take a coordinate-wise (with respect to the lattice basis) mod operation, thereby mapping the infinite number of points into a finite set of *color classes*.

That is, given a lattice Λ, and any positive integer q, we define a coloring procedure c_q as follows: for any lattice point $\boldsymbol{\lambda}$, represent $\boldsymbol{\lambda}$ as an integer combination of the canonical basis vectors of Λ, i.e. $\boldsymbol{\lambda} = \alpha_1 \boldsymbol{b}_1 + \cdots + \alpha_d \boldsymbol{b}_d$. Then, we set $c_i(\boldsymbol{\lambda}) = \alpha_i \bmod q$, i.e. we obtain $c(\boldsymbol{\lambda})$ by applying the mod operation coordinate-wise to each of the entries α_i. We can then encode $c(\boldsymbol{\lambda})$ using $d \log q$ bits, since there are q possible integer values for each coordinate.

Lemma 12. *For any ϵ-lattice Λ_ϵ, points $\boldsymbol{\lambda}_1 \neq \boldsymbol{\lambda}_2 \in \Lambda_\epsilon$ with $c_q(\boldsymbol{\lambda}_1) = c_q(\boldsymbol{\lambda}_2)$ have $\|\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2\| \geq 2q\epsilon$.*

Proof. Since $c_q(\boldsymbol{\lambda}_1) = c_q(\boldsymbol{\lambda}_2)$, the vector $\frac{1}{q}(\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2)$ must have integer coordinates under the canonical basis of Λ_ϵ. Therefore it is a point in Λ_ϵ. So, $\|\frac{1}{q}(\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2)\| \geq 2\epsilon$, since otherwise $B_\epsilon(\frac{1}{q}(\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2)) \cap B_\epsilon(0) \neq \emptyset$, which cannot happen since $r_p = \epsilon$. Then $\|\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2\| \geq 2q\epsilon$. □

A.2 Encoding Procedure

We are now ready to define our parameterized quantization procedure $Q_{\epsilon,q} : \mathbb{R}^d \to \{0,1\}^{d \log q}$ which maps input vectors to subsets of a *lattice*, specified with $d \log q$ bits: let Λ_ϵ be an ϵ-lattice, and let $\boldsymbol{x}$ be a fixed vector to quantize. We show that the convex hull of nearby lattice points contains $\boldsymbol{x}$:

Lemma 13. *Any $\boldsymbol{x}$ is within the convex hull of $B_{7\epsilon}(\boldsymbol{x}) \cap \Lambda_\epsilon$.*

Proof. We show that for any nonzero vector $\boldsymbol{w} \in \mathbb{R}^d$, there exists $\boldsymbol{\lambda} \in B_{7\epsilon}(\boldsymbol{x}) \cap \Lambda_\epsilon$ such that $\langle \boldsymbol{w}, \boldsymbol{\lambda} \rangle > \langle \boldsymbol{w}, \boldsymbol{x} \rangle$; it is well-known that this implies that $\boldsymbol{x}$ is in the convex hull of $B_{7\epsilon}(\boldsymbol{x}) \cap \Lambda_\epsilon$. For ℓ_1 and ℓ_2 norms, let $\boldsymbol{\lambda}$ be the closest point in Λ_ϵ to $\boldsymbol{x} + 4\epsilon \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$. Since $r_c \leq 3\epsilon$, $\|\boldsymbol{\lambda} - (\boldsymbol{x} + 4\epsilon \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|})\| < 3\epsilon$ (and so $\boldsymbol{\lambda} \in B_{7\epsilon}(\boldsymbol{x})$). So,

$$\begin{aligned}
\langle \boldsymbol{w}, \boldsymbol{\lambda} \rangle &= \langle \boldsymbol{w}, \boldsymbol{x} + 4\epsilon \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} + \boldsymbol{\lambda} - (\boldsymbol{x} + 4\epsilon \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}) \rangle \\
&= \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \langle \boldsymbol{w}, 4\epsilon \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \rangle + \langle \boldsymbol{w}, \boldsymbol{\lambda} - (\boldsymbol{x} + 4\epsilon \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}) \rangle \\
&\geq \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \frac{4\epsilon \|\boldsymbol{w}\|^2}{\|\boldsymbol{w}\|} - \|\boldsymbol{w}\| \cdot \|\boldsymbol{\lambda} - (\boldsymbol{x} + 4\epsilon \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|})\| \\
&> \langle \boldsymbol{w}, \boldsymbol{x} \rangle + 4\epsilon \|\boldsymbol{w}\| - 3\epsilon \|\boldsymbol{w}\| > \langle \boldsymbol{w}, \boldsymbol{x} \rangle \ .
\end{aligned}$$

Here the first inequality uses Hölder's inequality (and, in the case of ℓ_1 norm, the fact that $\|\boldsymbol{\lambda} - (\boldsymbol{x} + 4\epsilon \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|})\|_\infty \leq \|\boldsymbol{\lambda} - (\boldsymbol{x} + 4\epsilon \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|})\|_1$).

For ℓ_∞ norm, let $\boldsymbol{w}'$ be the vector $4\epsilon \cdot sign(\boldsymbol{w})$, i.e., the vector containing with entries 4ϵ at each positive coordinate of $\boldsymbol{w}$ and -4ϵ for each negative one. Then, $\|\boldsymbol{w}'\|_\infty = 4\epsilon$. Let $\boldsymbol{\lambda}$ be the closest point in Λ_ϵ to $\boldsymbol{x} + \boldsymbol{w}'$; as before, we have $\|\boldsymbol{\lambda} - (\boldsymbol{x} + \boldsymbol{w}')\|_\infty \leq 3\epsilon$ and $\boldsymbol{\lambda} \in B_{7\epsilon}(\boldsymbol{x})$. Then,

$$\begin{aligned}
\langle \boldsymbol{w}, \boldsymbol{\lambda} \rangle &= \langle \boldsymbol{w}, \boldsymbol{x} + \boldsymbol{w}' + \boldsymbol{\lambda} - (\boldsymbol{x} + \boldsymbol{w}') \rangle \\
&= \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \langle \boldsymbol{w}, \boldsymbol{w}' \rangle + \langle \boldsymbol{w}, \boldsymbol{\lambda} - (\boldsymbol{x} + \boldsymbol{w}') \rangle \\
&\geq \langle \boldsymbol{w}, \boldsymbol{x} \rangle + 4\epsilon \|\boldsymbol{w}\|_1 - \|\boldsymbol{w}\|_1 \cdot \|\boldsymbol{\lambda} - (\boldsymbol{x} + \boldsymbol{w}')\|_\infty \\
&> \langle \boldsymbol{w}, \boldsymbol{x} \rangle + 4\epsilon \|\boldsymbol{w}\|_1 - 3\epsilon \|\boldsymbol{w}\|_1 > \langle \boldsymbol{w}, \boldsymbol{x} \rangle \ .
\end{aligned}$$

Again, the first inequality uses Hölder's inequality. □

We can therefore show that we can probabilistically map $\boldsymbol{x}$ to these nearby lattice points in such a way that the expectation of the result is $\boldsymbol{x}$:

Enumerate the points in $B_{7\epsilon}(\boldsymbol{x}) \cap \Lambda_\epsilon$ as $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_z$. Since $\boldsymbol{x}$ is within the convex hull of $B_{7\epsilon}(\boldsymbol{x}) \cap \Lambda_\epsilon$, there must be some sequence of non-negative coefficients $a_1, \ldots, a_z$ such that $\sum_{i=1}^z a_i \boldsymbol{\lambda}_i = \boldsymbol{x}$. Let $\boldsymbol{z}$ be a random vector taking value $\boldsymbol{\lambda}_i$ with probability $\frac{a_i}{\sum_{i=1}^z a_i}$, for all i. Then, $\mathbf{E}[\boldsymbol{z}] = \boldsymbol{x}$, and $\|\boldsymbol{x} - \boldsymbol{z}\| < 7\epsilon$ (and so $\boldsymbol{z}$ is an unbiased estimator of $\boldsymbol{x}$ with variance most $(7\epsilon)^2 \leq 49\epsilon^2$).

We then set $Q_{\epsilon,q}(\boldsymbol{x})$ to be the color class of $\boldsymbol{z}$ in the coloring c_q of Λ_ϵ, which can be specified in $d \log q$ bits. $Q_{\epsilon,q}(\boldsymbol{x})$ now corresponds to a subset Λ'_ϵ containing $\boldsymbol{z}$, such that any two elements of Λ'_ϵ are of distance at least $2q\epsilon$ apart.

We summarize this process with pseudocode, taking as input the quantization parameters ϵ and q, Λ_ϵ, and the vector $\boldsymbol{x} \in \mathbb{R}^d$ to quantize.

Algorithm 1 LatticeEncode, to compute $Q_{\epsilon,q}(\boldsymbol{x})$

Let $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_z \in B_{7\epsilon}(\boldsymbol{x}) \cap \Lambda_\epsilon$, and $a_1, \ldots, a_z \geq 0$ such that $\sum_{i=1}^z a_i \boldsymbol{\lambda}_i = \boldsymbol{x}$.
Let $\boldsymbol{z}$ be a random vector taking value $\boldsymbol{\lambda}_i$ with probability $\frac{a_i}{\sum_{i=1}^z a_i}$, for all i.
Output $c_q(\boldsymbol{z})$, i.e., express $\boldsymbol{z}$ in the canonical basis of Λ_ϵ and take $\bmod\ q$ coordinate-wise.

We also summarize the properties of our quantization scheme in the following result:

Lemma 14. *There is a function $Q_{\epsilon,q} : \mathbb{R}^d \to \{0,1\}^b$ which maps each $\boldsymbol{x} \in \mathbb{R}^d$ to a string $Q_{\epsilon,q}(\boldsymbol{x})$ of $b = d \log q$ bits, specifying a subset $\Lambda'_\epsilon \subset \Lambda_\epsilon$ with the following properties: there exists $\boldsymbol{z} \in \Lambda'_\epsilon$ such that $\boldsymbol{z}$ is an unbiased estimator of $\boldsymbol{x}$ with $\|\boldsymbol{z} - \boldsymbol{x}\| < 7\epsilon$, and for all $\boldsymbol{w} \in \Lambda'_\epsilon \setminus \{\boldsymbol{z}\}$, $\|\boldsymbol{z} - \boldsymbol{w}\| \geq 2q\epsilon$.*

A.3 Decoding Procedure

We must now also define a procedure $R_{\epsilon,q} : \{0,1\}^{d \log q} \times \mathbb{R}^d \to \mathbb{R}^d$ to decode quantized values, using a machine's own input $\boldsymbol{x}_v$ (as the second input to the function): to do so, we simply take the point in the subset Λ'_ϵ encoded by the received quantized value $Q_{\epsilon,q}(\boldsymbol{x})$ which is closest to $\boldsymbol{x}_v$:

Algorithm 2 LatticeDecode, to compute $R_{\epsilon,q}(Q_{\epsilon,q}(\boldsymbol{x}), \boldsymbol{x}_v)$

Let $\Lambda'_\epsilon := \{\mathbf{s} \in \Lambda_\epsilon : c_q(\mathbf{s}) = Q_{\epsilon,q}(\boldsymbol{x})\}$, the subset of lattice points matching $Q_{\epsilon,q}(\boldsymbol{x})$.
Output the closest point in Λ'_ϵ to $\boldsymbol{x}_v$.

Lemma 15. *If $\|\boldsymbol{x} - \boldsymbol{x}_v\| \leq (q-7)\epsilon$, then the decoding procedure $R_{\epsilon,q}(Q_{\epsilon,q}(\boldsymbol{x}), \boldsymbol{x}_v)$ correctly returns the vector $\boldsymbol{z}$ which is an unbiased estimator of $\boldsymbol{x}$.*

Proof. We upper-bound the distance to $\boldsymbol{z}$:

$$\|\boldsymbol{x}_v - \mathbf{z}\| \leq \|\boldsymbol{x}_v - \boldsymbol{x}\| + \|\boldsymbol{x} - \mathbf{z}\| < (q-7)\epsilon + 7\epsilon = q\epsilon \ ,$$

and lower-bound the distance to any other point $\mathbf{y} \in \Lambda'_\epsilon \setminus \{\boldsymbol{z}\}$:

$$\|\boldsymbol{x}_v - \mathbf{y}\| \geq \|\mathbf{y} - \mathbf{z}\| - \|\boldsymbol{x}_v - \mathbf{z}\| > 2q\epsilon - q\epsilon = q\epsilon \ .$$

Therefore $\mathbf{z}$ is the closest point in Λ'_ϵ to $\boldsymbol{x}_v$, and will be returned by the decoding procedure. □

Lemmas 14 and 15 together then imply our main theorem about the properties of our pairwise encode-decode process, Theorem 1.

B Algorithms

We now apply our quantization procedure to MeanEstimation and VarianceReduction. We first give a simple algorithm using a star topology, in which machines communicate directly with a randomly chosen *leader*, in order to prove communication bounds in expectation. Then, we show how to use a binary-tree communication structure in order to obtain these same bound with certainty on each machine.

