Atomo: Communication-efficient Learning via Atomic Sparsification

Hongyi Wang[*1], Scott Sievert[*2], Zachary Charles[2], Dimitris Papailiopoulos[2], and Stephen J. Wright[1]

[1]Department of Computer Sciences, UW–Madison
[2]Department of Electrical and Computer Engineering, UW–Madison

June 7, 2018

Abstract

Distributed model training suffers from communication overheads due to frequent gradient updates transmitted between compute nodes. To mitigate these overheads, several studies propose the use of sparsified stochastic gradients. We argue that these are facets of a general sparsification method that can operate on any possible *atomic decomposition*. Notable examples include element-wise, singular value, and Fourier decompositions. We present Atomo, a general framework for atomic sparsification of stochastic gradients. Given a gradient, an atomic decomposition, and a sparsity budget, Atomo gives a random unbiased sparsification of the atoms minimizing variance. We show that methods such as QSGD and TernGrad are special cases of Atomo and show that sparsifiying gradients in their singular value decomposition (SVD), rather than the coordinate-wise one, can lead to significantly faster distributed training.

1   Introduction

Distributed computing systems have become vital to the success of modern machine learning systems. Work in parallel and distributed optimization has shown that these systems can obtain massive speed up gains in both convex and non-convex settings [9, 18, 19, 26, 34, 41]. Several machine learning frameworks such as TensorFlow [1], MXNet [10], and Caffe2 [27], come with distributed implementations of popular training algorithms, such as mini-batch SGD. However, the empirical speed-up gains offered by distributed training, often fall short of the optimal linear scaling one would hope for. It is now widely acknowledged that communication overheads are the main source of this speedup saturation phenomenon [18, 45, 47, 39, 22].

Communication bottlenecks are largely attributed to frequent gradient updates transmitted between compute nodes. As the number of parameters in state-of-the-art models scales to hundreds of millions [23, 24], the size of gradients scales proportionally. These bottlenecks become even more pronounced in the context of federated learning [36, 29], where edge devices (*e.g.*, mobile phones, sensors, etc) perform decentralized training, but suffer from low-bandwidth during up-link.

---

[*]These authors contributed equally.

To reduce the cost of of communication during distributed model training, a series of recent studies propose communicating low-precision or sparsified versions of the computed gradients during model updates. Partially initiated by a 1-bit implementation of SGD by Microsoft in [45], a large number of recent studies revisited the idea of low-precision training as a means to reduce communication [17, 4, 56, 51, 14, 55, 40, 14, 15, 6]. Other approaches for low-communication training focus on sparsification of gradients, either by thresholding small entries or by random sampling [47, 35, 32, 2, 33, 8, 43, 49]. Several approaches, including QSGD and TernGrad, implicitly combine quantization and sparsification to maximize performance gains [4, 51, 29, 30, 48], while providing provable guarantees for convergence and performance. We note that quantization methods in the context of gradient based updates have a rich history, dating back to at least as early as the 1970s [21, 3, 5].

Our Contributions   An atomic decomposition represents a vector as a weighted sum of pair-wise orthonormal vectors (so-called atoms), in an inner product space. In this work, we show that stochastic gradient sparsification and quantization are facets of a general approach that sparsifies a gradient on any of its possible atomic decompositions, *e.g.*, its entry-wise or singular value decomposition, its Fourier representation, etc. With this in mind, we develop ATOMO, a general framework for atomic sparsification of stochastic gradients. ATOMO sets up and optimally solves a meta-optimization that minimizes the variance of the sparsified gradient, subject to the constraints that it is sparse on the atomic basis, and also is an unbiased estimator of the input.

We show that 1-bit QSGD and TernGrad are in fact special cases of ATOMO, and each is optimal (in terms of variance and sparsity), in different parameter regimes. Then, we argue that for some neural network applications, viewing the gradient as a concatenation of matrices (each corresponding to a layer), and applying atomic sparsification to their SVD is meaningful and well-motivated by the fact that these matrices are "nearly" low-rank, *e.g.*, see Fig. 1. We show that ATOMO on the SVD of each layer's gradient, can lead to less variance, and faster training, for the same communication budget as that of QSGD or TernGrad. We present extensive experiments showing that using ATOMO with SVD sparsification, can lead to up to $2\times$ faster training time (including the time to compute the SVD) compared to QSGD, on VGG and ResNet-18, and SVHN and CIFAR-10.

