RATQ: An Efficient Quantizer for Stochastic Optimization

Abstract

We present Rotated Adaptive Tetra-iterated Quantizer (RATQ), a fixed-length quantizer for gradients in first order stochastic optimization. For gradient oracles with almost surely bounded Euclidean norm of the oracle output, we establish an information theoretic lower bound for optimization accuracy using finite precision gradients and show that RATQ, when used along with SGD, is merely a factor $O(\log \log \log^* d)$ far from this lower bound. We remark that RATQ offers an almost optimal-size covering of the unit ball in Euclidean space, an object of long-standing interest and many applications.

For oracles with mean square bounded oracles, we use a *gain-shape* quantizer which separately quantizes the Euclidean norm and uses RATQ to quantize the normalized unit norm vector. We establish lower bounds for performance of any optimization procedure and shape quantizer, when used with a uniform gain quantizer. Finally, we propose an adaptive quantizer for gain which when used with RATQ for shape quantizer outperforms uniform gain quantization.

1 Introduction

Stochastic gradient descent (SGD) and its variants are popular optimization methods for machine learning. In its basic form, SGD performs iterations $x_{t+1} = x_t - \eta \hat{g}(x_t)$, where $\hat{g}(x)$ is a noisy estimate of the subgradient of the function being optimized at x. Our focus in this work is on a distributed implementation of this algorithm where the output $\hat{g}(x)$ of the first order oracle must be quantized to a precision of r bits. This

abstraction models important scenarios ranging from distributed optimization to federated learning, and is of independent theoretical interest.

We study the tradeoff between the convergence rate of first order optimization algorithms and the precision r available per subgradient update. We consider two *oracle models*: the first where the subgradient estimate's Euclidean norm is *almost surely bounded* and the second where it is *mean square bounded*. Our main contributions include new quantizers for the two oracle models and theoretical insights into the limitations imposed by heavy-tailed gradient distributions admitted under the mean square bounded oracles. A more specific description is provided below.

1.1 Prior work and our contributions

SGD and the oracle model abstraction for it appeared in classic works [21] and [19], respectively. Recently, variants of this problem with quantization or communication constraints on oracle output have received a lot of attention [1, 4–6, 9, 12, 16, 20, 22–25]. Our work is motivated by results in [5, 23], and we elaborate on the connection.

Specifically, [5] considers a problem very similar to ours. The paper [23] considers the related problem of distributed mean estimation, but the quantizer and its analysis is directly applicable to distributed optimization. The two papers present slightly different quantizers that encode each input using a variable number of bits. Both these quantizers are of optimal expected precision for almost surely bounded oracles. However, their worst-case (fixed-length) performance is suboptimal. In fact, the problem of designing fixed-length quantizers for almost surely bounded oracles is closely related to designing small-size covering for the Euclidean unit ball. There has been a longstanding interest in this problem in the vector quantization and information theory literature (*cf.* [8, 11, 15, 17, 26]). We propose a new quantizer that is merely a factor $O(\log \log \log^* d)$ far from an optimal information theoretic benchmark which we establish.

In a different direction, for mean square bounded ora-

cles, prior works including [5] remains vague about the quantizer. Most of the works use gain-shape quantizers that separately quantize the Euclidean norm and the normalized vector. But they operate under an engineering assumption: "the standard 32 bit precision suffices for describing the gain." We suppose that this folklore wisdom is prescribing the use of the standard uniform quantizer for gain quantization. We carefully examine the validity of this assumption and the design of the gain quantizers for mean square bounded oracles.

We establish an information theoretic lower bound which shows (using a heavy-tailed oracle) that the precision used for gain quantizer must exceed $\log T$ when the gain is quantized uniformly for T iterations and we seek $O(1/\sqrt{T})$ optimization accuracy. Thus, 32 bits are good for roughly a billion iterations with uniform gain quantizers, but not beyond that. Interestingly, we present a new, adaptive gain quantizer which can attain the same performance using only $\log \log T$ bits for quantizing gain. Thus, if one has 32 bits to spare for gain, then by using our quantizer we can handle algorithms with $2^{2^{32}}$ iterations, sufficient for any practical application.

1.2 Organization

We formalize our problem in the next section, describe our results for almost surely bounded oracles in Section 3 and for mean square bounded oracles in Section 4. We present application to distributed mean estimation in Section 5 and a high-level overview of our proof techniques in Section 6.

2 The setup and preliminaries

2.1 Problem setup

We fix the number of iterations T of the optimization algorithm (the number of times the first order oracle is accessed) and the precision r allowed to describe each subgradient. Our fundamental metric of performance is the minimum error (as a function of T and r) with which such an algorithm can find the optimum value.

Formally, we want to find the minimum value of an unknown convex function $f : \mathcal{X} \to \mathbb{R}$ using *oracle access* to noisy subgradients of the function (*cf.* [7, 19]). We assume that the function f is convex over the compact, convex domain $\mathcal{X}$ such that $\sup_{x,y \in \mathcal{X}'} \|x - y\|_2 \leq D$; we denote the set of all such $\mathcal{X}$ by $\mathbb{X}$. For a query point $x \in \mathcal{X}$, the oracle outputs random estimates of the subgradient $\hat{g}(x)$ which for all $x \in \mathcal{X}$ satisfy

$$\mathbb{E}\left[\hat{g}(x)|x\right] \in \partial f(x), \tag{1}$$

$$\mathbb{E}\left[\|\hat{g}(x)\|_2^2|x\right] \leq B^2, \tag{2}$$

where $\partial f(x)$ denotes the set of subgradients of f at x.

Definition 2.1 (Mean square bounded oracle)**.** A first order oracle which upon a query x outputs the subgradient estimate $\hat{g}(x)$ satisfying the assumptions (1) and (2) is termed a mean square bounded oracle. We denote by $\mathcal{O}$ the set of pairs (f, O) with a convex function f and a mean square bounded oracle O.

The variant with almost surely bounded oracles has also been considered (*cf.* [2, 19]), where we assume for all $x \in \mathcal{X}$

$$P(\|\hat{g}(x)\|_2^2 \leq B^2|x) = 1. \tag{3}$$

Definition 2.2 (Almost surely bounded oracle)**.** A first order oracle which upon a query x outputs only the subgradient estimate $\hat{g}(x)$ satisfying the assumptions (1) and (3) is termed an almost surely bounded oracle. We denote the class of convex functions and oracle's satisfying assumptions (1) and (3) by $\mathcal{O}_0$.