B.1 Star-Topology Algorithm

Algorithm 3 MeanEstimation

Nominate one machine v at random to be *leader*
All other machines u send $Q_{\epsilon,q}(\boldsymbol{x}_u)$ to v (and v simulates sending $Q_{\epsilon,q}(\boldsymbol{x}_v)$)
Machine v decodes (using $\boldsymbol{x}_v$) and averages all received inputs to obtain $\hat{\boldsymbol{\mu}}$
Machine v sends $Q_{\epsilon,q}(\hat{\boldsymbol{\mu}})$ to all machines
All machines u decode (using $\boldsymbol{x}_u$) and output the result

Theorem 16. *For any $q = \Omega(1)$, Algorithm 3 performs* MeanEstimation *using $O(d \log q)$ communication bits per machine in expectation, with $O(\frac{y^2}{q})$ output variance.*

Proof. Let $q \geq 28$, and $\epsilon = \frac{2y}{q}$. We then have $(q-7)\epsilon \geq \frac{3}{4}q\epsilon = 1.5y$. By Lemma 15, for any $\boldsymbol{x}$, $\boldsymbol{z}$ with $\|\boldsymbol{x} - \boldsymbol{z}\| \leq 1.5y \leq (q-7)\epsilon$, decoding $R_{\epsilon,q}(Q_{\epsilon,q}(\boldsymbol{x}), \boldsymbol{z})$ is successful. Therefore, v correctly decodes all received messages (since all inputs are within distance y), obtaining independent unbiased estimates of each machine's input, each within distance 7ϵ, and therefore with variance at most $49\epsilon^2$. Then $\hat{\boldsymbol{\mu}}$ is an unbiased estimator of $\boldsymbol{\mu}$ with variance at most $\frac{49\epsilon^2}{n}$, and at distance at most 7ϵ from $\boldsymbol{\mu}$. $\hat{\boldsymbol{\mu}}$ is also at most distance

$y + 7\epsilon \leq 1.5y$ from any machine's input, so again by Lemma 15, all machines correctly decode the encoded $\hat{\boldsymbol{\mu}}$. The output is therefore is unbiased estimate of $\hat{\boldsymbol{\mu}}$ with variance at most $49\epsilon^2$, i.e., an unbiased estimate of $\boldsymbol{\mu}$ with variance at most $49\epsilon^2 + \frac{49\epsilon^2}{n} \leq 74\epsilon^2 = O(\frac{y^2}{q^2})$. We thereby obtain a MeanEstimation algorithm achieving $O(\frac{y^2}{q^2})$ output variance (and we simplify to $O(\frac{y^2}{q})$; this does not weaken the result because replacing q with q^2 does not asymptotically increase the number of bits required). All nodes except v use using $O(d \log q)$ communication; v uses $O(nd \log q)$. Since v is chosen uniformly at random, this gives the stated expected communication bounds. □

We can perform a simple reduction to extend Algorithm 3 to VarianceReduction:

Theorem 17. *For any $\alpha > 1$, and any q between $\Omega(1)$ and $O(n^2\alpha)$, Algorithm 3 performs* VarianceReduction *using $O(d \log q)$ communication bits per machine in expectation, with $O(\frac{\alpha n \sigma^2}{q})$ output variance, succeeding with probability at least $1 - \frac{1}{\alpha}$.*

Proof. By Chebyshev's inequality, for any $\alpha > 1$, $\mathbf{Pr}\left[\|\boldsymbol{x}_v - \boldsymbol{\nabla}\| \geq \sigma\sqrt{\alpha n}\right] \leq \frac{1}{\alpha n}$, and therefore by a union bound,

$$\mathbf{Pr}\left[\text{All inputs within pairwise distance } 2\sigma\sqrt{\alpha n}\right] \geq \mathbf{Pr}\left[\text{All inputs within distance } \sigma\sqrt{\alpha n} \text{ of } \boldsymbol{\nabla}\right] \geq 1 - \frac{1}{\alpha} \ .$$

Therefore, we can reduce to a MeanEstimation instance with $y = 2\sigma\sqrt{\alpha n}$, succeeding with probability $1 - \frac{1}{\alpha}$. Using $q = O(n^2\alpha)$ (and $\epsilon = \frac{2y}{q} = \frac{4\sigma\sqrt{\alpha n}}{q}$), we obtain an unbiased estimate of μ with $O(\frac{\alpha n \sigma^2}{q})$ variance, and therefore an unbiased estimate of $\boldsymbol{\nabla}$ with $O(\frac{\sigma^2}{n} + \frac{\alpha n \sigma^2}{q}) = O(\frac{\alpha n \sigma^2}{q})$ variance, using $O(d \log q)$ expected communication per machine. □

Notably, setting $\alpha = n^c$, for arbitrarily large constant c, and $q = \Theta(n^2\alpha)$, Theorem 17 implies that we can achieve optimal $O(\frac{\sigma^2}{n})$ output variance using $O(d \log n)$ expected communication per machine, and succeeding with high probability in n (i.e., with success probability $1 - n^{-c}$).

B.2 Tree-Topology Algorithm

Algorithm 3 uses a star topology for simplicity (all machines communicate via a randomly chosen leader v), with the result that the communication bounds are only in expectation, and machine v must perform significantly more communication. In this section, we show that, using a tree topology to more evenly distribute the work, we can obtain the communication bounds of Theorems 16 and 17 with certainty for all machines. We note that, while we use a binary tree for simplicity, any connected communication graph would suffice.

Our algorithm to achieve worst-case bounds (as opposed to bounds in expectation) on communication cost is the following (Algorithm 4). It uses a parameter m which controls both the quantization parameters and the number of input estimates to average; we will set m to achieve a trade-off in the same form as Theorem 16 later.

Algorithm 4 MeanEstimation(m)

Sample a set T of $\min(m, n)$ machines uniformly at random.
Arrange nodes into a complete binary tree, with nodes in T as leaves.
Collect estimates from T to root of tree, averaging and encoding with $Q_{\frac{y}{m^2}, m^3}$ at every step.
Compute final average at root, and broadcast to all machines (via a binary tree) encoded with $Q_{\frac{y}{m^2}, m^3}$.
Decode and output result at all machines.

We now describe in more detail the steps of the algorithm:

Sampling a set of machines We begin by sampling a set T of $\min(m, n)$ machines (so if $m \geq n$, T is simply the set M of all machines). Our goal will then be to estimate the average of the inputs $\boldsymbol{x}_v, v \in T$, and broadcast this estimate to all machines to output. If $m \geq n$ then clearly $\boldsymbol{\mu}_T := \frac{1}{n}\sum_{v \in T} \boldsymbol{x}_v$ is equal to $\boldsymbol{\mu} = \frac{1}{n}\sum_{v \in M} \boldsymbol{x}_v$. Otherwise, it is well-known that the sample mean $\boldsymbol{\mu}_T := \frac{1}{m}\sum_{v \in T} \boldsymbol{x}_v$ is an unbiased estimator of the population mean $\boldsymbol{\mu} = \frac{1}{n}\sum_{v \in M} \boldsymbol{x}_v$, with $O(\frac{y^2}{m})$ variance. Therefore, if we can guarantee that all machines output the same value $\boldsymbol{EST}$ which is an unbiased estimator of $\boldsymbol{\mu}_T$ with variance $O(\frac{y^2}{m})$, we have correctly performed MeanEstimation with output variance $O(\frac{y^2}{m})$.

Arranging a communication tree Our communication structure will be a complete binary tree, with the machines in T acting as the leaves (since we have choice of m and provide asymptotic bounds, we may assume it is a power of 2). The roles of the remaining nodes of the communication tree may be taken by any arbitrary machines, so long as all machines take only $O(1)$ roles. We will then send messages *up* the tree (i.e., from child nodes to their parents), in order to convey estimates of the inputs of the leaf nodes (machines in T) to the root, which can then compute the final average.

Collecting estimates from leaves to root When a node in the communication tree receives and decodes a vector from both of its children, it takes the average, encodes with $Q_{\frac{y}{m^2}, m^3}$, and sends the result to its parent. We will denote by $\boldsymbol{A}_v$ the average input of all descendant leaves of a tree node v. Our goal is then to show that the average computed by node v is an unbiased estimator of $\boldsymbol{A}_v$ with low variance. Since $\boldsymbol{A}_r = \boldsymbol{\mu}_T$ for the root r of the communication tree, we will then have an unbiased estimator of $\boldsymbol{\mu}_T$ (and therefore $\boldsymbol{\mu}$) as desired.

To bound the estimator error at each step, we will employ an inductive argument. Considering nodes by their depth in the tree (with leaves at depth 0 and the root r at depth $\log\min(m, n)$), we show the following:

Lemma 18. *A node v at depth i sends to its parent an unbiased estimator $\boldsymbol{a}_v$ of $\boldsymbol{A}_v$ with $\|\boldsymbol{A}_v - \boldsymbol{a}_v\| \leq \frac{7iy}{m^2}$, encoded with $Q_{\frac{y}{m^2}, m^3}$.*

Proof. By induction. Clearly the claim is true for leaf nodes at 0, which encode exactly $\boldsymbol{A}_v$ (i.e. their own input). Assuming the claim is true for i, we prove for $i + 1$:

Node v receives two values $Q_{\frac{y}{m^2}, m^3}(\boldsymbol{a}_u)$ and $Q_{\frac{y}{m^2}, m^3}(\boldsymbol{a}_w)$ from its children u and w. By the inductive assumption, $\|\boldsymbol{A}_u - \boldsymbol{a}_u\| \leq \frac{7iy}{m^2}$, and so $\|\boldsymbol{x}_v - \boldsymbol{a}_u\| \leq \|\boldsymbol{A}_u - \boldsymbol{a}_u\| + \|\boldsymbol{x}_v - \boldsymbol{A}_u\| \leq \frac{7iy}{m^2} + y < (m^3 - 7)\frac{y}{m^2}$ (for m at least a sufficiently large constant). By Lemma 15, therefore, v correctly decodes the message from u, to recover an unbiased estimator $\boldsymbol{z}_u$ of $\boldsymbol{a}_u$ with $\|\boldsymbol{z}_u - \boldsymbol{a}_u\| < \frac{7y}{m^2}$. The same argument holds for w.

Node v then takes as $\boldsymbol{a}_v$ the average of $\boldsymbol{z}_u$ and $\boldsymbol{z}_w$. Since $\|\boldsymbol{z}_u - \boldsymbol{A}_u\| \leq \|\boldsymbol{A}_u - \boldsymbol{a}_u\| + \|\boldsymbol{z}_u - \boldsymbol{a}_u\| < \frac{7iy}{m^2} + \frac{7y}{m^2} = \frac{7(i+1)y}{m^2}$ (and the same holds for w):

$$\|\boldsymbol{A}_v - \boldsymbol{a}_v\| \leq \frac{\|\boldsymbol{z}_u - \boldsymbol{A}_u\| + \|\boldsymbol{z}_w - \boldsymbol{A}_w\|}{2} < \frac{7(i+1)y}{m^2} \quad .$$

This completes the proof by induction. □

Computing the final average at the root By Lemma 18, the root node r, at depth $\log\min(m, n)$, computes an unbiased estimator $\boldsymbol{a}_r$ of $\boldsymbol{\mu}_T$ with $\|\boldsymbol{a}_r - \boldsymbol{\mu}_T\| \leq \frac{7y\log\min(m,n)}{m^2} \leq \frac{7y\log m}{m^2}$. It then encodes this vector with $Q_{\frac{y}{m^2}, m^3}$, and broadcasts it to all other machines via an arbitrary binary tree (all machines performing the role of one node in the tree). Nodes in the tree relay the *same* message to their children until all nodes have received the message. Then, all machines decode the message and output the resulting vector.

Decoding and outputting $\boldsymbol{EST}$

All machines have now received an unbiased estimator $\boldsymbol{a}_r$ of $\boldsymbol{\mu}_T$ with $\|\boldsymbol{a}_r - \boldsymbol{\mu}_T\| \leq \frac{7y \log m}{m^2}$, encoded with $Q_{\frac{y}{m^2}, m^3}$. Any machine v has

$$\|\boldsymbol{a}_r - \boldsymbol{x}_v\| \leq \|\boldsymbol{a}_r - \boldsymbol{\mu}_T\| + \|\boldsymbol{x}_v - \boldsymbol{\mu}_T\| \leq \frac{7y \log m}{m^2} + y < (m^3 - 7)\frac{y}{m^2} \ ,$$

and so by Lemma 15, v correctly decodes the message to recover an unbiased estimator $\boldsymbol{z}_r$ of $\boldsymbol{a}_r$ with $\|\boldsymbol{z}_r - \boldsymbol{a}_r\| < \frac{7y}{m^2}$.

All nodes therefore output $\boldsymbol{z}_r$, which is an unbiased estimator of $\boldsymbol{\mu}_T$, with

$$\|\boldsymbol{z}_r - \boldsymbol{\mu}_T\| \leq \|\boldsymbol{z}_r - \boldsymbol{a}_r\| + \|\boldsymbol{\mu}_T - \boldsymbol{a}_r\| \leq \frac{7y}{m^2} + \frac{7y \log m}{m^2} = O(\frac{y}{m}) \ .$$

Proof of Theorem 2. As noted earlier, an unbiased estimator of $\boldsymbol{\mu}_T$ is also an unbiased estimator of $\boldsymbol{\mu}$ (with $O(\frac{y^2}{m})$ additional variance), and $\boldsymbol{z}_r$ is therefore a correct solution to MeanEstimation with variance $\mathbf{Var}[\boldsymbol{z}_r] \leq \|\boldsymbol{z}_r - \boldsymbol{\mu}_T\|^2 + O(\frac{y^2}{m}) = O(\frac{y^2}{m})$.

We can bound the communication cost as follows: all machines have sent and received $O(1)$ vectors encoded with $Q_{\frac{y}{m^2}, m^3}$, which require $O(d \log m)$ bits. We set $m = q$ to reach an expression of the same form as Theorem 16, and obtain Theorem 2.

□

We can apply the same reduction as for Theorem 17 to obtain the following for VarianceReduction, using that a MeanEstimation algorithm with $y = 2\sigma\sqrt{\alpha n}$ solves VarianceReduction with probability at least $1 - \frac{1}{\alpha}$:

Theorem 19. *For any $\alpha > 1$, and any q between $\Omega(1)$ and $O(n^2\alpha)$, Algorithm 4 performs* VarianceReduction *using $O(d \log q)$ communication bits per machine in total, with $O(\frac{\alpha n \sigma^2}{q})$ output variance, succeeding with probability at least $1 - \frac{1}{\alpha}$.*

As a corollary to this, we can obtain Theorem 3:

Proof of Theorem 3. We set $\alpha = n^c$, for arbitrarily large constant c, and $q = \Theta(n^2\alpha)$. Theorem 19 then implies that we can achieve optimal $O(\frac{\sigma^2}{n})$ output variance using $O(d \log n)$ total communication per machine, and succeeding with high probability in n (i.e., with success probability $1 - n^{-c}$). □

C Error Detection In Quantization

In this section we equip our quantization method with a mechanism of *error detection*: that is, we wish to detect when the encode and decode vectors are too far for a successful decoding.

We first replace the coloring used in Section A.1 with a new coloring method, which will allow us to detect this kind of error with high probability.

Lemma 20. *For any Λ_ϵ and any r at least a sufficiently large constant, there exists a coloring $c_r : \Lambda_\epsilon \to [r^{O(d)}]$ such that for any $q \in \mathbb{N}$ with $4 \leq q \leq r$ and any $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2 \in \Lambda_\epsilon$ with $\|\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2\| \leq 1.5q^{q^d}\epsilon$, there are at most $2q^d$ points in $S := \Lambda_\epsilon \cap (B_{r^3\epsilon}(\boldsymbol{\lambda}_1) \cup B_{r^3\epsilon}(\boldsymbol{\lambda}_2))$ whose colors are not unique in S.*

Proof. We show that such a coloring exists by the probabilistic method: we randomly construct a candidate coloring, and prove that it satisfies the necessary properties with positive probability. Then, one such coloring must exist.