Relation to Prior Work   ATOMO is closely related to work on communication-efficient distributed mean estimation in [30] and [48]. These works both note, as we do, that variance (or equivalently the mean squared error) controls important quantities such as convergence, and they seek to find a low-communication vector averaging scheme that minimizes it. Our work differs in two key aspects. First, we derive a closed-form solution to the variance minimization problem for all input gradients. Second, ATOMO applies to any atomic decomposition, which allows us to compare entry-wise against singular value sparsification for matrices. Using this, we derive explicit conditions for which SVD sparsification leads to lower variance for the same sparsity budget.

The idea of viewing gradient sparsification through a meta-optimization lens was also used in [50]. Our work differs in two key ways. First, [50] consider the problem of minimizing the sparsity of a gradient for a fixed variance, while we consider the reverse problem, that is, minimizing the variance subject to a sparsity budget. The second more important difference is that while [50] focuses on entry-wise sparsification, we consider a general problem where we sparsify according to any atomic decomposition. For instance, our approach directly applies to sparsifying the singular values of a matrix, which gives rise to faster training algorithms.
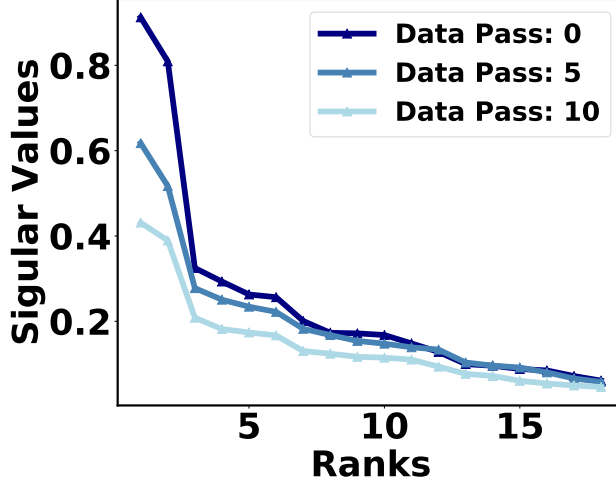
Figure 1: The singular values of a convolutional layer's gradient, for ResNet-18 while training on CIFAR-10. The gradient of a layer can be seen as a matrix, once we vectorize and appropriately stack the convolutional filters. For all presented data passes, there is a sharp decay in singular values, with the top 3 standing out.

Finally, low-rank factorizations and sketches of the gradients when viewed as matrices were proposed in [53, 44, 25, 52, 29]; arguably most of these methods (with the exception of [29]) aimed to address the high flops required when training low-rank models. Though they did not directly aim to reduce communication, this arises as a useful side effect.

2   Problem Setup

In machine learning, we often wish to find a model w minimizing the *empirical risk*

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(w; x_i) \tag{1}$$

where $x_i \in \mathbb{R}^d$ is the i-th data point. One way to approximately minimize $f(w)$ is by using stochastic gradient methods that operate as follows:

$$w_{k+1} = w_k - \gamma \widehat{g}(w_k)$$

where w_0 is some initial model, γ is the step size, and $\widehat{g}(w)$ is a stochastic gradient of $f(w)$, *i.e.*it is an unbiased estimate of the true gradient $g(w) = \nabla f(w)$. Mini-batch SGD, one of the most common algorithms for distributed training, computes $\widehat{g}$ as an average of B gradients, each evaluated on randomly sampled data from the training set. Mini-batch SGD is easily parallelized in the parameter server (PS) setup, where a PS stores the global model, and P compute nodes split the effort of computing the B gradients. Once the PS receives these gradients, it applies them to the model, and sends it back to the compute nodes.

To prove convergence bounds for stochastic-gradient based methods, we usually require $\widehat{g}(w)$ to be an unbiased estimator of the full-batch gradient, and to have small second moment $\mathbb{E}\|\widehat{g}(w)\|^2$, as

3

this controls the speed of convergence. To see this, suppose w^* is a critical point of f, then we have

$$\mathbb{E}[\|w_{k+1} - w^*\|_2^2] = \mathbb{E}[\|w_k - w^*\|_2^2] - \underbrace{\left(2\gamma\langle\nabla f(w_k), w_k - w_*\rangle - \gamma^2\mathbb{E}[\|\widehat{g}(w_k)\|_2^2]\right)}_{\text{progress at step } t}.$$

Thus, the progress of the algorithm at a single step is, in expectation, controlled by the term $\mathbb{E}[\|\widehat{g}(w_k)\|_2^2]$; the smaller it is, the bigger the progress. This is a well-known fact in optimization, and most convergence bounds for stochastic-gradient based methods, including mini-batch, involve upper bounds on $\mathbb{E}[\|\widehat{g}(w_k)\|_2^2]$ in convex and nonconvex settings [11, 20, 41, 7, 7, 16, 42, 28, 13, 54]. In short, recent results on low-communication variants of SGD design unbiased quantized or sparse gradients, and try to minimize their variance [4, 30, 50]. Note that when we require the estimate to be unbiased, minimizing the variance is equivalent to minimizing the second moment.

