RATQ: A Universal Fixed-Length Quantizer for Stochastic Optimization

Prathamesh Mayekar
Indian Institute of Science

Himanshu Tyagi
Indian Institute of Science

Abstract

We present Rotated Adaptive Tetra-iterated Quantizer (RATQ), a fixed-length quantizer for gradients in first order stochastic optimization. RATQ is easy to implement and involves only a Hadamard transform computation and adaptive uniform quantization with appropriately chosen dynamic ranges. For noisy gradients with almost surely bounded Euclidean norms, we establish an information theoretic lower bound for optimization accuracy using finite precision gradients and show that RATQ almost attains this lower bound. For mean square bounded noisy gradients, 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 and is, in fact, close to optimal.

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 Robbins and Monro (1951) and Nemirovsky and Yudin (1983), respectively. Recently, variants of this problem with quantization or communication constraints on oracle output have received a lot of attention Acharya et al. (2019); Agarwal et al. (2018); Alistarh et al. (2017); Basu et al. (2019); De Sa et al. (2015); Gupta et al. (2015); Karimireddy et al. (2019); Ramezani-Kebrya et al. (2019); Stich et al. (2018); Suresh et al. (2017); Wang et al. (2018); Wen et al. (2017). Our work is motivated by results in Alistarh et al. (2017); Suresh et al. (2017), and we elaborate on the connection.

Specifically, Alistarh et al. (2017) considers a problem very similar to ours. The paper Suresh et al. (2017) 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

Proceedings of the 23[rd]International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

The full version of this paper Mayekar and Tyagi (2019) contains the proofs of all results discussed here, as well as other additional details.

(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.* Csiszár and Narayan (1991); Gersho and Gray (2012); Hughes and Narayan (1987); Lapidoth (1997); Wyner (1967)). We propose a new quantizer that is merely a factor $O(\log\log\log\ln^* d)$ far from an optimal information theoretic benchmark which we establish.

In a different direction, for mean square bounded oracles, prior works including Alistarh et al. (2017) 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 the primitive of distributed mean estimation in Section 5.

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.* Bubeck (2015); Nemirovsky and Yudin (1983)). We assume that the function f is convex over the compact, convex domain $\mathcal{X}$ such that $\sup_{x,y\in\mathcal{X}} \|x-y\|_2 \le D$; 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

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

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

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.* Agarwal et al. (2012); Nemirovsky and Yudin (1983)), where we assume for all $x \in \mathcal{X}$

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

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.

Remark 1 (Memoryless, fixed length quantizers). We note that the quantizers in $\mathcal{Q}_r$ are memoryless as well as fixed length quantizers. That is, each new subgradient estimate at time t will be quantized without using any information from the previous updates to a fixed length code of r bits.

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

Before proceeding further, we recall the results for the case $r = \infty$ (*cf.* Agarwal et al. (2009); Nemirovski (1995); Nemirovsky and Yudin (1983)). These bounds will serve as a basic benchmark for our problem.

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

2.2 Quantizer performance for finite precision optimization

Our overall optimization protocol throughout is the *projected SGD* (PSGD) (see Bubeck (2015)). 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.

1: **for** $t = 0$ to $T - 1$ **do**
$\quad x_{t+1} = \Gamma_{\mathcal{X}}\left(x_t - \eta Q(\hat{g}(x_t))\right)$
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 Hu et al. (2016), 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]

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

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

The corresponding quantities for almost surely bounded oracles are

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

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*

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

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

Remark 2 (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 Agarwal et al. (2012), along with the strong data processing inequality in Duchi et al. (2014).

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 $\Omega(d)$.*

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:** $Q^{\mathsf{e}}_{\mathtt{at},R}(Y) = \{Q^{\mathsf{e}}_{\mathtt{at}}(\tilde{Y}_1) \cdots Q^{\mathsf{e}}_{\mathtt{at}}(\tilde{Y}_{\lceil d/s \rceil})\}$

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

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

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

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 $\|Y\|_2^2 \le B^2$, 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 *subvectors* of size s each, and finally quantizes each subvector using a *Coordinate-wise Uniform Quantizer* (CUQ). However, the dynamic-range used for each subvector is chosen adaptively from a set of h *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.

Rotation. 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 Horadam (2012))[2], define $R := \frac{1}{\sqrt{d}} \cdot HD$, where D is a diagonal matrix with each diagonal entry generated uniformly from $\{-1, +1\}$. The input vector Y is multiplied by R in the rotation step. The matrix D can be generated using shared randomness between the encoder and decoder.