Let $f : \left[\left(r^{q^d}\right)^d\right] \to [r^{10d}]$ be a uniformly random function. Our candidate coloring c is then given by $c(\boldsymbol{\lambda}) = f(\boldsymbol{\lambda} \bmod r^{q^d})$ (where mod is taken coordinate-wise with respect to the canonical basis of the lattice Λ_ϵ). So, in effect, we divide the lattice points into r^{dq^d}

classes by taking mod r^{q^d} , and then randomly color the classes. Any two points $\boldsymbol{\lambda}, \boldsymbol{\lambda}'$ in the same class are of distance at least $2 \cdot r^{q^d}\epsilon$ apart, since $r^{-q^d}(\boldsymbol{\lambda} - \boldsymbol{\lambda}')$ is a non-zero lattice vector, and so is of length at least 2ϵ.

Fixing some pair of points $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2 \in \Lambda_\epsilon$ (and therefore some set $S = \Lambda_\epsilon \cap (B_{r^3\epsilon}(\boldsymbol{\lambda}_1) \cup B_{r^3\epsilon}(\boldsymbol{\lambda}_2)))$, we note that all pairs of points in S are at most distance $1.5q^{q^d}\epsilon + 2r^3\epsilon < 2 \cdot r^{q^d}\epsilon$ apart, so they are all in different classes, and therefore their colors were chosen independently uniformly at random.

By Lemma 10, in any ball of radius $r^3\epsilon$, there are at most $\left(\frac{r^3\epsilon+\epsilon}{\epsilon}\right)^d = \left(r^3+1\right)^d$ lattice points. Therefore, $|S| \leq 2\left(r^3+1\right)^d$. Our goal now is to show that few points in S are given non-unique colors.

Order the points in S arbitrarily. If there are more than $2q^d$ colors used by multiple points in S, then there are at least q^d points assigned a color already used by another point prior in the ordering. For any fixed set S' of q^d points in S, the probability that all receive a color already used by another point prior in the ordering is at most $\left(\frac{2\left(r^3+1\right)^d}{r^{10d}}\right)^{q^d}$, since the colors are chosen uniformly and independently, and at most $2\left(r^3+1\right)^d$ are used by prior points.

So,

$$
\begin{aligned}
\mathbf{Pr}\left[|\{\text{points with non-unique colors in } S\}| > 2q^d\right] &\leq \sum_{\substack{S' \subseteq S \\ |S'| = q^d}} \left(\frac{2\left(r^3+1\right)^d}{r^{10d}}\right)^{q^d} \\
&= \binom{2\left(r^3+1\right)^d}{q^d} \left(\frac{2\left(r^3+1\right)^d}{r^{10d}}\right)^{q^d} \\
&\leq \left(\frac{2e\left(r^3+1\right)^d}{q^d}\right)^{q^d} \left(\frac{2\left(r^3+1\right)^d}{r^{10d}}\right)^{q^d} \\
&< \left(\frac{r^{7d}}{r^{10d}}\right)^{q^d} \\
&= r^{-3dq^d} \ .
\end{aligned}
$$

We take a union bound over all pairs of classes for $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2$; this incorporates all choices for $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2$ since the coloring is identical with respect to two pairs of points from the same pair of classes. There are fewer than r^{2dq^d} such pairs, so the probability that any do not satisfy the condition is at most r^{-dq^d}. Finally, we take a union bound over all q with $4 \leq q \leq r$, and see that the probability that any such q does not satisfy the condition is less than $\sum_{q=4}^{r} r^{-dq^d} < 1$. Therefore, we have a positive probability that c satisfies the criteria of the lemma, so by the probabilistic method, such a good coloring must exist.

□

We utilize this coloring in the following way: when encoding a vector $\boldsymbol{x}_u$ with a lattice point $\boldsymbol{z}$, we can now ensure that for any decode vector $\boldsymbol{x}_v$, with high probability, the *color* of $\boldsymbol{z}$ is unique among points within $r^3\epsilon$ of $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$. Therefore, when machine v decodes, it either recovers $\boldsymbol{z}$ or knows that $\|\boldsymbol{x}_v - \boldsymbol{z}\| > r^3\epsilon$.

Now that we have means of detecting whether the encode and decode vectors are too far apart for successful decoding, we can design a process that iteratively attempts to quantize using more bits until decoding succeeds:

Algorithm 5 RobustAgreement(ϵ, q)

Let $\hat{\boldsymbol{x}}_u$ be chosen uniformly at random from $B_{q^2\epsilon}(\boldsymbol{x}_u)$.
Map $\hat{\boldsymbol{x}}_u$ to a lattice point $\boldsymbol{z} \in B_{7\epsilon}(\hat{\boldsymbol{x}}) \cap \Lambda_\epsilon$ in unbiased fashion, as in Lemma 14.
$r \leftarrow q$
loop
  Machine u sends $c_r(\boldsymbol{z})$ to v
  Machine v computes $\boldsymbol{y} \in \Lambda_\epsilon$, the closest lattice point to $\boldsymbol{x}_v$ such that $c_r(\boldsymbol{y}) = c_r(\boldsymbol{z})$
  if $\|\boldsymbol{x}_v - \boldsymbol{y}\| \leq \frac{1}{2}r^3\epsilon$ **then**
    v outputs $\boldsymbol{y}$, procedure terminates
  else
    v sends the message Far to u
    $r \leftarrow r^2$
  end if
end loop

Here the purpose of first choosing a vector $\hat{\boldsymbol{x}}_u$ uniformly from $B_{q^2\epsilon}(\boldsymbol{x}_u)$ is to ensure that the probability that $\boldsymbol{z}$ takes any particular value is low, since there are a small number of points which do *not* have non-unique colors which we wish to avoid with high probability.

Lemma 21. *Lattice point $\boldsymbol{z}$ is an unbiased estimator of $\boldsymbol{x}_u$, with $\|\boldsymbol{z} - \boldsymbol{x}_u\| \leq (q^2+7)\epsilon$. Furthermore, the probability of $\boldsymbol{z}$ taking any particular value $\hat{\boldsymbol{z}}$ is at most $(\frac{7}{q^2})^d$.*

Proof. By our choice of $\boldsymbol{z}$ we clearly have $\|\boldsymbol{z} - \boldsymbol{x}_u\| \leq (q^2+7)\epsilon$; unbiasedness follows from Lemma 14. For $\boldsymbol{z}$ to take some particular value $\hat{\boldsymbol{z}}$, we must have $\mathbf{Pr}\left[\|\hat{\boldsymbol{z}} - \hat{\boldsymbol{x}}\| \leq 7\epsilon\right]$, which is the case with probability at most $\frac{Vol(B_{7\epsilon})}{Vol(B_{q^2\epsilon})} = (\frac{7}{q^2})^d$. □

We next analyze what the properties of the communication procedure within the loop of Algorithm 5, for some fixed r:

Lemma 22. *During a loop of* RobustAgreement*:*

- *$c_r(\boldsymbol{z})$ takes $O(d \log r)$ bits to send.*
- *If $\|\boldsymbol{x}_u - \boldsymbol{x}_v\| \leq \frac{1}{2}r^3\epsilon$, with probability at least $1 - (\frac{14}{q})^d$, v outputs $\boldsymbol{z}$.*
- *If $\|\boldsymbol{x}_u - \boldsymbol{x}_v\| \leq q^{q^d}\epsilon$, with probability at least $1 - (\frac{14}{q})^d$, v either outputs $\boldsymbol{z}$ or sends* Far *to u.*

Proof. $c_r(\boldsymbol{z})$ is a color from $[r^{O(d)}]$, which takes $O(d \log r)$ bits to specify.

Let $\boldsymbol{\lambda}_u$ be the closest lattice point to $\boldsymbol{x}_u$, and $\boldsymbol{\lambda}_v$ be the closest lattice point to $\boldsymbol{x}_v$. By the properties of an ϵ-lattice, $\|\boldsymbol{\lambda}_u - \boldsymbol{x}_u\|$ and $\|\boldsymbol{\lambda}_v - \boldsymbol{x}_v\|$ are at most 3ϵ.

If $\|\boldsymbol{x}_u - \boldsymbol{x}_v\| \leq q^{q^d}\epsilon$, then $\|\boldsymbol{\lambda}_u - \boldsymbol{\lambda}_v\| \leq q^{q^d}\epsilon + 6\epsilon \leq 1.5q^{q^d}$. So, by Lemma 20, there are at most $2q^d$ points with non-unique colors in $S = \Lambda_\epsilon \cap (B_{r^3\epsilon}(\boldsymbol{\lambda}_u) \cup B_{r^3\epsilon}(\boldsymbol{\lambda}_v))$. We have $\boldsymbol{z} \in \Lambda_\epsilon \cap (B_{r^3\epsilon}(\boldsymbol{\lambda}_u)) \subset S$, so the probability of its color being unique in S is at least $1 - 2q^d \cdot (\frac{7}{q^2})^d = 1 - (\frac{14}{q})^d$, by Lemma 21.

In this case, $\boldsymbol{y}$ is either $\boldsymbol{z}$, or is outside $B_{r^3\epsilon(\boldsymbol{\lambda}_v)}$. If $\|\boldsymbol{x}_u - \boldsymbol{x}_v\| \leq \frac{1}{2}r^3\epsilon$, then $\|\boldsymbol{y} - \boldsymbol{\lambda}_v\| \leq r^3\epsilon$, so we must have $\boldsymbol{y} = \boldsymbol{z}$, and v correctly outputs it. Otherwise, we may have $\boldsymbol{y} \notin B_{r^3\epsilon(\lambda_v)}$, but if so, v sends Far to u, which is also permitted.

□

We can then show the properties of the algorithm as a whole:

Lemma 23. *If $\|\boldsymbol{x}_u - \boldsymbol{x}_v\| \leq q^{q^d}\epsilon$, with probability at least $1 - \log\log(\frac{1}{\epsilon}\|\boldsymbol{x}_u - \boldsymbol{x}_v\|) \cdot O(q^{-d})$, Algorithm 5 provides machine v with an unbiased estimate $\boldsymbol{z}$ of $\boldsymbol{x}_u$, with $\|\boldsymbol{z} - \boldsymbol{x}_u\| \leq (q^2+7)\epsilon$, and uses $O\left(d \log(\frac{q}{\epsilon}\|\boldsymbol{x}_u - \boldsymbol{x}_v\|)\right)$ bits.*

Proof. After performing i iterations we have $r = q^{2^i}$. Therefore, after iteration $2\log\log(\|\boldsymbol{x}_u - \boldsymbol{x}_v\|/\epsilon)$ we have $\|\boldsymbol{x}_u - \boldsymbol{x}_v\| \leq \frac{1}{2}r^3\epsilon$. Then, so long as this iteration, and all previous iterations, are successful, v outputs an estimate $\boldsymbol{z}$ of $\boldsymbol{x}_u$ with $\|\boldsymbol{z} - \boldsymbol{x}_u\| \leq (q^2+7)\epsilon$, by Lemma 22.

Since we have $O(\log\log(\|\boldsymbol{x}_u - \boldsymbol{x}_v\|/\epsilon))$ iterations, and each fails with probability at most $\Omega(q)^{-d}$, our total failure probability is $\log\log(\|\boldsymbol{x}_u - \boldsymbol{x}_v\|/\epsilon) \cdot O(q^{-d})$ by a union bound over all iterations. The number of bits used per iteration is $O(d\log r)$; since $\log r$ doubles each iteration, the total is at most twice the bound for the final iteration. If this is the first iteration, the bound is $O(d\log q)$; otherwise it is $O(d\log(\frac{1}{\epsilon}\|\boldsymbol{x}_u - \boldsymbol{x}_v\|))$. So, the total number of bits used is $O(d\log(\frac{q}{\epsilon}\|\boldsymbol{x}_u - \boldsymbol{x}_v\|))$. □

C.1 Application to VarianceReduction

We now apply our error detection in order to improve our bounds on expected communication for performing VarianceReduction. The idea is that, since we only have a probabilistic bound on distance between inputs, some pairs of inputs may be significantly further away: we can now detect these cases using error detection, and increase the number of bits used for quantization accordingly.

We use the star-topology algorithm (Algorithm 3), and simply replace each pairwise encoding and decoding interaction with RobustAgreement(ϵ, q):

Algorithm 6 VarianceReduction

Nominate one machine v uniformly at random to be *leader*
All other machines u perform RobustAgreement(ϵ, q) to send $\boldsymbol{x}_u$ to v
Machine v averages all received estimates to obtain $\hat{\boldsymbol{\nabla}}$
Machine v performs RobustAgreement(ϵ, q) with each other machine to send $\hat{\boldsymbol{\nabla}}$, using the same choice of $\boldsymbol{z}$
All machines output the resulting estimate

We are now ready to prove our main result applying error detection to VarianceReduction, Theorem 4.

Proof of Theorem 4. We set $\mathfrak{q}$ to be $q + cn^{\frac{2}{d}}$ for some sufficiently large constant c, and run Algorithm 6 with $\mathfrak{q}$ and $\epsilon = \sigma/(\mathfrak{q}^4)$. We first consider the distances between encode and decode vectors in the first stage of messages, where all other machines u send their input to the leader v using RobustAgreement. In each case, $\mathbf{E}\left[\|\boldsymbol{x}_v - \boldsymbol{x}_u\|\right] \leq \mathbf{E}\left[\|\boldsymbol{x}_v - \boldsymbol{\nabla}\| + \|\boldsymbol{x}_u - \boldsymbol{\nabla}\|\right] \leq 2\sigma$. So long as we succeed (as specified by Lemma 23), node v then receives the estimate $\tilde{\boldsymbol{x}}$ of $\boldsymbol{x}_u$, with $\|\tilde{\boldsymbol{x}} - \boldsymbol{x}_u\| \leq (\mathfrak{q}^2+7)\epsilon < \sigma/\mathfrak{q}$ (since $\mathfrak{q}$ is chosen to be at least 3), and uses $O\left(d\log(\frac{\mathfrak{q}}{\epsilon}\|\boldsymbol{x}_u - \boldsymbol{x}_v\|)\right)$ bits. We have $\mathbf{E}\left[d\log(\frac{\mathfrak{q}}{\epsilon}\|\boldsymbol{x}_u - \boldsymbol{x}_v\|)\right] = O(d\log\mathfrak{q}) = O(d\log q + \log n)$.

The leader node v now averages these estimates to obtain $\hat{\boldsymbol{\nabla}}$. Each of the received estimates is an independent unbiased estimator of $\boldsymbol{\nabla}$ with variance $O(\sigma^2)$ and therefore $\hat{\boldsymbol{\nabla}}$ is an unbiased estimator of $\boldsymbol{\nabla}$ with variance $O(\sigma^2/n)$.