Since variance is a proxy for speed of convergence, in the context of communication-efficient stochastic gradient methods, one can ask: *What is the smallest possible variance of an unbiased stochastic gradient that can be represented with k bits?* Note that under the unbiased assumption, minimizing variance is equivalent to minimizing the second moment of the random vector. This meta-optimization can be cast as the following meta-optimization:

$$\min_g \mathbb{E}\|\widehat{g}(w)\|^2$$
$$\text{s.t. } \mathbb{E}[\widehat{g}(w)] = g(w)$$
$$\widehat{g}(w) \text{ can be expressed with } k \text{ bits}$$

Here, the expectation is taken over the randomness of $\widehat{g}$. We are interested in designing a stochastic approximation $\widehat{g}$ that "solves" this optimization. However, it seems difficult to design a formal, tractable version of the last constraint. In the next section, we replace this with a simpler constraint that instead requires that $\widehat{g}(w)$ is sparse with respect to a given atomic decomposition.

3   ATOMO: Atomic Decomposition and Sparsification

Suppose that V is an inner product space over $\mathbb{R}$. We will let $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$ denote the inner product and induced norm on V.

Definition 1. *Suppose $g = \sum_{i=1}^n \lambda_i a_i$ where $\lambda_i \neq 0$. We say that this decomposition is atomic if $\{a_i\}_{i=1}^n$ is an orthonormal set of vectors.*

In what follows, you may think of g as the stochastic gradient. One example of an atomic decomposition is the entry-wise decomposition of a vector $g \in \mathbb{R}^n$ as $g = \sum_i g_i e_i$ where $\{e_i\}_{i=1}^n$ is the standard basis. Similarly, any orthonormal basis $\{b_i\}_{i=1}^n$ of $\mathbb{R}^n$ gives rise to an atomic decomposition of any $g \in \mathbb{R}^n$. Another important example is the singular value decomposition of a matrix $X = \sum_{i=1}^r \sigma_i u_i v_i^T$, as one can verify that $\{u_i v_i^T\}_{i=1}^r$ form an orthonormal set of matrices under the trace inner product. We will focus on sparsification of the SVD in the following sections.

In general, we are interested in sparsifying g, *i.e.*, finding a sparsified, decomposition that consists of fewer atoms. To relax the optimization stated in the previous section, we would like g to be expressed by s atoms. Our primary motivation is that to communicate g, from say a compute node to a PS, we have to communicate each element of g individually. We can sparsify g in whatever decomposition is most beneficial, and send the weighted atoms of that instead. For instance, if a

matrix X is low rank then its singular value decomposition is naturally sparse. Therefore, it may be beneficial to sparsify on the level of singular values instead of sparsifying the entries of X.

Given some atomic decomposition of $g = \sum_{i=1}^{n} \lambda_i a_i$, we wish to compute an estimator $\widehat{g}$ that is sparse, unbiased, and such that $\mathbb{E}[\|\widehat{g}\|^2]$ is small. Note that because $\widehat{g}$ is unbiased, minimizing the second moment $\mathbb{E}[\|\widehat{g}\|^2]$ is equivalent to minimizing the variance $\mathbb{E}[\|\widehat{g} - g\|^2]$. We will use the following sparsification method:

$$\widehat{g} = \sum_{i=1}^{n} \frac{\lambda_i t_i}{p_i} a_i \tag{2}$$

where $t_i \sim \text{Bernoulli}(p_i)$, for $0 < p_i \leq 1$. We refer to this sparsification scheme as *atomic sparsification*. Note that the t_i's are independent. We have the following lemma about $\widehat{g}$.