In our setting, the outputs of the oracle are passed through a quantizer. An *r-bit quantizer* consists of randomized mappings $(Q^{\mathsf{e}}, Q^{\mathsf{d}})$ with the encoder mapping $Q^{\mathsf{e}} : \mathbb{R}^d \to \{0, 1\}^r$ and the decoder mapping $Q^{\mathsf{d}} : \{0, 1\}^r \to \mathbb{R}^d$. The overall quantizer is given by the composition mapping $Q = Q^{\mathsf{d}} \circ Q^{\mathsf{e}}$. Denote by $\mathcal{Q}_r$ the set of all such r-bit quantizers.

For an oracle $(f, O) \in \mathcal{O}$ and an r-bit quantizer Q, let $QO = Q \circ O$ denote the composition oracle that outputs $Q(\hat{g}(x))$ for each query x. Let π be an algorithm with at most T iterations with oracle access to QO. We will call such an algorithm an *optimization protocol*. Denote by Π_T the set of all such optimization protocols with T iterations.

Denoting the combined optimization protocol with its oracle QO by π^{QO} and the associated output as $x^*(\pi^{QO})$, we measure the performance of such an optimization protocol for a given (f, O) using the metric $\mathcal{E}(f, \pi^{QO})$ defined as $\mathcal{E}(f, \pi^{QO}) := \mathbb{E}\left[f(x^*(\pi^{QO})) - \min_{x \in \mathcal{X}} f(x)\right]$. The fundamental quantity of interest in this work are minmax errors

$$\mathcal{E}_0^*(T, r) := \sup_{\mathcal{X} \in \mathbb{X}} \inf_{\pi \in \Pi_T} \inf_{Q \in \mathcal{Q}_r} \sup_{(f,O) \in \mathcal{O}_0} \mathcal{E}(f, \pi^{QO}),$$

$$\mathcal{E}^*(T, r) := \sup_{\mathcal{X} \in \mathbb{X}} \inf_{\pi \in \Pi_T} \inf_{Q \in \mathcal{Q}_r} \sup_{(f,O) \in \mathcal{O}} \mathcal{E}(f, \pi^{QO}).$$

Clearly, $\mathcal{E}^*(T, r) \geq \mathcal{E}_0^*(T, r)$.

2.2 A benchmark from prior results

We recall results for the classic setting with $r = \infty$. Prior work gives a complete characterization of the minmax errors $\mathcal{E}_0^*(T, \infty)$ and $\mathcal{E}^*(T, \infty)$ for this setting; see, for instance, [3, 18, 19]. We summarize these well-known results below (*cf.* [19], [2, Theorem 1a]).

Theorem 2.3. *For an absolute constant c_0, we have* $c_0 DB/\sqrt{T} \leq \mathcal{E}_0^*(T, \infty) \leq \mathcal{E}^*(T, \infty) \leq DB/\sqrt{T}$.

2.3 Quantizer performance for finite precision optimization

Our overall optimization protocol throughout is the *projected SGD* (PSGD) (see [7]). In fact, we establish lower bound showing roughly the optimality of PSGD with our quantizers.

In PSGD the standard SGD updates are projected back to the domain using the projection map $\Gamma_{\mathcal{X}}$ given by $\Gamma_{\mathcal{X}}(y) := \min_{x \in \mathcal{X}} \|x - y\|_2$. We use the *quantized PSGD* algorithm described in Algorithm 1.

Require: $x_0 \in \mathcal{X}, \eta \in \mathbb{R}^+, T$ and access to composed oracle QO
1: **for** $t = 0$ to $T - 1$ **do**
$$x_{t+1} = \Gamma_{\mathcal{X}}(x_t - \eta Q(\hat{g}(x_t)))$$
2: **Output:** $\frac{1}{T} \cdot \sum_{t=1}^T x_t$

Algorithm 1: Quantized PSGD with quantizer Q

The quantized output $Q(\hat{g}(x_t))$, too, constitutes a noisy oracle, but it can be biased for mean square bounded oracles. Though biased first-order oracles were considered in [14], the effect of quantizer-bias has not been studied in the past. The performance of a quantizer Q, when it is used with PSGD for mean square bounded oracles, is controlled by the worst-case L_2 norm $\alpha(Q)$ of its output and the worst-case bias $\beta(Q)$ defined as[1]

$$\alpha(Q) := \sup_{Y \in \mathbb{R}^d : \mathbb{E}[\|Y\|_2^2] \leq B^2} \sqrt{\mathbb{E}[\|Q(Y)\|_2^2]},$$
$$\beta(Q) := \sup_{Y \in \mathbb{R}^d : \mathbb{E}[\|Y\|_2^2] \leq B^2} \|\mathbb{E}[Y - Q(Y)]\|_2. \quad (4)$$

The corresponding quantities for almost surely bounded oracles are

$$\alpha_0(Q) := \sup_{Y \in \mathbb{R}^d : \|Y\|_2 \leq B \text{ a.s.}} \sqrt{\mathbb{E}[\|Q(Y)\|_2^2]},$$
$$\beta_0(Q) := \sup_{Y \in \mathbb{R}^d : \|Y\|_2 \leq B \text{ a.s.}} \|\mathbb{E}[Y - Q(Y)]\|_2. \quad (5)$$

Using a slight modification of the standard proof of convergence for PSGD, we get the following result.

Theorem 2.4. *For any quantizer Q, the output x_T of optimization protocol π given in Algorithm 1 satisfies*

$$\sup_{(f,O) \in \mathcal{O}_0} \mathcal{E}(f, \pi^{QO}) \leq D\left(\frac{\alpha_0(Q)}{\sqrt{T}} + \beta_0(Q)\right),$$
$$\sup_{(f,O) \in \mathcal{O}} \mathcal{E}(f, \pi^{QO}) \leq D\left(\frac{\alpha(Q)}{\sqrt{T}} + \beta(Q)\right),$$

---

[1]We omit the dependence on B and d from our notation.

when the parameter η is set to $D/(\alpha_0(Q)\sqrt{T})$ and $D/(\alpha(Q)\sqrt{T})$, *respectively.*

Remark 1 (Choice of learning rate). We fix the parameter η of Algorithm 1 to $D/(\alpha_0(Q)\sqrt{T})$ and $D/(\alpha(Q)\sqrt{T})$ for all the results in Section 3 and Section 4, respectively.