In the second stage of messages, v sends $\hat{\boldsymbol{\nabla}}$ to all other nodes using RobustAgreement, taking the same choice of $\boldsymbol{z}$ in each, in order to ensure that machines all have the same output. z is an unbiased estimate of $\boldsymbol{\nabla}$ with variance $O(\frac{\sigma^2}{n} + \frac{\sigma^2}{\mathfrak{q}}) = O(\frac{\sigma^2}{\min n,q})$. Again, for each such node u we have $\mathbf{E}\left[\|\boldsymbol{x}_u - \hat{\boldsymbol{\nabla}}\|\right] = O(\sigma)$, so we use $O(d\log q + \log n)$ bits in expectation. Therefore each node uses $O(d\log q + \log n)$ total bits in expectation, except the leader v which uses $O(nd\log q + n\log n)$. Since v is chosen uniformly at random, we have an $O(d\log q + \log n)$-bit bound on expected communication for each machine.

In all applications of RobustAgreement we perform, denoting the encode and decode vectors $\boldsymbol{a}$, $\boldsymbol{b}$ respectively (i.e., in the first stage $\boldsymbol{a} = \boldsymbol{x}_u$ and $\boldsymbol{b} = \boldsymbol{x}_v$ for each u, and in the second $\boldsymbol{a} = \hat{\boldsymbol{\nabla}}$ and $\boldsymbol{b} = \boldsymbol{x}_u$), we have $\mathbf{E}\left[\|\boldsymbol{a} - \boldsymbol{b}\|^2\right] \leq (2\sigma)^2$ and so $\mathbf{Pr}\left[\|\boldsymbol{a} - \boldsymbol{b}\| \leq 4n\sigma\right] \leq \frac{1}{4n^2}$.

Since we perform RobustAgreement $2n$ times, with probability at least $1-\frac{1}{2n}$, $\|\boldsymbol{a}-\boldsymbol{b}\| \leq 4n\sigma$ in all cases by a union bound. So, with high probability we always satisfy the condition $\|\boldsymbol{a}-\boldsymbol{b}\| \leq \mathfrak{q}^{\mathfrak{q}^d}\epsilon = \mathfrak{q}^{\mathfrak{q}^d-4}\sigma$, and have failure probability at most $\log\log(\frac{1}{\epsilon}\|\boldsymbol{a}-\boldsymbol{b}\|)\cdot O(\mathfrak{q}^{-d}) = O(\mathfrak{q}^{-d}\log\log(n\mathfrak{q}))$ for each application of RobustAgreement. So, the probability that all applications succeed is at least $(1-\frac{1}{2n})(1-O(\mathfrak{q}^{-d}\log\log(n\mathfrak{q}))) \geq 1-\frac{1}{n}$.

Then, all machines correctly receive $\boldsymbol{z}$, an estimate of $\boldsymbol{\nabla}$ with variance at most $O(\frac{\sigma^2}{\min n,q})$. □

D ℓ_2 Guarantees for the Cubic Lattice

One key question when applying our scheme in practice is **the choice of lattice**. Ideally this would depend on the norm under which the results will be evaluated. However, asymptotically optimal lattices for ℓ_1 and ℓ_2 norms can be computationally expensive to generate and use in practice. For this reason, in our practical implementations we will employ the standard **cubic lattice**: this gives us computationally-efficient ($\tilde{O}(d)$-time) quantization algorithms, is optimal under ℓ_∞ norm, and, as we will see, performs surprisingly well even evaluated under ℓ_1 or ℓ_2 norm. We will investigate more complex lattices, tailored to ℓ_1 or ℓ_2 norm, in future work. There are two possible approaches to doing so: one would be to apply *general* lattice algorithms (e.g. Laarhoven (2016)) in applications where d is still fairly low. This is often the case in, for example, gradient descent for neural network training, where coordinates are already divided into fairly small *buckets*. The second possible approach would be to find *specific* lattices which admit more efficient algorithms, and also have a good r_c/r_p ratio under ℓ_1 or ℓ_2 norm.

Using the cubic lattice, though, need not sacrifice too much by way of theoretical guarantees under ℓ_2-norm, since, as noted in Suresh et al. (2017), a random rotation using the Walsh-Hadamard transform can ensure good bounds on the ratio between ℓ_2 norm and ℓ_∞ norm of vectors.

Let H be the $d \times d$ normalized Hadamard matrix $H_{i,j} = d^{-1/2}(-1)^{\langle i-1,j-1\rangle}$, where $\langle i,j\rangle$ is the dot-product of the $\log_2 d$-dimension $\{0,1\}$-valued vectors given by i, j expressed in binary (we must assume here that d is a power of two, but this does not affect asymptotic results). We use the following well-known properties of H (see e.g. Horadam (2007)):

- H is orthonormal, and so preserves ℓ_2-norm distances;
- $H^{-1} = H$.

Let D be a $d \times d$ diagonal matrix where each $D_{i,i}$ is drawn uniformly at random from $\{-1,1\}$. H is a fixed, known matrix, and D costs d bits to communicate, so we can assume that both matrices are known to all machines at a cost of only $O(d)$ bits of communication per machine.

Before applying our MeanEstimation or VarianceReduction algorithms, we apply the transformation HD to all inputs - we then invert the transform (i.e. apply $(HD)^{-1} = D^{-1}H$) before final output. As shown in Ailon & Chazelle (2009), both the forward and inverse transform require only $O(d\log d)$ computation.

Lemma 24. *For any set S of n^2 vectors in $\mathbb{R}_d$, with probability at least $1-2d^{-1}$, all vectors $\boldsymbol{x} \in S$ have*

$$\|HD\boldsymbol{x}\|_\infty = O(d^{-1/2}\|\boldsymbol{x}\|_2\sqrt{\log nd})$$

Proof. We follow a similar argument to Ailon & Chazelle (2009). Fix some vector $\boldsymbol{x} \in S$ and some coordinate $j \in [d]$. Notice that $(HD\boldsymbol{x})_j = \sum_{i=1}^d \alpha_i \boldsymbol{x}_i$, where each $\alpha_i = \pm d^{-1/2}$ is chosen independently and uniformly. By a Chernoff-type bound (c.f. Ailon & Chazelle (2009)), we therefore obtain $\mathbf{Pr}\left[|(HD\boldsymbol{x})_j| \geq s\|\boldsymbol{x}\|_2\right] \leq 2e^{-s^2 d/2}$. Plugging in $s = 2\sqrt{\frac{\ln nd}{d}}$, we get:

$$\mathbf{Pr}\left[|(HD\boldsymbol{x})_j| \geq 2\sqrt{\frac{\ln nd}{d}}\|\boldsymbol{x}\|_2\right] \leq 2e^{-2\ln nd} = 2(nd)^{-2} \ .$$

We then take a union bound over all $\boldsymbol{x} \in S$ and $j \in [d]$ to find that with probability at least $1 - 2d^{-1}$, all $\boldsymbol{x} \in S$ have $\|HD\boldsymbol{x}\|_\infty = O(d^{-1/2}\|\boldsymbol{x}\|_2\sqrt{\log nd})$. □

We can now obtain analogue of Theorem 2 using the cubic lattice:

Theorem 25. *For any $q = \Omega(1)$,* MeanEstimation *can be performed using the cubic lattice with each machine using $O(d \log q)$ communication bits in total, with $O(\frac{y^2 \log nd}{q})$ output variance under ℓ_2 norm, succeeding with probability at least $1 - 2d^{-1}$.*

Proof. We apply the random rotation HD as in Lemma 24, and then proceed as usual using Algorithm 4, applying the inverse rotation to the final output. Let y_2 denote the usual maximum input distance parameter under ℓ_2-norm. We first show that after applying the random rotation we have $y_\infty = O(d^{-1/2} y_2 \sqrt{\log nd})$:

There are n machines v with input vectors $\boldsymbol{x}_v \in \mathbb{R}^d$. We form a set $S = \{\boldsymbol{x}_v - \boldsymbol{x}_u$ for all machines $u, v\}$. This set then has at most n^2 elements. Applying the rotation HD to all input vectors implicitly applies it to this set S of *differences*, since HD is a linear transformation. Therefore, by Lemma 24, we have

$$\max_{\boldsymbol{z} \in S} \|HD\boldsymbol{z}\|_\infty = O(d^{-1/2} \max_{\boldsymbol{z} \in S} \|\boldsymbol{z}\|_2 \sqrt{\log nd}) = O(d^{-1/2} y_2 \sqrt{\log nd}),$$

so we can set $y_\infty = O(d^{-1/2} y_2 \sqrt{\log nd})$ and satisfy that $\|HD(\boldsymbol{x}_v - \boldsymbol{x}_u)\|_\infty \leq y_\infty$ for all machines u, v, with probability at least $1 - 2d^{-1}$.

Now we apply Algorithm 4 using the cubic lattice. Since the cubic lattice is optimal under ℓ_∞ norm, we obtain an output variance (under ℓ_∞ norm) of $O(\frac{y_\infty^2}{q}) = O(\frac{d^{-1} y_2^2 \log nd}{q})$ when using $O(d \log q)$ bits per machine, by Theorem 2 (but our output $HD\boldsymbol{EST}$ is currently an unbiased estimator of the rotated mean $HD\boldsymbol{\mu}$ rather than $\boldsymbol{\mu}$ itself). Since, for any vector $\boldsymbol{z} \in \mathbb{R}^d$, $\|\boldsymbol{z}\|_2^2 \leq d\|\boldsymbol{z}\|_\infty^2$, when applied with $\boldsymbol{z} = HD(\boldsymbol{EST} - \boldsymbol{\mu})$ we get an output variance under ℓ_2-norm of $O(\frac{y_2^2 \log nd}{q})$

When applying the final inverse rotation to the output vector $HD\boldsymbol{EST}$, we also implicitly imply it to the error vector $HD(\boldsymbol{EST} - \boldsymbol{\mu})$. This inverse rotation preserves distances under ℓ_2-norm, so we still have $O(\frac{y_2^2 \log nd}{q})$ ℓ_2-norm variance, now of an unbiased estimator $\boldsymbol{EST}$ of the mean $\boldsymbol{\mu}$ of our original inputs (unbiasedness is preserved by linearity of expectation, since HD and $(HD)^{-1}$ are linear transformations). □

By applying the same analysis to Theorems 3 and 4, we obtain the VarianceReduction results of Theorem 5.

Proof of Theorem 5. We perform the same procedure as in Theorem 25, applying the random rotation HD to obtain a bound on ℓ_∞-norm, applying Algorithms 3 and 4 respectively, using the $\ell_i nfty$-norm-optimal cubic lattice, and then applying the inverse rotation before output to obtain an ℓ_2-norm output variance bound at only a $O(\log nd)$ factor higher than optimal.

Specifically, we can obtain unbiased estimates of input mean $\boldsymbol{\mu}$ with variance $O(\frac{\sigma^2 \log nd}{n^2})$ using strictly $O(d \log n)$ bits (with sufficiently high constant within the asymptotic notation), and $O(\frac{\sigma^2 \log nd}{q})$ using $O(d \log q + \log n)$ bits in expectation. Since $\boldsymbol{\mu}$ is itself an $O(\frac{\sigma^2}{n})$-variance unbiased estimator of our true vector $\boldsymbol{\nabla}$, we obtain final output variances of $O(\frac{\sigma^2 \log nd}{n^2} + \frac{\sigma^2}{n}) = O(\frac{\sigma^2 \log d}{n})$ and $O(\frac{\sigma^2 \log nd}{q} + \frac{\sigma^2}{n})$ respectively. □

E Lower Bounds

We prove matching lower bounds for MeanEstimation and VarianceReduction, using an argument based on bounding the volume of space for which a node can output a good

estimate, if it receives a limited number of communication bits. Our lower bounds will be against algorithms with *shared randomness*: that is, we assume access to a common random string s, drawn from some distribution, for all machines.

We begin with the simpler MeanEstimation bound:

Theorem 26. *For any* MeanEstimation *algorithm in which any machine receives at most b bits,*

$$\mathbf{E}\left[\|\boldsymbol{EST}-\mu\|^2\right] = \Omega(y^2 2^{-\frac{2b}{d}}) \ .$$

Proof. We construct a hard input for mean estimation as follows: fix a machine v, and arbitrarily fix its input vector $\boldsymbol{x}_v$. Machine v's output $\boldsymbol{EST}$ is dependent only on $\boldsymbol{x}_v$, the random string s, and the string of bits B_v that v receives (from any other machines) during the course of the algorithm. We denote by b_v the number of such bits. If $b_v < b$ for some b, then v has $\sum_{i=0}^{b-1} 2^i < 2^b$ possible strings B_v, and therefore fewer than 2^b possible output distributions (over choice of s).

For each B_v we denote by OUT_{B_v} the set of points $\boldsymbol{z} \in \mathbb{R}^d$ for which $\boldsymbol{P}_s[\|\boldsymbol{EST}-\boldsymbol{z}\| \leq \delta] > \frac{1}{2}$, when B_v is the string received. For any B_v, OUT_{B_v} has volume at most $Vol(B_{2\delta})$, since the δ-balls of any two points in OUT_{B_v} must intersect (as the probabilities of $\boldsymbol{EST}$ falling within the balls sum to more than 1). We further denote OUT_v to be the union of these sets over all B_v. Then:

$$Vol(OUT_v) \leq \sum_{B_v} Vol(OUT_{B_v}) < 2^b \cdot Vol(B_{2\delta}) \leq 2^b \left(\frac{4\delta}{y}\right)^d Vol(B_{\frac{y}{2}}) \ .$$

We choose $\delta = \frac{y}{4} 2^{-\frac{b}{d}}$, and see that:

$$Vol(OUT_v) < 2^b \left(\frac{4 \cdot \frac{y}{4} 2^{-\frac{b}{d}}}{y}\right)^d Vol(B_{\frac{y}{2}}) = 2^b \cdot 2^{-b} \cdot Vol(B_{\frac{y}{2}}) = Vol(B_{\frac{y}{2}}) \ .$$

Therefore there is some point $\boldsymbol{z}$ in $B_{\frac{y}{2}}(\boldsymbol{x}_v) \setminus Vol(OUT_v)$. We choose μ to be such a point $\boldsymbol{z}$, by setting one other machine's input to $2\boldsymbol{z} - \boldsymbol{x}_v$, and any others to $\boldsymbol{z}$ (note that this satisfies the condition that all inputs are within distance y). Then, regardless of B_v, $\boldsymbol{P}_s[\|\boldsymbol{EST}-\mu\| \leq \delta] \leq \frac{1}{2}$, and so $\mathbf{E}\left[\|\boldsymbol{EST}-\mu\|^2\right] \geq \frac{1}{2} \cdot \delta^2 = \Omega(y^2 2^{-\frac{2b}{d}})$. □

This lower bound is for algorithms with a fixed communication cost, but we show that it can easily be extended to apply to *expected* communication cost, proving Theorem 6:

Proof of Theorem 6. By Markov's inequality, a machine v which receives at most b bits in expectation receives at most $1.5b$ bits with probability at least $\frac{1}{3}$. Then,

$$\mathbf{E}\left[\|\boldsymbol{EST}-\mu\|^2\right] \geq \frac{1}{3}\mathbf{E}\left[\|\boldsymbol{EST}-\mu\|^2 \mid v \text{ receives at most } 1.5b \text{ bits}\right] = \Omega(y^2 2^{-\frac{3b}{d}}) \ .$$

□

The bounds for VarianceReduction are somewhat more complex since we must define a hard *input distribution* in which the input vectors are independent estimates of the true vector $\boldsymbol{\nabla}$ and have variance at most σ^2.