Lemma 1. *If $g = \sum_{i=1}^{n} \lambda_i a_i$ is an atomic decomposition then $\mathbb{E}[\widehat{g}] = g$ and $\mathbb{E}[\|\widehat{g}\|^2] = \sum_{i=1}^{n} \lambda_i^2 / p_i$.*

Let $\lambda = [\lambda_1, \ldots, \lambda_n]^T$, $p = [p_1, \ldots, p_n]^T$. In order to ensure that this estimator is sparse, we fix some *sparsity budget s*. That is, we require $\sum_i p_i = s$; note that this a *sparsity on average* constraint. We wish to minimize $\mathbb{E}[\|\widehat{g}\|^2]$ subject to this constraint. In other words, we wish to solve the following problem:

$$\min_p \sum_{i=1}^{n} \frac{\lambda_i^2}{p_i} \quad \text{subject to } 0 < p_i \leq 1, \quad \sum_{i=1}^{n} p_i = s. \tag{3}$$

An equivalent form of this optimization problem was previously presented in [30] (Section 6.1). The authors considered this problem for entry-wise sparsification and found a closed-form solution for $s \leq \|\lambda\|_1 / \|\lambda\|_\infty$. We give a version of their result but extend this to a closed-form solution for all s. A similar optimization problem was given in [50], which instead minimizes sparsity subject to a variance constraint.

Algorithm 1: ATOMO probabilities

> **Input** : $\lambda \in \mathbb{R}^n$ with $|\lambda_1| \geq \ldots |\lambda_n|$; sparsity budget s such that $0 < s \leq n$.
> **Output:** $p \in \mathbb{R}^n$ solving (3).
> $i = 1$;
> **while** $i \leq n$ **do**
> > **if** $|\lambda_i| s \leq \sum_{j=i}^{n} |\lambda_i|$ **then**
> > > **for** $k = i, \ldots, n$ **do**
> > > > $p_k = |\lambda_k| s \left(\sum_{j=i}^{n} |\lambda_i| \right)^{-1}$;
> > >
> > > **end**
> > > $i = n + 1$;
> >
> > **else**
> > > $p_i = 1, s = s - 1$;
> > > $i = i + 1$;
> >
> > **end**
>
> **end**

We will show that the Algorithm 1 produces a probability vector $p \in \mathbb{R}^n$ solving (3) for $0 < s \leq n$. Note that (3) has a non-empty feasible set only for $0 < s \leq n$. Define $f(p) := \sum_{i=1}^{n} \lambda_i^2 / p_i$. To

understand how to solve (3), we first consider the following relaxation:

$$\min_{p} \sum_{i=1}^{n} \frac{\lambda_i^2}{p_i} \quad \text{subject to} \quad 0 < p_i, \quad \sum_{i=1}^{n} p_i = s. \tag{4}$$

We have the following lemma about the solutions to (4), first shown in [30].

Lemma 2 ([30]). *Any feasible vector p to (4) satisfies $f(p) \geq \frac{1}{s} \|\lambda\|_1^2$. This is achieved iff*

$$p_i = \frac{|\lambda_i| s}{\|\lambda\|_1}. \tag{5}$$

Lemma 2 implies that if we ignore the constraint that $p_i \leq 1$, then the optimal p is achieved by setting $p_i = |\lambda_i| s / \|\lambda\|_1$. If the quantity in the right-hand side is greater than 1, this does not give us an actual probability. This leads to the following definition.

Definition 2. *An atomic decomposition $g = \sum_{i=1}^{n} \lambda_i a_i$ is s-unbalanced at entry i if $|\lambda_i| s > \|\lambda\|_1$.*

Fix the atomic decomposition of g. If there are no s-unbalanced entries then we say that the g is s-balanced. We have the following lemma which guarantees that g is s-balanced for s not too large.

Lemma 3. *An atomic decomposition $g = \sum_{i=1}^{n} \lambda_i a_i$ is s-balanced iff $s \leq \|\lambda\|_1 / \|\lambda\|_\infty$.*

Lemma 2 gives us the optimal way to sparsify s-balanced vectors, since the p that is optimal for (4) is feasible for (3). Moreover, the iff condition in Lemma 2 implies that the optimal assignment of the p_i are between 0 and 1 iff v is s-balanced. Suppose now that g is s-unbalanced at entry j. We cannot assign p_j as in (5). We will show that setting $p_j = 1$ is optimal in this setting. This comes from the following lemma.

Lemma 4. *Suppose that g is s-unbalanced at entry j and that q is feasible in (3). Then $\exists p$ that is feasible in (3) such that $f(p) \leq f(q)$ and $p_j = 1$.*

We therefore derive the following theorem characterizing solutions to (3).