3 Main results for almost surely bounded oracles

Our main results will be organized along two regimes: the high-precision and the low-precision regime. For the high-precision regime, we seek to attain the optimal convergence rate of $1/\sqrt{T}$ using the minimum precision possible. For the low-precision regime, we seek to attain the fastest convergence rate possible for a given, fixed precision r.

3.1 A precision-dependent lower bound

We begin with a simple refinement of the lower bound implied by Theorem 2.3 The proof of this result is obtained by appropriately modifying the proof in [2], along with the strong data processing inequality in [10].

Theorem 3.1. *There exists an absolute constant c, independent of d, T, and r such that*

$$\mathcal{E}^*(T, r) \geq \mathcal{E}_0^*(T, r) \geq \frac{cDB}{\sqrt{T}} \cdot \sqrt{\frac{d}{\min\{d, r\}}}.$$

As a corollary, we get that there is no hope of getting the desired convergence rate of $1/\sqrt{T}$ by using a precision of less than d.

Corollary 3.2. *For $\mathcal{E}_0^*(T, r)$ or $\mathcal{E}^*(T, r)$ to be less than $DB/\sqrt{T}$, the precision r must be at least $\Omega(d)$.*

3.2 RATQ: Our quantizer for the ℓ_2 ball

We propose *Rotated Adaptive Tetra-iterated Quantizer* (RATQ) to quantize any random vector Y with $\|Y\|_2^2 \leq B^2$, which is what we need for almost surely bounded oracles. RATQ first rotates the input vector, then divides the coordinates of the rotated vectors into smaller groups, and finally quantizes each subgroup-vector using a *Coordinate-wise Uniform Quantizer* (CUQ). However, the dynamic-range used for each sub-vector is chosen adaptively from a set of tetra-iterated levels. We call this adaptive quantizer *Adaptive Tetra-iterated Uniform Quantizer* (ATUQ), and it is the main workhorse of our construction. The encoder and decoder for RATQ are given in Algorithm 2 and Algorithm 3, respectively. The details of all the components involved are described below.

Require: Input $Y \in \mathbb{R}^d$, rotation matrix R
1: Compute $\tilde{Y} = RY$
2: **for** $i \in [d/s]$ **do**
    $\tilde{Y}_i^T = [\tilde{Y}((i-1)s+1), \cdots \tilde{Y}(\min\{is, d\})]^T$
3: **Output:** $Q_{\mathtt{at},R}^{\mathtt{e}}(Y) = \{Q_{\mathtt{at}}^{\mathtt{e}}(\tilde{Y}_1) \cdots Q_{\mathtt{at}}^{\mathtt{e}}(\tilde{Y}_{\lceil d/s \rceil})\}$

Algorithm 2: Encoder $Q_{\mathtt{at},R}^{\mathtt{e}}(Y)$ for RATQ

Require: Input $\{Z_i, j_i\}$ for $i \in [\lceil d/s \rceil]$, rotation matrix R
1: $Y^T = [Q_{\mathtt{at},R}^{\mathtt{d}}(Z_1, j_1), \cdots, Q_{\mathtt{at},R}^{\mathtt{d}}(Z_{\lceil d/s \rceil}, j_{\lceil d/s \rceil})]^T$
2: **Output:** $Q_{\mathtt{at}}^{\mathtt{d}}(\{Z_i, j_i\}_{i=1}^{\lceil d/s \rceil}) = R^{-1}Y$

Algorithm 3: Decoder $Q_{\mathtt{at},R}^{\mathtt{d}}(Z, j)$ for RATQ

Rotation and division into subvectors. RATQ first rotates the input vector by multiplying it with a random Hadamard matrix. Specifically, denoting by H the $d \times d$ Walsh-Hadamard Matrix (see [13])[2], define $R := \frac{1}{\sqrt{d}} \cdot HD$, where D is a diagonal matrix with each diagonal enetry generated uniformly from $\{-1, +1\}$. The input vector y is multiplied by R in the rotation step. The matrix D can either be generated using shared randomness between the encoder and decoder.

Next, the rotated vector of dimension d is partitioned into $\lceil d/s \rceil$ smaller subvectors. The ith subvector comprises the coordinates $\{(i-1)s+1, \cdots, \min\{is, d\}\}$, for all $i \in [d/s]$. Note that the dimension of all the sub vectors except the last one is s, with the last one having a dimension of $d - s\lfloor d/s \rfloor$.

CUQ. RATQ uses CUQ as a subroutine; we describe the latter for d dimensional inputs, but it will only be applied to subvectors of lower dimension in RATQ. CUQ has a dynamic range $[-M, M]$ associated with it, and it uniformly quantizes each coordinate of the input to k-levels as long as the component is within the dynamic-range $[-M, M]$. Specifically, it partitions the interval $[-M, M]$ into parts $I_\ell := (B_{M,k}(\ell), B_{M,k}(\ell + 1)]$, $\ell \in \{0, \ldots, k-1\}$, where $B_{M,k}(\ell)$ are given by

$$B_{M,k}(\ell) := -M + \ell \cdot \frac{2M}{k-1}, \quad \forall \ell \in \{0, \ldots, k-1\}.$$

Note that we need to communicate $k + 1$ symbols per coordinate – k of these symbols correspond to the k uniform levels and the additional symbol corresponds to the overflow symbol $\emptyset$. Thus we need a total precision of $d \lceil \log(k+1) \rceil$ bits to represent the output of the CUQ encoder. The encoder and decoders used in CUQ are given in Algorithms 4 and 5, respectively. In the decoder, we have set $B_{M,k}(\emptyset)$ to 0.