We first show a bound on the amount of communication bits that *all* machines must use (Theorem 8, before proceeding to Theorem 7 which shows that some machines must use more bits.

Proof of Theorem 8. We define an input distribution I as follows: we choose $\boldsymbol{\nabla}$ uniformly at random from $B_{d^2\sigma}(\mathbf{0})$. We then independently choose each machine v's input $\boldsymbol{x}_v$ uniformly at random from $B_\sigma(\boldsymbol{\nabla})$.

For any machine v, we define event $A_v = \{\|\boldsymbol{x}_v\| \leq (d^2-1)\sigma\}$.

$$\begin{aligned}\mathbf{Pr}\left[A_v\right] &\geq \mathbf{Pr}\left[\|\boldsymbol{\nabla}\| \leq (d^2-2)\sigma\right] \\ &\geq \frac{Vol(B_{(d^2-2)\sigma})}{Vol(B_{d^2\sigma})} \\ &\geq \left(\frac{(d^2-2)\sigma}{d^2\sigma}\right)^d \\ &\geq \left(1-\frac{2}{d^2}\right)^d \\ &\geq 1-\frac{4}{d} \ .\end{aligned}$$

Now, conditioning on A_v, $\boldsymbol{\nabla}$ is distributed uniformly in $B_\sigma(\boldsymbol{x}_v)$. We next show that in this case, v will, with high probability, not be able to closely estimate $\boldsymbol{\nabla}$.

Using the same argument and definitions as in proof of Theorem 26, we have:

$$Vol(OUT_v) \leq \sum_{B_v} Vol(OUT_{B_v}) < 2^b \cdot Vol(B_{2\delta}) \leq 2^b \left(\frac{2\delta}{\sigma}\right)^d Vol(B_\sigma) \ .$$

We choose $\delta = \frac{\sigma}{4}2^{-\frac{b}{d}}$, and see that $Vol(OUT_v) < 2^{-d}Vol(B_\sigma)$. Therefore, $\boldsymbol{P}_I[\boldsymbol{\nabla} \in OUT_v \mid A_v] \leq 2^{-d}$, and so $P_I[\boldsymbol{\nabla} \notin OUT_v] \leq (1-\frac{4}{d})(1-2^{-d}) \leq \frac{1}{2}$. When $\boldsymbol{\nabla} \notin OUT_v$, with probability at least $\frac{1}{2}$, $\boldsymbol{P}_s[\|\boldsymbol{EST}-\mu\| \leq \delta] \leq \frac{1}{2}$.

So, we have that:

$$\mathbf{Pr}\left[\|\boldsymbol{EST}-\mu\| > \delta\right] \geq \frac{1}{2}\mathbf{Pr}\left[\boldsymbol{\nabla} \notin OUT_v\right] > \frac{1}{2}(1-\frac{4}{d})\boldsymbol{P}_I[\bigcup_{v\in M} A_v] > 0.2 \ .$$

Then, $\mathbf{E}\left[\|\boldsymbol{EST}-\boldsymbol{\nabla}\|^2\right] > 0.2\delta^2 = \Omega(\sigma^2 n 2^{-\frac{2b}{d}})$. □

Theorem 7 is proven similarly, but using a different input distribution that causes some inputs to be further from $\boldsymbol{\nabla}$.

Proof of Theorem 7. We define an input distribution I as follows: we choose $\boldsymbol{\nabla}$ uniformly at random from $B_{d^2n\sigma}(\mathbf{0})$. We then independently choose each machine v's input $\boldsymbol{x}_v$ as follows: with probability $\frac{1}{n}$, $\boldsymbol{x}_v$ is chosen uniformly from $B_{\sigma\sqrt{n}}(\boldsymbol{\nabla})$, and otherwise $\boldsymbol{x}_v = \boldsymbol{\nabla}$.

For any machine v, we are interested in the event A_v that

- $\boldsymbol{x}_v$ is chosen from $B_{\sigma\sqrt{n}}(\boldsymbol{\nabla})$, and
- $\|\boldsymbol{x}_v\| \leq d^2 n\sigma - \sigma\sqrt{n}$;

If we condition on $\boldsymbol{\nabla} \in B_{d^2n\sigma-2\sigma\sqrt{n}}(\mathbf{0})$, the second criterion will be true for all v, and so the events A_v will occur independently with probability $\frac{1}{n}$. We use this to show that the probability that A_v occurs for *some* v is greater than $\frac{1}{2}$:

$$\begin{aligned}
\boldsymbol{P}_I[\bigcup_{v\in M} A_v] &\geq \boldsymbol{P}_I[\{\boldsymbol{\nabla} \in B_{d^2n\sigma-2\sigma\sqrt{n}}(\mathbf{0})\} \cap \bigcup_{v\in M} A_v] \\
&\geq \boldsymbol{P}_I[\{\boldsymbol{\nabla} \in B_{d^2n\sigma-2\sigma\sqrt{n}}(\mathbf{0})\}] \cdot \boldsymbol{P}_I[\bigcup_{v\in M} A_v \mid \boldsymbol{\nabla} \in B_{d^2n\sigma-2\sigma\sqrt{n}}(\mathbf{0})] \\
&= \frac{Vol(B_{d^2n\sigma-2\sigma\sqrt{n}})}{Vol(B_{d^2n\sigma})} \cdot \left(1 - (\frac{n-1}{n})^n\right) \\
&\geq \left(\frac{d^2n\sigma - 2\sigma\sqrt{n}}{d^2n\sigma}\right)^d \cdot (1-e^{-1}) \\
&> 0.6\left(1-\frac{1}{d^2}\right)^d \geq 0.6\left(1-\frac{2}{d}\right) > 0.5 \ .
\end{aligned}$$

Now, conditioning on A_v, $\boldsymbol{\nabla}$ is distributed uniformly in $B_{\sigma\sqrt{n}}(\boldsymbol{x}_v)$. We next show that in this case, v will, with high probability, not be able to closely estimate $\boldsymbol{\nabla}$.

Again we have:

$$Vol(OUT_v) \leq \sum_{B_v} Vol(OUT_{B_v}) < 2^b \cdot Vol(B_{2\delta}) \leq 2^b \left(\frac{2\delta}{\sigma\sqrt{n}}\right)^d Vol(B_{\sigma\sqrt{n}}) \ .$$

We choose $\delta = \frac{\sigma\sqrt{n}}{4}2^{-\frac{b}{d}}$, and see that $Vol(OUT_v) < 2^{-d}Vol(B_{\sigma\sqrt{n}})$. Therefore, $\boldsymbol{P}_I[\boldsymbol{\nabla} \in OUT_v \mid A] \leq 2^{-d}$, and so $P_I[\boldsymbol{\nabla} \in OUT_v] \leq \frac{1}{2}$. When $\boldsymbol{\nabla} \notin OUT_v$, with probability at least $\frac{1}{2}$, $\boldsymbol{P}_s[\|\boldsymbol{EST} - \mu\| \leq \delta] \leq \frac{1}{2}$.

So, we have that:

$$\mathbf{Pr}\left[\|\boldsymbol{EST} - \mu\| > \delta\right] \geq \frac{1}{2}\mathbf{Pr}\left[\boldsymbol{\nabla} \notin \bigcap_{v\in M} OUT_v\right] > \frac{1}{2}(1-2^{-d})\boldsymbol{P}_I[\bigcup_{v\in M} A_v] > 0.2 \ .$$

Then, $\mathbf{E}\left[\|\boldsymbol{EST} - \boldsymbol{\nabla}\|^2\right] > 0.2\delta^2 = \Omega(\sigma^2 n 2^{-\frac{2b}{d}})$. Again, by Markov's inequality, we can obtain that if machine v receives at most b bits *in expectation*, we have $\mathbf{E}\left[\|\boldsymbol{EST} - \boldsymbol{\nabla}\|^2\right] = \Omega(\sigma^2 n 2^{-\frac{3b}{d}})$ □

Theorems 26 and 8 imply that to obtain output variances of $O(\frac{y^2}{q})$ and $O(\frac{\sigma^2}{q})$ for MeanEstimation and VarianceReduction respectively, we require $\Omega(d\log q)$ communication bits per machine, thereby matching the upper bounds of Theorems 16 and 4 (the latter in expectation). Theorem 7 further implies that to achieve any variance reduction *at all* (i.e. for output variance to be lower than input variance σ^2), at least one machine must receive $\Omega(d\log n)$ bits, matching the absolute upper bound on bits required to achieve the optimal output variance $O(\frac{\sigma^2}{n})$ by Theorem 3.

F Sublinear Communication

In this section we show an extension to the quantization scheme (and thereby also Algorithm 3) allowing us to use a *sublinear* (in d) number of bits in expectation, providing a variance trade-off for the *full range* of communication for both MeanEstimation and VarianceReduction. Here our methods apply only to ℓ_2-norm.

Theorem 27. *For any $q > 0$,* MeanEstimation *can be performed using $O(d\log(1+q))$ communication bits per machine in expectation, with $O(\frac{y^2}{q^2})$ output variance under ℓ_2 norm.*

This result matches Theorem 16 for $\Omega(d)$ communication, but also provides bounds for $o(d)$ communication (by setting $q = o(1)$). We can further show good concentration on the amount of communication bits required, and extend to VarianceReduction as before.

Our sublinear-communication quantization method will involve mapping points to their *closest* lattice point (rather than randomly to a set of near by points who form a convex hull around them). We will therefore be concerned with the (open) Voronoi regions of lattice points, since these are the regions that will be quantized to the same point:

Definition 28. *For any lattice Λ, the Voronoi region of a lattice point $\boldsymbol{\lambda}$ is the set of all $\boldsymbol{x} \in \mathbb{R}^d$ to whom $\boldsymbol{\lambda}$ is closer than any other lattice point, i.e., $\boldsymbol{Vor}(\boldsymbol{\lambda}) := \{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x} - \boldsymbol{\lambda}\|_2 < \|\boldsymbol{x} - \boldsymbol{\lambda}'\|_2, \forall \boldsymbol{\lambda}' \in \Lambda \setminus \{\boldsymbol{\lambda}\}\}$.*

Voronoi regions of a lattice are known to have several useful properties, of which we will use the following:

Lemma 29. *The Voronoi regions of a lattice are:*

1. *open convex polytopes,*
2. *symmetric, i.e., $\boldsymbol{\lambda} + \boldsymbol{x} \in \boldsymbol{Vor}(\boldsymbol{\lambda}) \iff \boldsymbol{\lambda} - \boldsymbol{x} \in \boldsymbol{Vor}(\boldsymbol{\lambda})$, and*
3. *identical, i.e., $\boldsymbol{\lambda} + \boldsymbol{x} \in \boldsymbol{Vor}(\boldsymbol{\lambda}) \iff \boldsymbol{\lambda}' + \boldsymbol{x} \in \boldsymbol{Vor}(\boldsymbol{\lambda}')$ for any $\boldsymbol{\lambda}' \in \Lambda$.*

Proof. Property 1 follows since $\boldsymbol{Vor}(\boldsymbol{\lambda})$ is the intersection of half-spaces formed by the set of points which are closer to $\boldsymbol{\lambda}$ than $\boldsymbol{\lambda}'$, for any $\boldsymbol{\lambda}' \in \Lambda \setminus \{\boldsymbol{\lambda}\}$.

Property 2 follows if there exists $\boldsymbol{\lambda}' \in \Lambda$ such that $\|\boldsymbol{\lambda} + \boldsymbol{x} - \boldsymbol{\lambda}'\|_2 \leq \|\boldsymbol{x}\|_2$, then $\hat{\boldsymbol{\lambda}} = 2\boldsymbol{\lambda} - \boldsymbol{\lambda}'$ is also a lattice point in Λ, and

$$\|\boldsymbol{\lambda} - \boldsymbol{x} - \hat{\boldsymbol{\lambda}}\|_2 = \|\boldsymbol{\lambda} - \boldsymbol{x} - (2\boldsymbol{\lambda} + \boldsymbol{\lambda}')\|_2 = \| - (\boldsymbol{\lambda} + \boldsymbol{x} - \boldsymbol{\lambda}')\|_2 \leq \|\boldsymbol{x}\|_2 \ .$$

Therefore $\boldsymbol{\lambda} + \boldsymbol{x} \notin \boldsymbol{Vor}(\boldsymbol{\lambda}) \implies \boldsymbol{\lambda} - \boldsymbol{x} \notin \boldsymbol{Vor}(\boldsymbol{\lambda})$, which by contradiction proves the property.

Property 3 follows since the relative positions of all other lattice points are identical with respect to $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$. □

In particular, Property 3 implies that all Voronoi regions of a lattice have the same volume, which we denote $\boldsymbol{V}$. We note that, within any finite ball, the set of points which do *not* fall in a Voronoi region (i.e. have multiple equidistant closest lattice points) has measure 0 (under the standard Lebesgue measure of $\mathbb{R}^d$), and therefore do not affect the probabilities of events under this measure.