Theorem 5. *Suppose we sparsify g as in (2) with sparsity budget s.*

1. *If g is s-balanced, then*

$$\mathbb{E}[\|\widehat{g}\|^2] \geq \frac{1}{s} \|\lambda\|_1^2$$

with equality if and only if $p_i = |\lambda_i| s / \|\lambda\|_1$.

2. *If g is s-unbalanced, then*

$$\mathbb{E}[\|\widehat{g}\|^2] > \frac{1}{s} \|\lambda\|_1^2$$

and is minimized by p with $p_j = 1$ where $j = \operatorname{argmax}_{i=1,\ldots,n} |\lambda_i|$.

This theorem implies that Algorithm 1 produces a vector $p \in \mathbb{R}^n$ solving (3). Note that due to the sorting requirement in the input, the algorithm requires $O(n \log n)$ operations. As we discuss in Appendix C, we could instead do this in $O(sn)$ operations by, instead of sorting and iterating through the values in order, simply selecting the next unvisited index i maximizing $|\lambda_i|$ and performing the same test/updates. As we show in Appendix C, we need to select at most s indices before the if statement in Algorithm 1 holds. Whether to sort or do selection depends on the size of s relative to $\log n$.

4 Relation to QSGD and TernGrad

In this section, we will discuss how ATOMO is related to two recent quantization schemes, 1-bit QSGD [4] and TernGrad [51]. We will show that in certain cases, these schemes are versions of the ATOMO for a specific sparsity budget s. Both schemes use the entry-wise atomic decomposition.

4.1 1-bit QSGD

QSGD takes as input $g \in \mathbb{R}^n$ and $b \geq 1$. This b governs the number of quantization buckets. When $b = 1$, this is referred to as 1-bit QSGD. 1-bit QSGD produces a random vector $Q(g)$ defined by

$$Q(g)_i = \|g\|_2 \operatorname{sign}(g_i)\zeta_i.$$

Here, the $\zeta_i \sim \text{Bernoulli}(|g_i|/\|g\|_2)$ are independent random variables. A straightforward computation shows that $Q(g)$ can be defined equivalently by

$$Q(g)_i = \frac{g_i t_i}{|g_i|/\|g\|_2} \tag{6}$$

where $t_i \sim \text{Bernoulli}(|g_i|/\|g\|_2)$. Therefore, 1-bit QSGD exactly uses the atomic sparsification framework in (2) with $p_i = |g_i|/\|g\|_2$. The total sparsity budget is therefore given by

$$s = \sum_{i=1}^{n} p_i = \frac{\|g\|_1}{\|g\|_2}.$$

By Lemma 3 any g is s-balanced for this s. Therefore, Theorem 5 implies that the optimal way to assign p_i with this given s is $p_i = |g_i|/\|g\|_2$. Since this agrees with (6), this implies that 1-bit QSGD performs variance-optimal entry-wise sparsification for sparsity budget $s = \|g\|_1/\|g\|_2$.

4.2 TernGrad

Similarly, TernGrad takes as input $g \in \mathbb{R}^n$, and produces a sparsified version $T(g)$ given by

$$T(g)_i = \|g\|_\infty \operatorname{sign}(g_i)\zeta_i$$

where $\zeta_i \sim \text{Bernoulli}(|g_i|/\|g\|_\infty)$. A straightforward computation shows that $T(g)$ can be defined equivalently by

$$T(g)_i = \frac{g_i t_i}{|g_i|/\|g\|_\infty} \tag{7}$$

where $t_i \sim \text{Bernoulli}(|g_i|/\|g\|_\infty)$. Therefore, TernGrad exactly uses the atomic sparsification framework in (2) with $p_i = |g_i|/\|g\|_\infty$. The total sparsity budget is given by

$$s = \sum_{i=1}^{n} p_i = \frac{\|g\|_1}{\|g\|_\infty}.$$

By Lemma 3, any g is s-balanced for this s. Therefore, Theorem 5 implies that the optimal way to assign p_i with this given s is $p_i = |g_i|/\|g\|_\infty$. This agrees with (7). Therefore, TernGrad performs variance-optimal entry-wise sparsification for sparsity budget $s = \|g\|_1/\|g\|_\infty$.