Require: Parameters $M \in \mathbb{R}^+$ and input $Y \in \mathbb{R}^d$
1: **for** $i \in [d]$ **do**
2:   **if** $|Y(i)| > M$ **then**
    $Z(i) = \emptyset$
3:   **else**
4:     **for** $\ell \in \{0, \ldots, k-1\}$ **do**
5:       **if** $Y(i) \in (B_{M,k}(\ell), B_{M,k+1}(\ell+1)]$ **then**
$$Z(i) = \begin{cases} \ell + 1, & w.p. \ \frac{Y(i) - B_{M,k}(\ell)}{B_{M,k}(\ell+1) - B_{M,k}(\ell)} \\ \ell, & w.p. \ \frac{B_{M,k}(\ell+1) - Y(i)}{B_{M,k}(\ell+1) - B_{M,k}(\ell)} \end{cases}$$
6: **Output:** $Q_{\mathtt{u}}^{\mathtt{e}}(Y; M) = Z$

Algorithm 4: Encoder $Q_{\mathtt{u}}^{\mathtt{e}}(Y; M)$ of CUQ

Require: Parameters $M \in \mathbb{R}^+$ and input $Z \in \{0, \ldots, k-1, \emptyset\}^d$
1: Set $\hat{Y}(i) = B_{M,k}(Z(i))$, for all $i \in [d]$
2: **Output:** $Q_{\mathtt{u}}^{\mathtt{d}}(Z; M) = \hat{Y}$

Algorithm 5: Decoder $Q_{\mathtt{u}}^{\mathtt{d}}(Z; M)$ of CUQ

ATUQ. The quantizer ATUQ is CUQ with its dynamic-range chosen in an adaptive manner. In order to a quantize a particular input vector, it first chooses a dynamic range from $[-M_i, M_i]$, $1 \le i \le h$. To describe these M_is, we first define the ith tetra-iteration for e, denoted by e^{*i}, recursively as follows:

$$e^{*1} := e, \quad e^{*i} := e^{e^{*(i-1)}}, \quad i \in \mathbb{N}.$$

Also, for any non negative number b, we define $\ln^* b := \inf\{i \in \mathbb{N} : e^{*i} \ge b\}$. With this notation, the values M_is are defined in terms of the starting point m as follows:

$$M_0^2 = m, \quad M_i^2 = m \cdot e^{*i}, \quad \forall i \in [h-1].$$

ATUQ finds the smallest level M_i which bounds the infinity norm of the input vector; if no such M_i exists, it simply uses M_{h-1}. It then uses CUQ with dynamic range $[-M_i, M_i]$ to quantize the input vector. In RATQ, we apply ATUQ to each subvector. The decoder of ATUQ is simply the decoder of CUQ using the dynamic range outputted by the ATUQ encoder.

Note that in order to represent the output of ATUQ for d dimensional inputs, we need a precision of at the most $\lceil \log h \rceil + d \lceil \log(k+1) \rceil$ bits: $\lceil \log h \rceil$ bits to represent the dynamic range and at the most $d \lceil \log(k+1) \rceil$ bits to represent the output of CUQ. The encoder and decoder for AUTQ are given in Algorithms 6 and 7, respectively.

When ATUQ is applied to each subvector in RATQ, each of the $\lceil d/s \rceil$ subvectors are represented using less than $\lceil \log h \rceil + s \lceil \log(k+1) \rceil$ bits. Thus, the overall precision for RATQ is less than[3] $\lceil d/s \rceil \cdot \lceil \log h \rceil +$

---

[2]We assume that d is a power of 2.

[3]log denotes the logarithm to the base 2.

Require: Input $Y \in \mathbb{R}^d$
 1: **if** $\|Y\|_\infty > M_{h-1}$ **then**
        Set $M^* = M_{h-1}$
 2: **else**
        Set $j^* = \min\{j : \|Y\|_\infty \leq M_j\}$, $M^* = M_{j^*}$
 3: Set $Z = Q_u^e(Y; M^*, k)$
 4: **Output:** $Q_{at}^e(Y) = \{Z, j^*\}$

Algorithm 6: Encoder $Q_{at}^e(Y)$ for ATUQ

Require: Input $\{Z, j\}$ with $Z \in \{0, \ldots, k-1, \emptyset\}^d$
    and $j \in \{0, \ldots h-1\}$
 1: **Output:** $Q_{at}^d(Z, j) = Q_u^d(Z; M_j)$

Algorithm 7: Decoder $Q_{at}^d(Z, j)$ for ATUQ

$d\lceil \log(k+1) \rceil$ bits. The decoder of RATQ is simply formed by collecting the output of the ATUQ decoders for all the subvectors to form a d-dimensional vector, and rotating it back using the matrix R^{-1} (the inverse of the rotation matrix used at the encoder).

Choice of parameters. Throughout the remainder of this section, we set our parameters m and h as follows

$$m = 3B^2/d, \quad \log h = \lceil \log(1 + \ln^*(d/3)) \rceil. \quad (6)$$

In particular, this results in $M_{h-1} \geq B$ whereby, for an input Y with $\|Y\|_2^2 \leq B^2$, RATQ outputs an unbiased estimate of Y.

3.3 RATQ in the high-precision regime

Theorem 3.3. *Let $Q_{at,R}$ be the quantizer RATQ with M_js set by (6). Then, for all $s, k \in \mathbb{N}$,*

$$\alpha_0(Q_{at,R}) \leq B\sqrt{\frac{3 + 6s}{(k-1)^2} + 1}, \quad \beta_0(Q_{at,R}) = 0. \quad (7)$$

Thus, α_0 is lower when s is small, but the overall precision needed grows since the number of subvectors increases. The following choice of parameters yields almost optimal performance:

$$s = \log h, \quad \log(k+1) = \lceil \log(2 + \sqrt{3 + 6s}) \rceil. \quad (8)$$

For these choices, we obtain the following.

Corollary 3.4. *The overall precision r used by the quantizer $Q = Q_{at,R}$ with parameters set as in (6), (8) satisfies*

$$r \leq d(1 + \Delta_1) + \Delta_2,$$

where $\Delta_1 = \left\lceil \log\left(2 + \sqrt{3 + 6\lceil \log(1 + \ln^*(d/3)) \rceil}\right) \right\rceil$ and $\Delta_2 = \lceil \log(1 + \ln^*(d/3)) \rceil$.
Furthermore, the optimization protocol π given in Algorithm 1 satisfies $\sup_{(f,O) \in \mathcal{O}_0} \mathcal{E}(f, \pi^{QO}) \leq \sqrt{2}DB/\sqrt{T}$.

Remark 2. The precision requirement in Corollary 3.4 matches the d bit lower bound of Corollary 3.2 upto a multiplicative factor of $O(\log \log \ln^*(d/3))$.