We now wish to bound the number of Voronoi regions which are close to any point in $\mathbb{R}^d$, in expectation; these will be the regions of lattice points which are sufficiently close to cause decoding errors if they are encoded with the same bit-string. For this we introduce the concept of an *expanded* Voronoi region:

Definition 30. *The* expanded Voronoi region $\boldsymbol{Vor}^+(\boldsymbol{\lambda})$ *of a lattice point $\boldsymbol{\lambda}$ is the set $\boldsymbol{Vor}^+(\boldsymbol{\lambda})$ of points within distance $2q\epsilon$ of $\boldsymbol{Vor}(\boldsymbol{\lambda})$.*

We bound the volume of such sets:

Lemma 31. *For any $\boldsymbol{\lambda}$ in Λ_ϵ, the expanded Voronoi region $\boldsymbol{Vor}^+(\boldsymbol{\lambda})$ has volume at most $(1 + 2q)^d \boldsymbol{V}$.*

Proof. Let $\widehat{\boldsymbol{Vor}}(\boldsymbol{\lambda})$ be $\boldsymbol{Vor}(\boldsymbol{\lambda})$ dilated by a factor of $1 + 2q$ around $\boldsymbol{\lambda}$; that is:

$$\widehat{\boldsymbol{Vor}}(\boldsymbol{\lambda}) := \{\lambda + (1 + 2q)\boldsymbol{x} : \lambda + \boldsymbol{x} \in \boldsymbol{Vor}(\boldsymbol{\lambda})\} \ .$$

Clearly the volume of $\widehat{\boldsymbol{Vor}}(\boldsymbol{\lambda})$ is $(1 + 2q)^d \boldsymbol{V}$; we now show that $\boldsymbol{Vor}^+(\boldsymbol{\lambda}) \subseteq \widehat{\boldsymbol{Vor}}(\boldsymbol{\lambda})$. $\boldsymbol{Vor}(\boldsymbol{\lambda})$ is the intersection of open half-spaces bounded by the hyperplanes of points equidistant from $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$, for any $\boldsymbol{\lambda}' \in \Lambda_\epsilon \setminus \{\boldsymbol{\lambda}\}$. Dilation by a factor of $1 + 2q$ therefore translates each of these hyperplanes by $2q \cdot \frac{\boldsymbol{\lambda}' - \boldsymbol{\lambda}}{2}$ respectively. Since ϵ is the packing radius of Λ_ϵ, $\|\frac{\boldsymbol{\lambda}' - \boldsymbol{\lambda}}{2}\|_2 \geq \epsilon$, and therefore each such hyperplane is translated a distance of at least $2q\epsilon$ away from $\boldsymbol{\lambda}$. So, $\widehat{\boldsymbol{Vor}}(\boldsymbol{\lambda})$ contains all points within distance $2q\epsilon$ of $\boldsymbol{Vor}(\boldsymbol{\lambda})$, completing the proof. □

We can now present the encoding algorithm (Algorithm 7):

Algorithm 7 SublinearEncode, to compute $Q'_{\epsilon,q}(\boldsymbol{x})$

$i \leftarrow 0$
loop
  Let $\boldsymbol{\theta}$ be a uniformly random vector in $\boldsymbol{Vor}(\mathbf{0})$.
  Let $\boldsymbol{z}$ be the closest lattice point in Λ_ϵ to $\boldsymbol{x}+\boldsymbol{\theta}$.
  Let $c' \sim C$ be a random coloring $\Lambda_\epsilon \to [1+2q]^{3d}$
  if there is no $\boldsymbol{z}' \in \Lambda_\epsilon$ with $\boldsymbol{x}+\boldsymbol{\theta} \in \boldsymbol{Vor}^+(\boldsymbol{z}')$ and $c'(\boldsymbol{z}) = c'(\boldsymbol{z}')$ **then**
    send $c'(\boldsymbol{z})$ and i, terminate
  end if
  $i \leftarrow i+1$
end loop

The algorithm works as follows: we first apply a random offset vector $\boldsymbol{\theta}$, with the purpose of making the quantization unbiased. We then round to the closest lattice point. To convert this lattice point to a finite bit-string, we then apply a random coloring (from a distribution we will specify shortly); we show a lower bound on the probability that the color given to $\boldsymbol{z}$ is unique among those lattice points for whom $\boldsymbol{x}+\boldsymbol{\theta}$ falls in the expanded Voronoi region. In this case, we send the color and the number i of the current iteration; if not, we repeat the whole process with fresh shared randomness.

We first show a bound on the expected number of lattice points for which $\boldsymbol{x}+\boldsymbol{\theta}$ falls in the expanded Voronoi region:

Lemma 32. *The number $V_{exp}(\boldsymbol{x}+\boldsymbol{\theta})$ of expanded Voronoi regions containing $\boldsymbol{x}+\boldsymbol{\theta}$ is at most $(1+2q)^{2d}$ with probability at least $1-(1+2q)^{-d}$.*

Proof. Since $\boldsymbol{\theta}$ is uniformly distributed in $\boldsymbol{Vor}(\mathbf{0})$, $\boldsymbol{x}+\boldsymbol{\theta}-\boldsymbol{z}$ is uniformly distributed in $\boldsymbol{Vor}(\mathbf{0})$ (due to Voronoi regions being open, the probability distribution function differs from uniformity, but only on a set of 0 measure).

Then, since Voronoi regions are identical (including in their intersections with expanded Voronoi regions), and $\boldsymbol{x}+\boldsymbol{\theta}$ falls in exactly 1 Voronoi region in expectation, it falls in $\frac{Vol(\boldsymbol{Vor}^+(\mathbf{0}))}{Vol(\boldsymbol{Vor}(\mathbf{0}))} \leq (1+2q)^d$ expanded Voronoi regions in expectation, by Lemma 31. Therefore, by Markov's inequality, the probability of falling within at least $(1+2q)^{2d}$ expanded Voronoi regions is at most $(1+2q)^{-d}$. □

We will need one further property of expanded Voronoi regions in order to show unbiasedness: that the number of expanded Voronoi regions containing a point is symmetric around any lattice point:

Lemma 33. *For any $\boldsymbol{x} \in \mathbb{R}^d$, $\boldsymbol{\lambda} \in \Lambda_\epsilon$, the points $\boldsymbol{\lambda}+\boldsymbol{x}$ and $\boldsymbol{\lambda}-\boldsymbol{x}$ are in the same number of expanded Voronoi regions $\boldsymbol{Vor}^+(\boldsymbol{\lambda}')$.*

Proof. This follow from the symmetry of Λ_ϵ with respect to λ: if $\boldsymbol{\lambda}+\boldsymbol{x} \in \boldsymbol{Vor}^+(\boldsymbol{\lambda}')$, then $2\boldsymbol{\lambda}-\boldsymbol{\lambda}'$ is also a lattice point, and $\boldsymbol{\lambda}-\boldsymbol{x} \in \boldsymbol{Vor}^+(2\boldsymbol{\lambda}-\boldsymbol{\lambda}')$. □

We next define our distribution of colorings: we will first apply the deterministic coloring c_{3+2q} as described in Section A. Then, by Lemma 12, any two points of the same color are of distance at least $2(3+2q)\epsilon$ apart. Since $\boldsymbol{Vor}(\boldsymbol{\lambda}) \subset B_{r_c}(\boldsymbol{\lambda}) \subseteq B_{3\epsilon}(\boldsymbol{\lambda})$, we have $\boldsymbol{Vor}^+(\boldsymbol{\lambda}) \subset B_{(3+2q)\epsilon}(\boldsymbol{\lambda})$, and therefore there is no intersection between the expanded Voronoi regions of any points of the same color under c_{3+2q}. The purpose of this initial coloring is to limit the amount of randomness required for second random stage of the coloring process.

We then choose uniformly random colorings $\hat{c} : [3+2q]^d \to [1+2q]^{3d}$; the final coloring is then $\hat{c} \circ c_{3+2q}$, and we define C to be the distribution of colorings generated in this way.

We will call such a coloring c' *successful* for a point $\boldsymbol{x}+\boldsymbol{\theta}$ if it meets the condition described in the algorithm, i.e., if there is no $\boldsymbol{z}' \in \Lambda_\epsilon$ with $\boldsymbol{x}+\boldsymbol{\theta} \in \boldsymbol{Vor}^+(\boldsymbol{z}')$ and $c'(\boldsymbol{z}) = c'(\boldsymbol{z}')$.

Lemma 34. *Over choice of $\boldsymbol{\theta}$ and c', $\mathbf{Pr}\left[c' \text{ is successful}\right] \geq 1 - 2(1+2q)^{-d}$, and*

$$\mathbf{E}\left[\boldsymbol{z} - \boldsymbol{\theta} | c' \text{ is successful}\right] = \boldsymbol{x} \ .$$

Proof. The probability of c' being successful is dependent entirely on the number $V_{exp}(\boldsymbol{x}+\boldsymbol{\theta})$ of expanded Voronoi regions containing $\boldsymbol{x}+\boldsymbol{\theta}$. By Lemma 32, this number is at most $(1+2q)^{2d}$ with probability at least $1-(1+2q)^{-d}$. In this case, the probability that $\boldsymbol{z}$ does not receive a unique color under c' is at most $(1+2q)^{2d} \cdot (1+2q)^{-3d} = (1+2q)^{2d} \cdot (1+2q)^{-d}$ by a union bound. Therefore, the total probability of c' not being successful is at most $2(1+2q)^{-d}$.

To show unbiasedness (that $\mathbf{E}\left[\boldsymbol{z} - \boldsymbol{\theta} | c' \text{ is successful}\right] = \boldsymbol{x}$), we note that $\boldsymbol{x} + \boldsymbol{\theta} - \boldsymbol{z}$ is uniformly distributed in $\boldsymbol{Vor}(\mathbf{0})$, and therefore $\mathbf{E}\left[\boldsymbol{z} - \boldsymbol{\theta}\right] = \boldsymbol{x}$; by Lemma 33, $V_{exp}(\boldsymbol{x}+\boldsymbol{\theta})$, and therefore the probability of successful coloring, is symmetric around any lattice point, and so conditioning on successful coloring preserves unbiasedness. □

Once we have a coloring in which there is no $\boldsymbol{z}' \in \Lambda_\epsilon$ with $\boldsymbol{x}+\boldsymbol{\theta} \in \boldsymbol{Vor}^+(\boldsymbol{z}')$ and $c'(\boldsymbol{z}) = c'(\boldsymbol{z}')$, the following simple decoding procedure can find $\boldsymbol{z}$ so long as quantization input $\boldsymbol{x}$ and decoding vector $\boldsymbol{x}_v$ are sufficiently close:

Algorithm 8 SublinearDecode, to compute $R'_{\epsilon,q}(Q'_{\epsilon,q}(\boldsymbol{x}), \boldsymbol{x}_v)$

Using i and shared randomness, reconstruct $\boldsymbol{\theta}$ and c'
Let $\hat{\boldsymbol{z}} \in \Lambda_\epsilon$ such that $B_{q\epsilon}(\boldsymbol{x}_v + \boldsymbol{\theta})$ intersects $\boldsymbol{Vor}(\hat{\boldsymbol{z}})$, and $c'(\hat{\boldsymbol{z}})$ matches $Q'_{\epsilon,q}(\boldsymbol{x})$
Output $\hat{\boldsymbol{z}} - \boldsymbol{\theta}$

Lemma 35. *If $\|\boldsymbol{x} - \boldsymbol{x}_v\|_2 \leq q\epsilon$, then the decoding procedure $R'_{\epsilon,q}(Q'_{\epsilon,q}(\boldsymbol{x}), \boldsymbol{x}_v)$ correctly returns the vector $\boldsymbol{z} - \boldsymbol{\theta}$ which is an unbiased estimator of $\boldsymbol{x}$.*

Proof. Clearly $\boldsymbol{z}$ is has the received color; we must show that $B_{q\epsilon}(\boldsymbol{x}_v+\boldsymbol{\theta})$ intersects $\boldsymbol{Vor}(\boldsymbol{z})$, and does not intersect $\boldsymbol{Vor}(\boldsymbol{z}')$ for any other $\boldsymbol{z}'$ with the same color. The former is the case since $\boldsymbol{x}+\boldsymbol{\theta} \in \boldsymbol{Vor}(\boldsymbol{z})$ and $\|(\boldsymbol{x}+\boldsymbol{\theta}) - (\boldsymbol{x}_v+\boldsymbol{\theta})\| \leq q\epsilon$. The latter holds since if $B_{q\epsilon}(\boldsymbol{x}_v+\boldsymbol{\theta})$ intersects $\boldsymbol{Vor}(\boldsymbol{z}')$, then so does $B_{2q\epsilon}(\boldsymbol{x}+\boldsymbol{\theta})$, and so $\boldsymbol{x}+\boldsymbol{\theta} \in \boldsymbol{Vor}^+(\boldsymbol{z}')$. However, the coloring was successful, so there is no $\boldsymbol{z}' \neq \boldsymbol{z}$ for which this is the case. So, Algorithm 8 must successfully decode $\boldsymbol{z}$. □

Theorem 36. *Algorithms 7 and 8 give a quantization procedure with the following properties: If $\|\boldsymbol{x} - \boldsymbol{x}_v\|_2 \leq q\epsilon$, then the decoder outputs an unbiased estimator $\hat{\boldsymbol{z}} - \boldsymbol{\theta}$ of $\boldsymbol{x}$ with $\|\boldsymbol{x} - (\hat{\boldsymbol{z}} - \boldsymbol{\theta})\|_2 \leq 3\epsilon$. Each machine communicates $O(b)$ bits in expectation (and with probability at least $1 - 2^{(1-b)2^b}$), for $b = d\log(1+q)$.*

Proof. It remains only to prove a bound on the number of bits transmitted: transmitting $c'(\boldsymbol{z})$ requires $\log\left((1+2q)^{3d}\right) = O(b)$ bits. To bound the number of encoding iterations required (and therefore the size of i), we see that each iteration independently succeeds with probability at least $1 - 2(1+2q)^{-d} \geq 1 - 2^{1-b}$, by Lemma 34. Then, for any $j \in \mathbb{N}$, $\mathbf{Pr}\left[i > 2^j\right] \leq 2^{(1-b)2^j}$. Transmitting a value of $i \leq 2^j$ requires j bits, and therefore, setting $j = b$, we see that transmitting i uses b bits with probability at least $1 - 2^{(1-b)2^b}$, and in this case the total amount of communication used is $O(b)$. □

Note that we use Algorithms 7 and 8 primarily for the sublinear communication regime, i.e., when $q < 1$, and in this case we have $b = d\log(1+q) = \Theta(dq)$.

F.1 MeanEstimation and VarianceReduction with Sublinear Communication

If we wish to use $o(d)$-bit messages, then by Theorems 26 and 8, we cannot achieve lower output variance than input variance. Therefore, there is no longer any benefit to averaging; we can instead simply choose a random machine to broadcast its quantized input to all other machines.