Remark 3 (Advantage of random rotation). While by almost sure assumption the input vector to the quantizer is inside the Euclidean ball of radius B, to set the dynamic range[3], we need upper bounds for each coordinate of the vector. After random rotation, each coordinate of the input vector is a centered subgaussian random variable with a variance of $O(B^2/d)$, as opposed to a variance factor of $O(B^2)$, which is all that can be said for the original input vector.

[2]We assume that d is a power of 2.
[3]We mean the interval $[-M, M]$.

Division into subvectors Next, the rotated vector of dimension d is partitioned into $\lceil d/s \rceil$ smaller subvectors. The i^{th} 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[4] $d\,\lceil \log(k+1) \rceil$ 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:** $Q^{\mathsf{e}}_{\mathtt{u}}(Y; M) = Z$

Algorithm 4: Encoder $Q^{\mathsf{e}}_{\mathtt{u}}(Y; M)$ 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:** $Q^{\mathsf{d}}_{\mathtt{u}}(Z; M) = \hat{Y}$

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

ATUQ and Adaptive Quantization of subvectors. The quantizer ATUQ is CUQ with its dynamic-range chosen in an adaptive manner. In order to quantize a particular input vector, it first chooses a dynamic

[4]We denote by $\log(\cdot)$ logarithm to the base 2 and by $\ln(\cdot)$ logarithm to the base e.

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

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

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

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

range from $[-M_i, M_i]$, $1 \le i \le h$. To describe these M_is, we first define the i^{th} 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:

$$M_0^2 = m + m_0, \quad M_i^2 = m \cdot e^{*i} + m_0, \quad \forall\, i \in [h-1].$$

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 most $\lceil \log h \rceil + d \lceil \log(k+1) \rceil$ bits: $\lceil \log h \rceil$ bits to represent the dynamic range and at 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 $\lceil d/s \rceil \cdot \lceil \log h \rceil + d \lceil \log(k+1) \rceil$ 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).

Remark 4 (Mean square error of ATUQ). The per coordinate mean square error between the input to ATUQ and its output roughly grows as

$$O\left(\frac{\sum_{i\in[h]} M_i^2 \cdot p(M_{i-1})}{(k-1)^2}\right), \tag{6}$$

where $p(M)$ is the probability of the ℓ_∞ norm of the input vector exceeding M and k denotes the number of levels of the uniform quantizer. This observation allows us to relate the mean square error to the tail-probabilities of the ℓ_∞ norm of the input vector. In particular, we exploit it to decide on the dimension s of subvectors as well as the growth rate of M_is.

Remark 5 (Growth rate of Tetration). A key distinguishing feature of RATQ is choosing the set of M_is to grow as a tetration, roughly as $M_{i+1} = e^{M_i}$. The large growth rate of a tetration allows us to cover the complete range of each coordinate using only a small number of dynamic ranges, which leads to an unbiased quantizer and reduces the communication. Also, after random rotation, each coordinate of the vector is a centered subgaussian random variable with a variance-parameter of $O(B^2/d)$ (see Remark 3), which, despite the large growth rate of a tetration, ensures that the per coordinate mean square error between the quantized output and the input is almost a constant, as can be seen from (6).

Choice of parameters. Throughout the remainder of this section, we set our parameters m, m_0, and h as follows

$$m = \frac{3B^2}{d}, \quad m_0 = \frac{2B^2 \ln s}{d}, \quad \log h = \lceil \log(1 + \ln^*(d/3)) \rceil. \tag{7}$$

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

3.3 RATQ in the high-precision regime

Theorem 3.3. *Let $Q_{\mathtt{at},R}$ be the quantizer RATQ with M_js set by (7). Then, for all $s, k \in \mathbb{N}$,*

$$\alpha_0(Q_{\mathtt{at},R}) \le B\sqrt{\frac{9 + 3\ln s}{(k-1)^2} + 1}, \quad \beta_0(Q_{\mathtt{at},R}) = 0. \tag{8}$$

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:

$$s = \log h, \quad \log(k+1) = \left\lceil \log(2 + \sqrt{9 + 3\ln s}) \right\rceil. \tag{9}$$

For these choices, we obtain the following.

Corollary 3.4. *The overall precision r used by the quantizer $Q = Q_{\mathtt{at},R}$ with parameters set as in (7), (9) satisfies*

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

where $\Delta_1 = \lceil \log\left(2 + \sqrt{9 + 3\ln \Delta_2}\right) \rceil$ and $\Delta_2 = \lceil \log(1 + \ln^*(d/3)) \rceil$.

Furthermore, the optimization protocol π given in Algorithm 1 satisfies[5] $\sup_{(f,O)\in\mathcal{O}_0} \mathcal{E}(f,\pi^{QO}) \le \sqrt{2}DB/\sqrt{T}$.