3.4 RATQ in the low-precision regime

Consider the low-precision regime where r is much smaller than d. Our quantizer in this regime adds another layer called *Random Coordinate Sampler* (RCS) to RATQ. RCS requires the encoder and the decoder to share a random set $S \subset [d]$ distributed uniformly over all subsets of $[d]$ of cardinality μd. We use RATQ with $s = 1$, namely the subvectors now consist of one-coordinate each, and obtain the encoded vector Y. The encoder Q_S^e of RCS only retains the coordinates in S and outputs the vector $Q_S^e(Y) := \{Y(i), i \in S\}$. The $Q_S^d(\tilde{Y})$ of RCS, when applied to a vector $\tilde{Y} \in \mathbb{R}^{\mu d}$, outputs $Q_S^d(\tilde{Y}) := \mu^{-1} \sum_{i \in S} \tilde{Y}(i)e_i$, where e_i denotes the ith element of standard basis for $\mathbb{R}^d$. This decoded vector of RCS is then passed through the decoder of RATQ to obtain the final quantized vector. In effect, the decoder of RCS substitutes the value 0 for coordinates not included in S. Note that since we need to retain RATQ encoder output for only μd coordinates, the overall precision of the quantizer is reduced by a factor of μ. Specifically, the composed quantizer satisfies

$$\alpha_0(\tilde{Q}) \leq \frac{\alpha_0(Q_{at,R})}{\sqrt{\mu}} \quad \text{and} \quad \beta_0(\tilde{Q}) = \beta_0(Q_{at,R}).$$

We now set parameter s and k to constants and sample roughly r coordinates. Specifically, we set

$$s = 1, \quad \log(k+1) = 3,$$
$$\mu d = \min\{d, \lfloor r/(3 + \lceil \log(1 + \ln^*(d/3)) \rceil) \rfloor\}. \quad (9)$$

For these choices, we have the following corollary.

Corollary 3.5. *For $r \geq 3 + \lceil \log(1 + \ln^*(d/3)) \rceil$, let Q be the composition of RCS and RATQ with parameters set as in (6), (9). Then, the optimization protocol π in Algorithm 1 satisfies $\sup_{(f,O) \in \mathcal{O}_0} \mathcal{E}(f, \pi^{QO}) \leq \sqrt{2}DB/\sqrt{\mu T}$.*

Remark 3. Note that the convergence rate slows down by a μ specified in (9), which matches the lower bound in Theorem 3.1 upto a multiplicative factor of $O(\log \ln^*(d/3))$

4 Main results for mean square bounded oracles

Moving to oracles satisfying the mean square bounded assumption, we now need to quantize random vectors Y such that $\mathbb{E}[\|Y\|_2^2] \leq B^2$. We take recourse to the

standard *gain-shape* quantization paradigm in vector quantization (*cf.*[11]). Specifically, we separately quantize the *gain* $\|Y\|_2$ and the *shape*[4] $Y/\|Y\|_2$ of Y, and form the estimate of Y by simply multiplying the estimates for the gain and the shape. Note that we already have a good shape quantizer: RATQ. We only need to modify the parameters in (6) to make it work for the unit sphere; we set

$$m = 3/d, \quad \log h = \lceil \log(1 + \ln^*(d/3)) \rceil . \quad (10)$$

We remark that quantizers proposed in most of the prior work can be cast in this gain-shape framework. However, most works simply state that gain is a single parameter which can be quantized using a fixed number of bits; for instance, a single double precision number is prescribed for storing the gain. However, the quantizer is not specified. We carefully analyze this problem and establish lower bounds when a uniform quantizer with a fixed dynamic range is used for quantizing the gain. Further, we present our own quantizer which significantly outperforms the uniform quantization.

4.1 Limitation of uniform gain quantization

We establish lower bounds for a general class of gain-shape quantizers $Q(y) = Q_g(\|y\|_2)Q_s(y/\|y\|_2)$ of precision r that satisfy the following *structural assumptions*:

1. **(Independent gain-shape quantization)** For any given $y \in \mathbb{R}^d$, the output of the gain and the shape quantizers are independent.

2. **(Bounded dynamic-range)** There exists $M > 0$ such that $y \in \mathbb{R}^d$ such that whenever $\|y\|_2 > M$, $Q(y)$ has a fixed distribution $P_\emptyset$.

3. **(Uniformity)** There exists $m \in [M/2^r, M]$ such that for every t in $[0, m]$,

   (a) $\mathrm{supp}(Q_g(t)) \subseteq \{0, m\}$;
   (b) If $P(Q_g(t) = m) > 0$, then

   $$\frac{P(Q_g(t_2) = m)}{P(Q_g(t_1) = m)} \leq \frac{t_2}{t_1}, \quad \forall 0 \leq t_1 \leq t_2 \leq m.$$

The first two assumptions are perhaps clear and hold for a large class of quantizers. The third one is the true limitation and is satisfied by different forms of uniform gain quantizers. For instance, for the one-dimensional version of CUQ with dynamic range $[0, M]$, which is an unbiased, uniform gain quantizer with k_g levels, it holds with $m = M/(k_g - 1)$ (corresponding to the innermost level $[0, M/(k_g-1)]$). It can also be shown to include a

---

[4]For the event $\|Y\|_2 = 0$, we follow the convention that $Y/\|Y\|_2 = e_1$.

deterministic uniform quantizer that rounds-off at the mid-point. The third condition, in essence, captures the unbiasedness requirement that the probability of declaring higher level is proportional to the value. Note that (t_2/t_1) on the right-side can be replaced with any constant multiple of (t_2/t_1).

Below we present lower bounds for performance of any optimization protocol using a gain-shape quantizer that satisfies the assumptions above. We present separate results for high-precision and low-precision regimes, but both are obtained using a general construction that exploits the admissibility of heavy-tail distributions for mean square bounded oracles. This construction is new and may be of independent interest.