Algorithm 9 SublinearMeanEstimation(q)

Choose a source machine u uniformly at random
Machine u broadcasts $Q'_{\frac{y}{q},q}(\boldsymbol{x}_u)$ to all other machines
Each machine v outputs $R'_{\frac{y}{q},q}(Q'_{\frac{y}{q},q}(\boldsymbol{x}_u), \boldsymbol{x}_v)$

Theorem 37. *For any $q = O(1)$, Algorithm 9 performs* MeanEstimation *with each machine using $O(dq)$ communication bits in expectation (and with probability at least $1 - 2^{(1-dq)2^{\Theta(dq)}}$), and with $O(\frac{y^2}{q^2})$ output variance under ℓ_2 norm.*

Proof. The input of the chosen source $\boldsymbol{x}_u$ is an unbiased estimator of the mean input $\boldsymbol{\mu}$ with, $\|\boldsymbol{x}_u - \boldsymbol{\mu}\|_2 \leq y$. Since all machines have $\|\boldsymbol{x}_u - \boldsymbol{x}_v\|_2 \leq \frac{y}{q} \cdot q = y$, all machines correctly decode a common unbiased estimate $\boldsymbol{z}$ of $\boldsymbol{x}_u$ with $\|\boldsymbol{z} - \boldsymbol{x}_u\|_2 \leq 3\frac{y}{q}$. For $q = O(1)$, the output is therefore an unbiased estimate of $\boldsymbol{\mu}$ with $O(\frac{y^2}{q^2})$ variance.

If the broadcast is performed using a binary tree communication structure, then each machine receives at most one message and sends at most two. We have $b = d\log(1+q) = \Theta(dq)$, and therefore each machine requires $O(dq)$ communication bits, in expectation and indeed with probability at least $1 - 2^{(1-dq)2^{\Theta(dq)}}$. □

Together with Theorem 16, this implies Theorem 27.

We again apply the same reduction as for Theorem 17 to obtain the following sublinear-communication result for VarianceReduction (though the name is less fitting, since variance is now necessarily *increased*):

Theorem 38. *For any $\alpha > 1$, and any $q = O(1)$, Algorithm 9 performs* VarianceReduction *using $O(dq)$ communication bits per machine (with probability at least $1-2^{(1-dq)2^{\Theta(dq)}}$), with $O(\frac{\alpha n \sigma^2}{q^2})$ output variance under ℓ_2 norm, succeeding with probability at least $1 - \frac{1}{\alpha}$.*

Proof. As in proof of Theorem 17, we have that a MeanEstimation algorithm using $y = 2\sigma\sqrt{\alpha n}$ performs VarianceReduction, succeeding with probability at least $1 - \frac{1}{\alpha}$, which we plug into Theorem 37. □

G Complete Experimental Report

We consider three distinct applications: data-parallel SGD, distributed power iteration for eigenvalue computation (Suresh et al., 2017), and adding communication compression to local SGD (Stich, 2018). We implement the practical version of our algorithm (Section G.1), which for simplicity we call LQSGD, using a cubic lattice and ℓ_∞ to measure input variance, and a version we call RLQSGD with the Hadamard rotation to ensure close-to-optimal variance.

G.1 The Algorithm in Practice

Practically, the LQSGD algorithm with two machines (u and v) works as follows: we first offset the cubic lattice by a uniformly random vector in $[-\frac{s}{2}, \frac{s}{2}]^d$, using shared randomness. This ensures that quantizing to the *nearest* lattice point now gives an unbiased estimator, avoiding the extra cost of the convex-hull method (though that can still be employed practically if shared randomness is unavailable). Machine u rounds to its closest lattice point and sends its mod-q color to machine v, who decodes it by finding its closest lattice point with the same color, and vice versa. Finding the closest lattice point under the cubic lattice can be performed coordinate-wise using $\tilde{O}(d)$ computation, and gives a lattice point within $\frac{s}{2}$ ℓ_∞ norm distance. Mod-q coloring can also be efficiently computed, and has the property that any two lattice points $\boldsymbol{\lambda}$, $\boldsymbol{\lambda}'$ with the same coloring have $\|\boldsymbol{\lambda} - \boldsymbol{\lambda}'\|_\infty \geq qs$. Therefore, if the input gradients $\boldsymbol{g}_0$, $\boldsymbol{g}_1$ have $\|\boldsymbol{g}_0 - \boldsymbol{g}_1\|_\infty \leq \frac{(q-1)s}{2}$, then decoding is successful. So,

assuming we have an estimate y such that for all $\boldsymbol{g}_0$, $\boldsymbol{g}_1$ we have $\|\boldsymbol{g}_0 - \boldsymbol{g}_1\|_\infty \leq y$, we set our side-length $s = \frac{2y}{q-1}$.

We also implement the algorithm using the structured random rotation described in Appendix D, which we call RLQSGD. Here, we also generate the matrix D on machines using shared randomness, and then apply the transformation HD to inputs before quantization. The algorithm then proceeds exactly as LQSGD; when setting lattice side length we use an estimate y_R of ℓ_∞-norm distance *after* applying HD, i.e., $||HD(g_0 - g_1)||_\infty \leq y_R$.

We will describe in each experiment how we set and update our estimate of y; generally, this can be done by simply measuring ℓ_∞ norms between inputs during the course of the algorithm, and multiplying these by a small constant factor (1.5 to 3.5) to ensure sufficient slack that all decodes succeed.

G.2 Least-Squares Regression

The classic problem of least-squares is as follows: given as input some matrix A and target vector $\boldsymbol{b}$, our goal is to find $\boldsymbol{w}^* = \text{argmin}_{\boldsymbol{w}} ||A\boldsymbol{w} - \boldsymbol{b}||_2^2$, i.e. the vector which, upon multiplication by A, minimizes Euclidean distance to $\boldsymbol{b}$.

To obtain instances of the problem in order to test our approach, we generate $\boldsymbol{w}^* \in \mathbb{R}^d$ and entries of $A \in \mathbb{R}^{S \times d}$ by sampling from $\mathcal{N}(0, 1)$, and we set $\boldsymbol{b} = A\boldsymbol{w}^*$. We then run distributed gradient descent using our quantization scheme, with the following settings:

1. $S = 8192$ samples in $d = 100$ dimensions
2. $n = 2$ worker machines
3. results are averaged over 5 random seeds [0, 10, 20, 30, 40]

In all experiments, in each iteration of gradient descent, the dataset (i.e., the rows of matrix A) will be randomly divided into two equal $\frac{S}{2}$-sized groups and provided to the two machines. We denote $\frac{S}{2}$-batch gradients by $\boldsymbol{g}_0$, $\boldsymbol{g}_1$ respectively. We use large batch sizes in order to be in a regime where our algorithm's assumptions can give a visible advantage over other bounded norm based approaches.

Experiment 1: Norms relevant to quantization schemes. Our first experiment is to provide justification of our main message, that *input variance* (or similar measure of mutual distance depending on the setting) is often far lower than *input norm* in practical applications, and is a more natural quantity with which to measure output variance of algorithms.

In Figures 4 and 5 we compare four quantities, in the setting of least-squares on two worker machines:

- batch gradient distance $\|\boldsymbol{g}_0 - \boldsymbol{g}_1\|_2$ (the closest proxy of our theoretical quantity y from MeanEstimation),
- $\|\boldsymbol{g}_0 - \boldsymbol{g}_1\|_\infty$ (the equivalent quantity under ℓ_∞ norm, which is the appropriate norm when using the cubic lattice),
- $\|\boldsymbol{g}_0\|_2$, the batch gradient norm, (which is used as the measure of input size in QSGD-L2 (Alistarh et al., 2017), as well as most other prior work)
- $\max(\boldsymbol{g}_0) - \min(\boldsymbol{g}_0)$, the batch gradient coordinate difference, used as the measure of input size in the QSGD implementation (Alistarh et al., 2017)

We see that the former two quantities, used by our algorithms, are far lower than the latter two, i.e., input variance/distance is significantly lower than input norm, and batch gradients are not centered around the origin. Similar effects are seen in a wide variety of applications and parameter settings, and justify our focus on input distances over norms. Iterations in the figures are under gradient descent performed using full (unquantized) gradient.

Experiment 2: Variances using quantization methods. We now apply quantization schemes to the communication gradients, and measure the variances induced, in order see whether the input norm versus variance difference demonstrated by Figures 4 and 5 translates to lower output variance for LQSGD over other schemes.

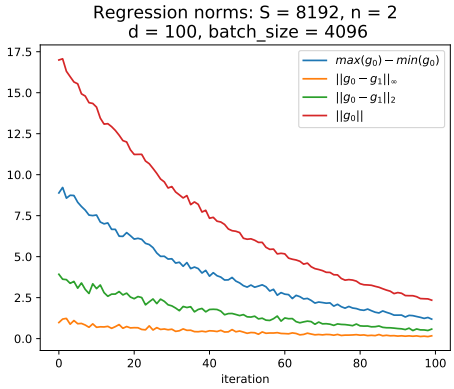

Figure 4: norms using fewer samples

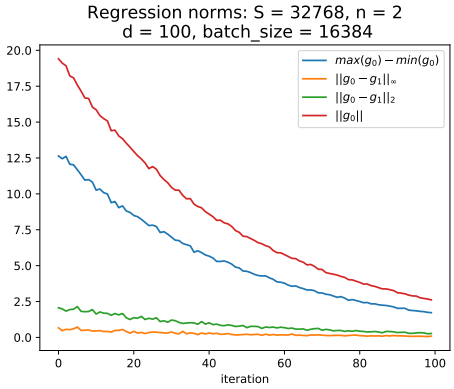

Figure 5: norms using more samples

In this experiment, we perform distributed stochastic gradient descent, by communicating, averaging, and applying the *quantized* batch gradients $\boldsymbol{g}_0$ and $\boldsymbol{g}_1$ of the two worker machines in each iteration. By construction, this resulting value EST will be an unbiased estimator of the full true gradient $\boldsymbol{\nabla}$, and we will examine the associated *output variance* $\mathbb{E}[\|EST - \boldsymbol{\nabla}\|_2^2]$, for different quantization schemes. We experiment with $q = 8$ for all methods (for QSGD the equivalent quantity is referred to as qlevel), which means that messages comprise of $\log_2 8 = 3$ bits per coordinate. In norm-based methods like QSGD, there is an additional communication overhead of one/two 64−bit floating point values for every quantization, used to convey input norms.

To allow machines using LQSGD to maintain an estimate y of distance between inputs, in the first iteration alone a pre-computed estimate is provided to both machines. (To ensure that this does not convey an unfair advantage, we allow the other methods to update using the full true gradient $\boldsymbol{\nabla}$ in the first iteration.) Henceforth, machines dynamically adjust y, to ensure that we always have $\|\boldsymbol{g}_0 - \boldsymbol{g}_1\|_\infty < y$, in order for decoding to be successful. For each iteration t, denoting as $Q(\boldsymbol{g}_0), Q(\boldsymbol{g}_1)$ the encoded (i.e., rounded to closest lattice point) estimates, machines use the value $y(t+1) = 1.5 \cdot \|Q(\boldsymbol{g}_0) - Q(\boldsymbol{g}_1)\|_\infty$ for the next iteration. Similarly, for RLQSGD, $y_R(t+1) = 1.5 \cdot \|HD(Q(\boldsymbol{g}_0) - Q(\boldsymbol{g}_1))\|_\infty$. This value is common knowledge to both machines as long as decoding in iteration t is successful, and sufficed, in this experiment, to ensure that all decodes succeeded.

Our baseline is *naive averaging* (i.e., what would be attained if we could communicate at full precision between the machines), and we compare against variants of *QSGD* (Alistarh et al., 2017), and the *Hadamard*-based quantization scheme of Suresh et al. (2017).

Figures 6 and 7 demonstrate that LQSGD provides the lowest output variance among the methods, and is the only method to achieve *variance reduction*, i.e. the output variance of the average of the quantized batch gradients is lower than the input variance of the *individual* batch gradients.

Experiment 3: Convergence using quantization methods. We next measure the effect on the *convergence* of the SGD process of the quantization schemes, using the same input data. That is, machines now apply the unbiased estimate of $\boldsymbol{\nabla}$ they obtain by averaging quantized batch gradient in each iteration. To clearly show the effects of quantization on convergence, we will use a high learning rate of 0.8. Estimation of y, and other parameter settings, remain the same as previously. Figures 8 and 9 demonstrate faster convergence for LQSGD over other methods.

Experiment 4: Exploration of potential variance with sub-linear quantization. We also wish to test the performance of our sublinear quantization scheme from Section F. Unfortunately, a naive implementation of the scheme (particularly, of finding the closest lattice point of a particular color under the chosen random coloring) is not computationally feasible under high enough dimension for meaningful results. Therefore, here we will only simulate what the effect of the quantization would be on output variance compared to

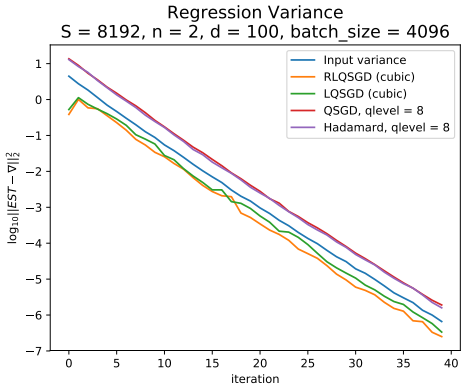

Figure 6: variance at 3 bits per coordinate, fewer samples

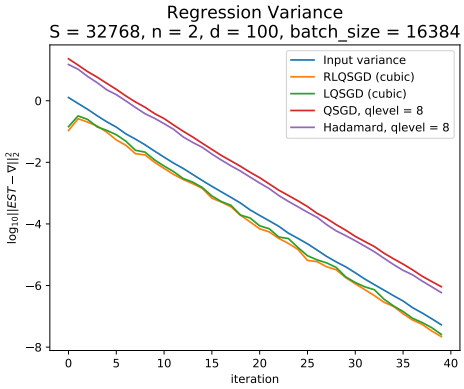

Figure 7: variance at 3 bits per coordinate, more samples

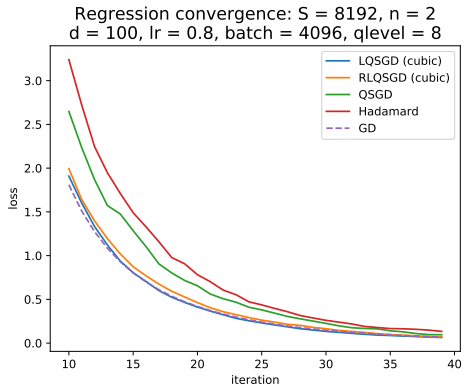

Figure 8: convergence at 3 bits per coordinate, fewer samples

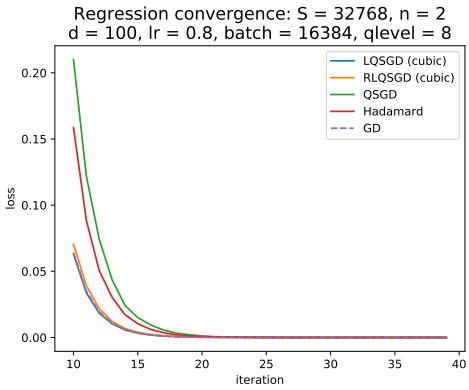

Figure 9: convergence at 3 bits per coordinate, more samples

state-of-the-art schemes; in future work we will explore how the scheme can be adapted for efficient implementation.