Remark 6. The precision requirement in Corollary 3.4 matches the d bit lower bound of Corollary 3.2 upto a multiplicative factor of $O(\log\log\log\ln^*(d/3))$.

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 $Q_S^{\mathsf{e}}(Y) := \{Y(i), i \in S\}$. The $Q_S^{\mathsf{d}}(\tilde{Y})$ of RCS, when applied to a vector $\tilde{Y} \in \mathbb{R}^{\mu d}$, outputs $Q_S^{\mathsf{d}}(\tilde{Y}) := \mu^{-1}\sum_{i\in S}\tilde{Y}(i)e_i$, 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

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

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

$$\begin{aligned} s = 1, \quad \log(k+1) = 3, \\ \mu d = \min\{d, \lfloor r/(3 + \lceil \log(1 + \ln^*(d/3))\rceil)\rfloor\}. \end{aligned} \tag{10}$$

For these choices, we have the following corollary.

Corollary 3.5. *For $r \ge 3 + \lceil\log(1+\ln^*(d/3))\rceil$, let Q be the composition of RCS and RATQ with parameters set as in (7), (10). Then, the optimization protocol π in Algorithm 1 satisfies* $\sup_{(f,O)\in\mathcal{O}_0} \mathcal{E}(f,\pi^{QO}) \le \sqrt{2}DB/\sqrt{\mu T}$.

Remark 7. Note that the convergence rate slows down by a μ specified in (10), which matches the lower bound in Theorem 3.1 upto a multiplicative factor of $O(\log\ln^*(d/3))$

[5] Note that all our quantizers use independent randomness across iterations.

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 $\mathbb{E}\left[\|Y\|_2^2\right] \le B^2$. We take recourse to the standard *gain-shape* quantization paradigm in vector quantization (*cf.*Gersho and Gray (2012)). Specifically, we separately quantize the *gain* $\|Y\|_2$ and the *shape*[6] $Y/\|Y\|_2$ 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 (7) to make it work for the unit sphere; we set

$$\begin{aligned} m = \frac{3}{d}, \quad m_0 = \frac{2\ln s}{d}, \\ \log h = \lceil \log(1 + \ln^*(d/3))\rceil\,. \end{aligned} \tag{11}$$

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[7].
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) $\mathtt{supp}(Q_g(t)) \subseteq \{0, m\}$;
   (b) If $P(Q_g(t) = m) > 0$, then $\dfrac{P(Q_g(t_2) = m)}{P(Q_g(t_1) = m)} \le \dfrac{t_2}{t_1}, \quad \forall\, 0 \le t_1 \le t_2 \le m.$

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

[7] This assumption implies that for a random input vector Y, conditioned on Y, the output of shape and gain quantizers are independent.

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

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

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 $\mathcal{X} = \{x : \|x\|_2 \le D/2\}$, there exists $(f, O) \in \mathcal{O}$ such that for any optimization protocol π using at most T iterations, we must have*

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

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:

$$M_{g,j}^2 = B^2 \cdot a_g^j, \quad 0 \le j \le h_g - 1. \tag{12}$$

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

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

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 $M_j = M_{g,j}$, $0 \le j \le h_g - 1$; 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 as described above.

Remark 8. Note that the mean square error corresponding to AGUQ is similar to one given in (6). But this time we cannot use a tetration for selecting possible dynamic-ranges $M_{g,i}$s, since gain need not be subgaussian. We now only have tail-probability bounds determined by the Markov inequality (heavy-tails) and can only increase $M_{g,i}$s geometrically

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 (11), (12), we have*

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

where $\alpha_{s,k} = \sqrt{\frac{9+3\ln s}{(k-1)^2} + 1}$ is the bound in (8).

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

$$a_g = 2, \quad \log h_g = \left\lceil \log(1 + \frac{1}{2}\log T) \right\rceil,$$
$$\log(k_g + 1) = \left\lceil \log\left(2 + \frac{1}{2}\sqrt{\log T + 1}\right) \right\rceil. \tag{13}$$

Corollary 4.4. *Denote by Q the quantizer A-RATQ with parameters set as in (11), (9), and (13). 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 $\sup_{(f,O)\in\mathcal{O}} \mathcal{E}(f,\pi^{QO}) \le 3DB/\sqrt{T}$.