Theorem 4.1. *Consider a gain-shape quantizer Q satisfying the assumptions above. Suppose that for $\mathcal{X} = \{x : \|x\|_2 \leq D/2\}$ we can find an optimization protocol π which, using at most T iterations, achieves $\sup_{f,O \in \mathcal{O}} \mathcal{E}(f, \pi^{QO}) \leq \frac{3DB}{\sqrt{T}}$. Then, we can find a universal constant c such that the overall precision r of the quantizer must satisfy*

$$r \geq c(d + \log T).$$

Theorem 4.2. *Consider a gain-shape quantizer Q satisfying the assumptions above . Suppose that the number of bits used by the gain quantizer are fixed independently of T. Then, for $\mathcal{X} = \{x : \|x\|_2 \leq D/2\}$, there exists $(f, O) \in \mathcal{O}$ such that for any optimization protocol π using at most T iterations, we must have*

$$\mathcal{E}(f, \pi^{QO}) \geq \min \frac{cDB}{T^{1/3}},$$

where c is a constant depending only on the number of bits used by the gain quantizer (but not on T).

4.2 A-RATQ in the high precision regime

Instead of quantizing the gain uniformly, we propose to use an adaptive quantizer termed *Adaptive Geometric Uniform Quantizer* (AGUQ) for gain. AGUQ operates similar to the one-dimensional ATUQ, except the possible dynamic-ranges $M_{g,0}, \ldots, M_{g,h}$ grow geometrically (and not using tetra-iterations) as follows:

$$M_{g,j}^2 = B^2 \cdot a_g^j, \quad 0 \leq j \leq h_g - 1. \quad (11)$$

Specifically, for a given gain $G \geq 0$, AGUQ first identifies the smallest j such that $G \leq M_{g,j}$ and then represents G using the one-dimensional version of CUQ with a dynamic range $[0, M_{g,j}]$ and k_g uniform levels

$$B_{M_{g,j},k}(\ell) := \ell \cdot \frac{M_{g,j}}{k_g - 1}, \quad \forall \ell \in \{0, \ldots, k-1\}.$$

As in ATUQ, if $G > M_{h_g-1}$, the overflow $\emptyset$ symbol is used and the decoder simply outputs 0. The overall

procedure is the similar to Algorithms (6) and (7) for $s = 1, h = h_g$, and $M_j = M_{g,j}, 0 \leq j \leq h_g - 1$; the only changes is that now we restrict to nonnegative interval $[0, M_{g,j}]$ for the one-dimensional version of CUQ with uniform levels k_g.

Our overall quantizer, termed the *adaptive-RATQ* (A-RATQ) is given by $Q(Y) := Q_a(\|Y\|_2) \cdot Q_{\mathtt{at},R}(Y/\|Y\|_2)$, where Q_a denotes the one dimensional AGUQ and $Q_{\mathtt{at},R}$ denotes the d-dimensional RATQ. Note that we use independent randomness for $Q_a(\|Y\|_2)$ and $Q_{\mathtt{at},R}(Y/\|Y\|_2)$, rendering them conditionally independent given Y.

The parameters s, k for RATQ and a_g, k_g for AGUQ are yet to be set. We first present a result which holds for all choices of these parameters.

Theorem 4.3. *For Q set to A-RATQ with parameters set as in* (10), (11), *we have*

$$\alpha(Q) \leq \alpha_{s,k} \cdot B\sqrt{\frac{1}{4(k_g-1)^2} + \frac{a_g(h_g-1)}{4(k_g-1)^2} + 1},$$

$$\beta(Q) \leq \frac{B^2}{M_{g,h_g-1}},$$

where $\alpha_{s,k} = \sqrt{\frac{3+6s}{(k-1)^2} + 1}$ is the bound in (7).

Note that RATQ yields an unbiased estimator; the bias in A-RATQ arises from AGUQ since the gain is not bounded. Further, AGUQ uses a precision of $\lceil \log h_g \rceil + \lceil \log(k_g + 1) \rceil$ bits, and therefore, the overall precision of A-RATQ is $\lceil \log h_g \rceil + \lceil \log(k_g + 1) \rceil + \lceil d/s \rceil \lceil \log h \rceil + d \lceil \log(k+1) \rceil$ bits.

In the high-precision regime, we set

$$a_g = 2, \quad \log h_g = \left\lceil \log(1 + \frac{1}{2}\log T) \right\rceil,$$

$$\log(k_g + 1) = \left\lceil \log\left(2 + \frac{1}{2}\sqrt{\log T + 1}\right) \right\rceil. \quad (12)$$

Corollary 4.4. *Denote by Q the quantizer A-RATQ with parameters set as in* (10), (8), *and* (12). *Then, the overall precision r used by Q is less than*

$$d(1 + \Delta_1) + \Delta_2 + \left\lceil \log\left(2 + \sqrt{\log T + 1}\right) \right\rceil,$$

where Δ_1 and Δ_2 are as in Corollary 3.4. Furthermore, the optimization protocol π given in algorithm 1 satisfies $\sup_{(f,O)\in\mathcal{O}} \mathcal{E}(f, \pi^{QO}) \leq 3DB/\sqrt{T}$.

4.3 A-RATQ in the low precision regime

In order to operate with a fixed precision r, we combine A-RATQ with RCS. We simply combine RCS with RATQ as in Section 3.4 to limit the precision. We

divide the total precision r in to r_g and r_s bits: r_g to quantize the gain, r_s to quantize the sub-sampled shape vector. We set

s, k, and μd as in (9), with r_s replacing r,

$$\log h_g = \log(k_g + 1) = \frac{r_g}{2}, \ a_g = (\mu T)^{\frac{1}{h_g+1}} \quad (13)$$

That is, our shape quantizer simply quantizes μd randomly chosen coordinates of the rotated vector using ATUQ with r_s bits, and the remaining bits are used by the gain quantizer AGUQ. The result below shows the performance of this quantizer.

Corollary 4.5. *For any r with gain quantizer being assigned $r_g \geq 4$ bits and shape quantizer being assigned $r_s \geq 3 + \lceil \log(1 + \ln^*(d/3)) \rceil$, let Q be the combination of RCS and A-RATQ with parameters set as in* (10), (11), (13). *Then for $\mu T \geq 1$ and $c = \sqrt{2}$, the optimization protocol π in Algorithm 1 can obtain $\sup_{(f,O)\in\mathcal{O}} \mathcal{E}(f, \pi^{QO})$*

$$\leq O\left(DB\left(\frac{d}{T\min\{d, \frac{r_s}{\log\ln^*(d/3)}\}}\right)^{\frac{1}{2}\cdot\frac{c^{r_g}-1}{c^{r_g}+1}}\right).$$

Remark 4. The precision used in Corollary 4.4 matches the lower bound in Corollary 3.2 upto an additive factor of $\log\log T$ (ignoring the mild factor of $\log\log\ln^*(d/3)$), which is much lower than the $\log T$ lower bound we established for uniform gain quantizers. Our fixed precision quantizer in Corollary 4.5 establishes the tradeoff between a faster convergence rate and reducing the dimension dependence. Specifically, for $d > r$, increasing r_g takes the convergence rate closer to $1/2$, but increases the dimension dependence to $d/(r - r_g)$.