In this experiment we again have two worker machines u, v, and we compare the variance induced when u sends its quantized batch gradient $\boldsymbol{g}_0$ to v, who decodes it using $\boldsymbol{g}_1$. The sublinear-communication quantization method that we compare with is the vQSGD cross polytope method with repetition (Gandikota et al., 2019). We will perform our experiments using $\frac{d}{2}$ bits, i.e., 0.5 bits per coordinate (and so we set the number of vQSGD repetitions accordingly).

We use a slightly different means of dynamically updating y, since variance from quantization means that the previous method no longer gives good estimates. Now, once in 5 iterations, machine u receives 2 batches, which allows it to compute two gradient estimates $\boldsymbol{g}_0$ and $\boldsymbol{g}_0'$, and compute $y = 1.6 \cdot ||\boldsymbol{g}_0 - \boldsymbol{g}_1||_\infty$ (which suffices to ensure correct decoding in this experiment). It then sends this to v as a 64-bit floating point value. Note that this method of updating y generalizes to many machines, and that the constant factor 1.6 can be changed as necessary.

We use the cubic lattice, and as before denote its side length by s. Given $\|\boldsymbol{g}_0 - \boldsymbol{g}_1\|_\infty < y$. We can see from the analysis of the sublinear scheme (in particular, in proof of Lemma 32) that if we choose a number of *colors* (i.e., bit-strings) equal to the expected number of expanded Voronoi regions covering a point, quantization will succeed with at least constant probability. This gives an expression of $\log_2 (1 + 2y/r_p)^d = d \log_2 (1 + 4y/s)$ bits. To use $0.5d$ bits, we set $\log_2 (1 + 4y/s) = 0.5$, from which we get $s = 4y/(\sqrt{2} - 1)$ (though our number of bits and therefore value of s slightly differs from this in order to exactly match

the communication used by vQSGD, where number of repetitions must be an integer). Since the randomization in the quantization effectively shifts each coordinate independently in $[-s/2, s/2]$, we get variance $ds^2/12$, which is what we plot in the figures.

Figures 10 and 11 demonstrate that sublinear LQSGD could be competitive with state-of-the-art methods, though it only outperforms when using large numbers of samples with respect to dimension. The steps in the graph are due to the periodic updates to y.

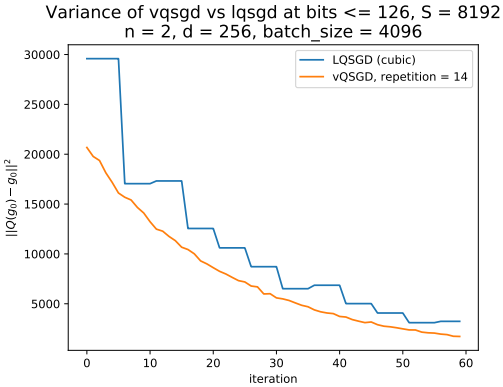

Figure 10: sublinear schemes, variance for fewer samples

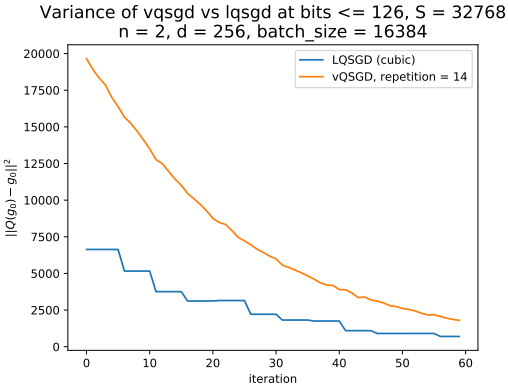

Figure 11: sublinear schemes, variance for more samples

Experiment 5: Convergence on real dataset using quantization schemes with more than 2 machines. We now test the performance of the (linear/superlinear) quantization scheme in a more realistic setting, on a real dataset using multiple machines. We use the dataset `cpusmall_scale` from the LIBSVM datasets (Chang & Lin, 2011), which has $S = 8192$ and $d = 12$. We initialize the initial weight to a point far from the origin relative to $\boldsymbol{w}_{\text{opt}}$, specifically the vector of -1000 in each coordinate. This is to synthetically study the behavior one can expect on a general convex loss function when w_{opt} is arbitrary. We study the convergence behavior with $q = 16$, $n = 8, 16$ and batch size $= \frac{S}{n}$.

We apply Algorithm 3, with one randomly chosen machine acting as leader to collect all the quantized batch gradients $Q(\boldsymbol{g}_i)$, and broadcast the average as a quantized vector i.e $Q\left(\sum_i Q(\boldsymbol{g}_i)/n\right)$. This machine also uses the quantized gradients $Q(\boldsymbol{g}_i)$ to evaluate and broadcast y for the next round as a 64-bit float using $y = 3 \cdot \max_{i,j} ||Q(\boldsymbol{g}_i) - Q(\boldsymbol{g}_j)||_\infty$, where the factor 3 is again chosen large enough to ensure that decoding is always successful in this experiment (and can be altered as necessary in other settings).

Figures 12 and 13 exhibit fast convergence compared to other methods.

G.3 Neural Network Training

Our next task is applying LQSGD for training neural networks in a data-parallel environment, where we use it to average gradients. We note that this application slightly extends the setting for distributed mean estimation (Suresh et al., 2017), since other methods can, e.g., employ historical information from previous iterations (Stich et al., 2018; Karimireddy et al., 2019).

Experiment 6: Gradient compression for neural network training. We train ResNets (He et al., 2016) on a subset of 10 classes from the ILSVRC dataset (Deng et al., 2009), with full-sized images, as well as on the CIFAR dataset (Krizhevsky). The former dataset is popular for model and hyperparameter calibration (Howard) relative to the full version of the dataset, and model performance on it is known to correlate well with performance on the full ILSVRC dataset. The latter is a classic small dataset in computer vision.

The results in Figure 14 show the Top-1 validation accuracy on ResNet18 (11M parameters) for LQSGD with an average of 4 bits per entry, versus 4-bit QSGD (L2 and LInf normalized

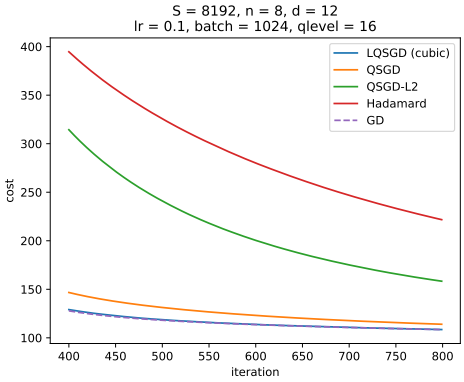

Figure 12: `cpusmall_scale` dataset, 8 machines

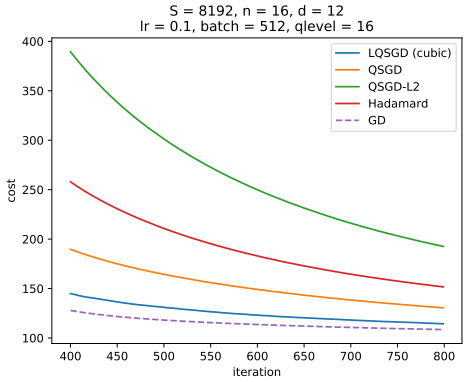

Figure 13: `cpusmall_scale` dataset, 16 machines

	ResNet18	
Compression type	**Train**	**Validation**
none	**88.7**	**88.5**
QSGD (Linf)	84.9	86.2
QSGD (L2)	84.7	86.1
EFSignSGD	50.4	55.9
PowerSGD	83.1	85.5
LQSGD	84.4	86.6

Figure 14: Accuracy results for ResNet18 on ILSVRC

	CIFAR100	
Compression type	**Train**	**Validation**
none	**100**	**76**
QSGD (Linf)	100	75
QSGD (L2)	96.9	71
EFSignSGD	18.8	23
PowerSGD	100	75
LQSGD	93.8	75

Figure 15: Top-1 Validation Accuracy for ResNet20 on CIFAR-100.

variants), PowerSGD (Vogels et al., 2019) with rank 16 (as suggested by the authors), and 1bitSGD/EFSignSGD (Seide et al., 2014a; Karimireddy et al., 2019). For LQSGD, each node uses one batch per epoch to estimate σ, and uses $y = 3\sigma$ as its upper bound estimate. (This results in a rate of incorrect decodings of $\sim 3\%$, which we allow, but which has no impact on convergence.) Results are averaged over 2 runs, since variance is small.

We note that all compression algorithms lose accuracy relative to the full-precision baseline. (The large gap for EFSignSGD is reasonable, since it uses the fewest bits.) Perhaps surprisingly, LQSGD loses the least accuracy relative to the baseline, although the difference relative to QSGD is small. We conclude that, although gradient compression is not the main application for lattice-based quantization, it can yield competitive results for this task.

Figure 15 presents additional accuracy results for neural network training with compressed gradients, training ResNet20 on the CIFAR-100 dataset (Krizhevsky; Krizhevsky et al., 2014). The network is trained for 200 epochs, with standard hyper-parameters. Quantization is applied at the level of each layer, and we use 4 bits per coordinate (except for EF-SignSGD which uses approximately one bit per component). The results confirm that LQSGD can be competitive with state-of-the-art quantization methods.

G.4 Power Iteration

Power iteration is an algorithm for finding the principal eigenvector of an input matrix X. In a distributed setting over two machines, the algorithm works as follows: the rows of our input matrix X are partitioned into two subsets X_0, X_1. At the beginning of each round, both machines have the same unit vector $\boldsymbol{x}$, which is the current estimate of the principal eigenvector of X. During the round, they must perform the update $\boldsymbol{x} \leftarrow \frac{X^T X \boldsymbol{x}}{||X^T X \boldsymbol{x}||}$. For this, machine i evaluates $\boldsymbol{u}_i = X_i^T X_i \boldsymbol{x}$ and shares it with the other machine; both machines can

then calculate $X^T X\boldsymbol{x} = X_0^T X_0 \boldsymbol{x} + X_1^T X_1 \boldsymbol{x} = \boldsymbol{u}_0 + \boldsymbol{u}_1$, and thereby perform the update step. We will apply quantization methods to communicate these vectors $\boldsymbol{u}_i$, in order to test the performance of LQSGD in this setting.

Experiment 7: Quantization for compression during power iteration. Each row of the input matrix X is generated from a multivariate gaussian with first two eigenvalues large and comparable, which is to ensure that power iteration doesn't converge too fast that we cannot observe the effect of quantization. The estimate $\boldsymbol{x}$ of the principal eigenvector is initialized to a random unit vector. We use $S = 8192$ samples, dimension $d = 128$, $q = 64$ i.e., 6 bits per coordinate.

From Figure 16, 17 one can see that the relevant norms fit with our general message concerning norms: distance between quantization inputs (used in LQSGD and RLQSGD) is substantially lower than difference between coordinate values within a single quantization input (used as a surrogate for input norm in QSGD). Also, we see that these norms settle quickly (relative to the number of iterations one would run to get a good estimate of the principal eigenvector) and are then near-constant. Hence, one can run the baseline algorithm (either on a single machine, or assuming full-precision communication) for a few iterations until $||\boldsymbol{u}_0 - \boldsymbol{u}_1||_\infty$ stabilizes, and then set the value of y for LQSGD to $2 \cdot \max ||\boldsymbol{u}_0 - \boldsymbol{u}_1||_\infty$ (where the maximum is over all iterations currently seen). Similarly, for RLQSGD we set $y_R = 2 \cdot \max ||HD(\boldsymbol{u}_0 - \boldsymbol{u}_1)||_\infty$. We then run LQSGD and RLQSGD from iteration 0, but using the computed value of y.

Upon doing so, Figure 16, 17 demonstrates substantially better estimation of the principal eigenvector for RLQSGD and LQSGD compared to other quantization methods.

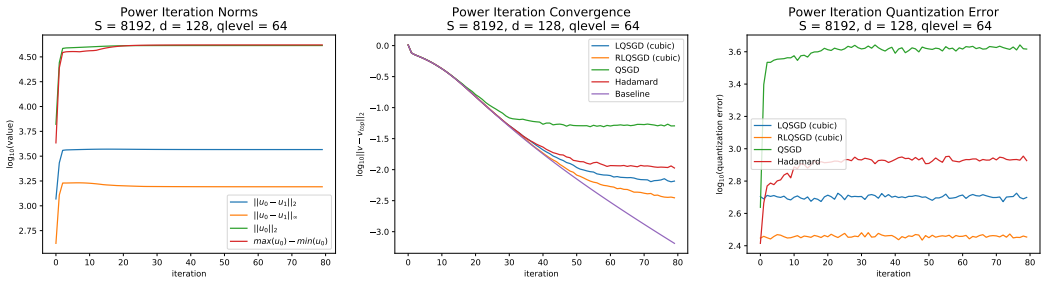

Figure 16: Power iteration: input norms (left), convergence (center) and quantization error (right). Principal eigenvector is e_2.

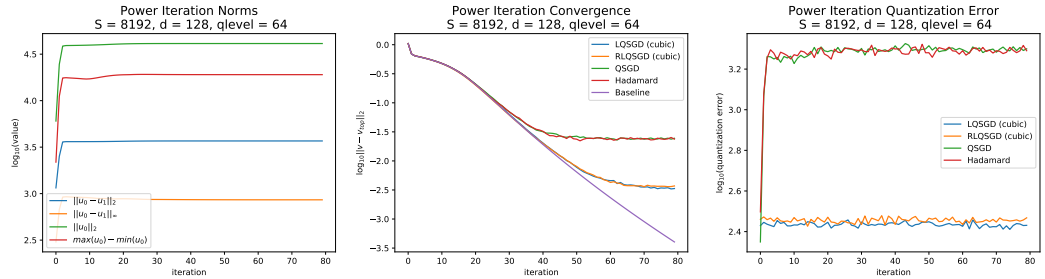

Figure 17: Power iteration: input norms (left), convergence (center) and quantization error (right). Principal eigenvector is along a random direction.