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

$$\begin{aligned} &s, k, \text{ and } \mu d \text{ as in (10), with } r_s \text{ replacing } r,\\ &\log h_g = \log(k_g+1) = \frac{r_g}{2},\ a_g = (\mu T)^{\frac{1}{h_g+1}} \end{aligned} \tag{14}$$

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 $r_g \ge 4$ bits and shape quantizer being assigned $r_s \ge 3+\lceil \log(1+\ln^*(d/3))\rceil$, let Q be the combination of RCS and A-RATQ with parameters set as in (11), (12), (14). Then for $\mu T \ge 1$, the optimization protocol π in Algorithm 1 can obtain*

$$\sup_{(f,O)\in\mathcal{O}} \mathcal{E}(f,\pi^{QO}) \le O\left(DB\left(\frac{d}{T\min\{d,\frac{r_s}{\log\ln^*(d/3)}\}}\right)^{\frac{1}{2}\cdot\frac{2^{r_g/2}-1}{2^{r_g/2}+1}}\right).$$

Remark 9. 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 d dependence), 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 that using only a constant number of bits for gain-quantization, we get very close to the lower bound in Theorem 3.1. For instance, given access to a large enough precision r, if we set r_g to be 16 bits, we get

$$\sup_{(f,O)\in\mathcal{O}} \mathcal{E}(f,\pi^{QO}) \lesssim O\left(DB\left(\frac{d}{T\min\{d,r\}}\right)^{\frac{1}{2}\cdot\frac{255}{257}}\right).$$

Remark 10. 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 the primitive of distributed mean estimation with limited communication, considered in Suresh et al. (2017). 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 $\mathcal{E}(\pi,B^d)=\max_{x_i\in B^d,\forall i\in[n]}\mathcal{E}(\pi,x)$. 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 (Suresh et al., 2017, 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 Suresh et al. (2017) for this problem achieves $\mathcal{E}(\pi_{srk},B^d)=O(1/n)$ with $r=\Omega(nd\log\log(d))$. This scheme 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 (11) and k, s as in (9) 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 $r=n\left(d(1+\Delta_1)+\Delta_2\right)$, 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 $O(\log\log\log\ln^*(d/3))$ away from the expected length of π_{svk}.

Acknowledgement

Prathamesh Mayekar is supported by a Ph.D. fellowship from Wipro Limited. Himanshu Tyagi is supported by a grant from Robert Bosch Center for Cyberphysical Systems (RBCCPS), Indian Institute of Science, Bangalore and the grant EMR/2016/002569 from the Department of Science and Technology (DST), India.

References

Acharya, J., De Sa, C., Foster, D. J., and Sridharan, K. (2019). Distributed Learning with Sublinear Communication. *arXiv:1902.11259*.

Agarwal, A., Bartlett, P. L., Ravikumar, P., and Wainwright, M. J. (2012). Information-Theoretic Lower Bounds on the Oracle Complexity of Stochastic Convex Optimization. *IEEE Transactions on Information Theory*, 5(58):3235–3249.

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

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

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

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

Bubeck, S. (2015). Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357.

Csiszár, I. and Narayan, P. (1991). Capacity of the gaussian arbitrarily varying channel. *IEEE Transactions on Information Theory*, 37(1):18–26.

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

Duchi, J. C., Jordan, M. I., Wainwright, M. J., and Zhang, Y. (2014). Optimality guarantees for distributed statistical estimation. *arXiv:1405.0782*.

Gersho, A. and Gray, R. M. (2012). *Vector quantization and signal compression*, volume 159. Springer Science & Business Media.

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

Horadam, K. J. (2012). *Hadamard matrices and their applications*.Princeton university press.

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

Hughes, B. and Narayan, P. (1987). Gaussian arbitrarily varying channels. *IEEE Transactions on Information Theory*, 33(2):267–284.

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

Lapidoth, A. (1997). On the role of mismatch in rate distortion theory. *IEEE Transactions on Information Theory*, 43(1):38–47.

Mayekar, P. and Tyagi, H. (2019). RATQ: A universal fixed-length quantizer for stochastic optimization. *arXiv:1908.08200*.

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

Nemirovsky, A. S. and Yudin, D. B. (1983). Problem complexity and method efficiency in optimization. *Wiley series in Discrete Mathematics and Optimization*.

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

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22:400–407.

Stich, S. U., Cordonnier, J.-B., and Jaggi, M. (2018). Sparsified SGD with memory. *Advances in Neural Information Processing Systems*, pages 4447–4458.

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

Wang, H., Sievert, S., Liu, S., Charles, Z., Papailiopoulos, D., and Wright, S. (2018). Atomo: Communication-efficient learning via atomic sparsification. *Advances in Neural Information Processing Systems*, pages 9850–9861.

Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., and Li, H. (2017). TernGrad: Ternary gradients to reduce communication in distributed deep learning. *Advances in Neural Information Processing Systems*, pages 1509–1519.

Wyner, A. D. (1967). Random packings and coverings of the unit n-sphere. *The Bell System Technical Journal*, 46(9):2111–2118.