Remark 5. We remark that A-RATQ satisfies assumptions (1) and (2) in Section 4.1 but not (3), and breaches the lower bound in Section 4.1.

5 Distributed mean estimation

Our final set of results discuss the performance of RATQ for distributed mean estimation with limited communication, considered in [23]. Formally, consider n vectors $\{x_i\}_{i=1}^n$ with each x_i in $\mathbb{R}^d$ and vector x_i available to client i. Each client communicates to a fusion center using r bits to enable the center to compute the sample mean $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$.

We measure the performance of a scheme π by the mean square error (MSE) between $\bar{x}$ and the value decoded by the center $\hat{\bar{x}}$, for the fixed set of input vectors $x = \{x_i\}_{i=1}^n$, given by $\mathcal{E}(\pi, x) = \mathbb{E}\left[\|\hat{\bar{x}} - \bar{x}\|_2^2\right]$. We consider the minmax MSE over the unit Euclidean ball B^d given by $\mathcal{E}(\pi, B^d) = \max_{x_i \in S^d, \forall i \in [n]} \mathcal{E}(\pi, x)$. The minimum error attained by variable-length codes is given

by $\mathcal{E}(\Pi_*(r), B^d) = \min_{\pi \in \Pi_*(r)} \max_{x_i \in B^d, \forall i \in [n]} \mathcal{E}(\pi, x)$, where $\Pi_*(r)$ denote a class of communication protocols, with access to public randomness, in which all the clients can encode its input vector upto an expected length of r bits.

The following lower bound is obtained easily from [23, Theorem 5].

Corollary 5.1. *For* $\mathcal{E}(\Pi_*(r), B^d) = O(1/n)$, *we must have* r *to be* $\Omega(nd)$.

The protocol π_{srk} proposed in [23] for this problem achieves $\mathcal{E}(\pi_{srk}, B^d) = O(1/n)$ with $r = \Omega(nd \log \log(d))$. This scheme is uses a quantizer which randomly rotates a input vector, similar to RATQ, before quantizing it uniformly. A simpler quantizer similar to CUQ with a variable-length entropic compression code, denoted by π_{svk}, achieves $\mathcal{E}(\pi_{svk}, B^d) = O(1/n)$ with $r = \Omega(nd)$. This establishes the orderwise optimality of π_{svk}. Thus, prior to our work, the best known fixed-length scheme for distributed mean estimation was π_{srk} which was off from the optimal performance attained by a variable-length code by a factor of $\log \log d$.

We now consider performance of a protocol π_{RATQ} in which RATQ with parameters m, h as in (10) and k, s as in (8) is employed by all the clients, and the center declares the average of the quantized values as its mean estimate.

Theorem 5.2. $\mathcal{E}(\pi_{RATQ}, B^d) = O(1/n)$ *when* $r = n(d(1 + \Delta_1) + \Delta_2)$, *where* Δ_1 *and* Δ_2 *are as in Corollary 3.4.*

Thus, RATQ enjoys the fixed length structure of π_{srk}, while being only $O(\log \log \ln^*(d/3))$ away from the expected length of π_{svk}.

6 Comments on proof techniques

Choice of adaptive dynamic-ranges. The two distinguishing features of RATQ are the grouping of coordinates of the rotated vector and the choice of adaptive dynamic-range parameters M_js. The latter appears in AGUQ as well, but with a different choice of M_js. The first feature of grouping exposes another parameter s, the number of elements in a group. As seen in (7), a small s results in a small α_0, but a larger cost in precision. For the second feature, we provide a high-level justification for our particular choice of M_js for both RATQ and AGUQ below.

Specifically, for the rotated vector RY, denote by $RY_{1,s}$ the first subvector given by $RY_{1,s} := [RY(1), \cdots, RY(s)]^T$. We define the worst-case tail probabilities for the infinity norm of this subvector in the almost sure setting as $p_0(x) := \sup_{Y: \|Y\|_2 \le B \text{ a.s.}} P(\|RY_{1,s}\|_\infty \ge x)$, and similarly,

for AGUQ, we define the worst-case tail probability for the mean-square setting as $p(x) := \sup_{Y \in \mathbb{R}^+ : \mathbb{E}[Y^2] \le B^2} P(Y \ge x)$. We show that for RATQ the worst-case second moment is bounded above by

$$\frac{d}{(k-1)^2} \left(m + \sum_{j=1}^{h-1} p_0(M_{j-1}) \cdot M_j^2 \right) + B^2,$$

and for AGUQ by

$$\frac{1}{4(k_g - 1)^2} \left(m + \sum_{j=1}^{h_g - 1} p(M_{g,j-1}) \cdot M_{g,j}^2 \right) + B^2.$$

These bounds have guided our choice of M_js and $M_{g,j}$s. Specifically, we have selected the parameters $M_{g,j}$s so that the terms $\sum_{j=1}^{h-1} p_0(M_{g,j-1}) \cdot M_{g,j}^2$ and $\sum_{j=1}^{h_g-1} p(M_{g,j-1}) \cdot M_{g,j}^2$ can be bounded appropriately. In particular, when the tail-bounds involved in these terms are sub-gaussian, which is the case for almost surely bounded oracles, we choose M_js as in RATQ, and when they are just quadratically decaying, as are norms of mean square bounded oracles, we choose $M_{g,j}$s as in AGUQ. The same bound can be used also to handle other assumptions for the oracles as well.

Proof sketch for the lower bound in Section 4.1. We consider functions $f_\alpha : \mathbb{R}^d \to \mathbb{R}$, $\alpha \in \{-1, 1\}$, given by $f_\alpha(x) := \delta \frac{B}{\sqrt{2}} |x(1) - \alpha D/2|$, $\alpha \in \{-1, 1\}$. For $x \in \mathbb{R}^d$ with $\|x\|_2 \le D/2$, the gradient of f_α is $\delta \alpha B e_1/\sqrt{2}$, where e_1 is the vector $[1, 0, 0, \ldots, 0]^T$. We consider the oracles O_α, $\alpha \in \{-1, 1\}$, which generate output with distribution $P_\alpha(L) = P_\alpha(-L) = \frac{1-\delta^2}{2}, P_\alpha(\alpha H) = \delta^2$, where the "light element" L equals $\frac{B}{\sqrt{2}} e_1$ and the "heavy element" H equals $\frac{B}{\sqrt{2}\delta^2} e_1$.

Note that when $H \ge M$, the quantizer cannot distinguish between the distributions corresponding to either function, as a consequence of assumption (2). By choosing the largest δ possible under this condition, a standard reduction of convex optimization to hypothesis testing yields $\max_{\alpha \in \{-1,1\}} \mathcal{E}(f, \pi^{QO}) \ge \frac{DB}{4} \min\{\frac{B}{M}, 1\}$.

Next, we consider the case when $L \le m$. In this case, we consider a slightly different family of oracles, parameterized by $y \in [0, 1]$, given by $P_\alpha(L) = P_\alpha(-L) = \frac{1-\delta^{1+y}}{2}, P_\alpha(\alpha H_{\delta,y}) = \delta^{1+y}$, where the heavy elements are given by $H_{\delta,y} = \frac{B}{\sqrt{2}\delta^y} e_1$. If the heavy elements, too, are less than m, then we can use assumption (3) to control the "information" revealed by the gain quantizer output. Thus, the only information possible is by the shape quantizer. The reported bound is obtained by choosing the best $y \in [0, 1]$ satisfying $H_{\delta,y} \le m$, and combining the obtained bound with the previous one.

References

[1] J. Acharya, C. De Sa, D. J. Foster, and K. Sridharan, "Distributed Learning with Sublinear Communication," *arXiv:1902.11259*, 2019.

[2] A. Agarwal, P. L. Bartlett, P. Ravikumar, and M. J. Wainwright, "Information-Theoretic Lower Bounds on the Oracle Complexity of Stochastic Convex Optimization," *IEEE Transactions on Information Theory*, vol. 5, no. 58, pp. 3235–3249, 2012.

[3] A. Agarwal, M. J. Wainwright, P. L. Bartlett, and P. K. Ravikumar, "Information-theoretic lower bounds on the oracle complexity of convex optimization," *Advances in Neural Information Processing Systems*, pp. 1–9, 2009.

[4] N. Agarwal, A. T. Suresh, F. X. X. Yu, S. Kumar, and B. McMahan, "cpSGD: Communication-efficient and differentially-private distributed SGD," *Advances in Neural Information Processing Systems*, pp. 7564–7575, 2018.

[5] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," *Advances in Neural Information Processing Systems*, pp. 1709–1720, 2017.

[6] D. Basu, D. Data, C. Karakus, and S. Diggavi, "Qsparse-local-SGD: Distributed SGD with Quantization, Sparsification, and Local Computations," *arXiv:1906.02367*, 2019.

[7] S. Bubeck, "Convex optimization: Algorithms and complexity," *Foundations and Trends® in Machine Learning*, vol. 8, no. 3-4, pp. 231–357, 2015.

[8] I. Csiszár and P. Narayan, "Capacity of the gaussian arbitrarily varying channel," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 18–26, 1991.

[9] C. M. De Sa, C. Zhang, K. Olukotun, and C. Ré, "Taming the wild: A unified analysis of Hogwild-style algorithms," *Advances in Neural Information Processing Systems*, pp. 2674–2682, 2015.

[10] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and Y. Zhang, "Optimality guarantees for distributed statistical estimation," *arXiv:1405.0782*, 2014.

[11] A. Gersho and R. M. Gray, *Vector quantization and signal compression.* Springer Science & Business Media, 2012, vol. 159.

[12] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," *Proceedings of the International Conference on Machine Learning (ICML' 15)*, pp. 1737–1746, 2015.

[13] K. J. Horadam, *Hadamard matrices and their applications.* Princeton university press, 2012.

[14] X. Hu, L. Prashanth, A. György, and C. Szepesvári, "(Bandit) Convex Optimization with Biased Noisy Gradient Oracles," *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS' 16)*, pp. 819–828, 2016.

[15] B. Hughes and P. Narayan, "Gaussian arbitrarily varying channels," *IEEE Transactions on Information Theory*, vol. 33, no. 2, pp. 267–284, 1987.

[16] S. P. Karimireddy, Q. Rebjock, S. U. Stich, and M. Jaggi, "Error feedback fixes signsgd and other gradient compression schemes," *arXiv:1901.09847*, 2019.

[17] A. Lapidoth, "On the role of mismatch in rate distortion theory," *IEEE Transactions on Information Theory*, vol. 43, no. 1, pp. 38–47, 1997.

[18] A. Nemirovski, "Information-based complexity of convex programming," 1995, Available Online http://www2.isye.gatech.edu/ne-mirovs/Lec_EMCO.pdf.

[19] A. S. Nemirovsky and D. B. Yudin, "Problem complexity and method efficiency in optimization." *Wiley series in Discrete Mathematics and Optimization*, 1983.

[20] A. Ramezani-Kebrya, F. Faghri, and D. M. Roy, "Nuqsgd: Improved communication efficiency for data-parallel sgd via nonuniform quantization," *arXiv preprint arXiv:1908.06077*, 2019.

[21] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 1951.

[22] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," *Advances in Neural Information Processing Systems*, pp. 4447–4458, 2018.

[23] A. T. Suresh, F. X. Yu, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," *Proceedings of the International Conference on Machine Learning (ICML' 17)*, vol. 70, pp. 3329–3337, 2017.

[24] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papail-
    iopoulos, and S. Wright, "Atomo: Communication-
    efficient learning via atomic sparsification," *Ad-
    vances in Neural Information Processing Systems*,
    pp. 9850–9861, 2018.

[25] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen,
    and H. Li, "TernGrad: Ternary gradients to reduce
    communication in distributed deep learning," *Ad-
    vances in Neural Information Processing Systems*,
    pp. 1509–1519, 2017.

[26] A. D. Wyner, "Random packings and coverings
    of the unit n-sphere," *The Bell System Technical
    Journal*, vol. 46, no. 9, pp. 2111–2118, 1967.