4.3 ℓ_q-quantization

We can generalize both of these with the following quantization method, which we refer to as ℓ_q-*quantization*. Fix $q \in (0, \infty]$. Given $g \in \mathbb{R}^n$, we define the ℓ_q-quantization of g, denoted $L_q(g)$ by

$$L_q(v)_i = \|g\|_q \operatorname{sign}(g_i)\zeta_i$$

where $\zeta_i \sim \text{Bernoulli}(|g_i|/\|g\|_q)$. Note that for all i, $|g_i| \le \|g\|_\infty \le \|g\|_q$, so this does give us a valid probability. We can define $L_q(v)$ equivalently by

$$L_q(g)_i = \frac{g_i t_i}{|g_i|/\|g\|_q} \tag{8}$$

where $t_i \sim \text{Bernoulli}(|g_i|/\|g\|_q)$. Therefore, ℓ_q-quantization exactly uses the atomic sparsification framework in (2) with $p_i = |g_i|/\|g\|_q$. The total sparsity budget is therefore

$$s = \sum_{i=1}^{n} p_i = \frac{\|g\|_1}{\|g\|_q}.$$

By Lemma 3, the optimal way to assign p_i with this given s is $p_i = |g_i|/\|g\|_q$. Since this agrees with (8), Theorem 5 implies the following theorem.

Theorem 6. ℓ_q-*quantization performs atomic sparsification in the standard basis with* $p_i = |g_i|/\|g\|_q$. *This solves* (3) *for* $s = \|g\|_1/\|g\|_q$ *and satisfies* $\mathbb{E}[\|L_q(g)\|_2^2] = \|g\|_1\|g\|_q$.

In particular, for $q = 2$ we get 1-bit QSGD while for $q = \infty$, we get TernGrad. This shows strong connections between quantization and sparsification methods. Note that as q increases, the sparsity budget for ℓ_q-quantization increases while the variance decreases. The choice of whether to set $q = 2$, $q = \infty$, or q to something else is therefore dependent on the total expected sparsity one is willing to tolerate, and can be tuned for different distributed scenarios.

5 Spectral-ATOMO: Sparsifying the Singular Value Decomposition

In this section we compare different methods for matrix sparsification. The first uses ATOMO on the entry-wise decomposition of a matrix, and the second uses ATOMO on the singular value decomposition (SVD) of a matrix. We refer to this second approach as Spectral-ATOMO. We show that under concrete conditions, Spectral-ATOMO incurs less variance than sparsifying entry-wise. We present these conditions and connect them to the equivalence of certain matrix norms.

Notation For a rank r matrix X, denote its singular value decomposition by

$$X = \sum_{i=1}^{r} \sigma_i u_i v_i^T.$$

Let $\sigma = [\sigma_1, \ldots, \sigma_r]^T$. Taking the ℓ_p norm of σ gives a norm on X, referred to as the Schatten p-norm. For $p = 1$, we get the spectral norm $\|\cdot\|_*$, for $p = 2$ we get the Frobenius norm $\|\cdot\|_F$, and for $p = \infty$, we get the spectral norm $\|\cdot\|_2$. We define the $\ell_{p,q}$ norm of a matrix X by

$$\|X\|_{p,q} = \left(\sum_{j=1}^{m} (\sum_{i=1}^{n} |X_{i,j}|^p)^{q/p} \right)^{1/q}.$$

When $p = q = \infty$, we define this to be $\|X\|_{\max}$ where $\|X\|_{\max} := \max_{i,j} |X_{i,j}|$.

Comparing matrix sparsification methods: Suppose that V is the vector space of real $n \times m$ matrices. Given $X \in V$, there are two standard atomic decompositions of X. The first is the entry-wise decomposition

$$X = \sum_{i,j} X_{i,j} e_i e_j^T.$$

The second is the singular value decomposition

$$X = \sum_{i=1}^{r} \sigma_i u_i v_i^T.$$

If r is small, it may be more efficient to communicate the $r(n+m)$ entries of the SVD, rather than the nm entries of the matrix. Let $\widehat{X}$ and $\widehat{X}_\sigma$ denote the random variables in (2) corresponding to the entry-wise decomposition and singular value decomposition of X, respectively. We wish to compare these two sparsifications.

In Table 1, we compare the communication cost and second moment of these two methods. The communication cost is the expected number of non-zero elements (real numbers) that need to be communicated. For $\widehat{X}$, a sparsity budget of s corresponds to s non-zero entries we need to communicate. For $\widehat{X}_\sigma$, a sparsity budget of s gives a communication cost of $s(n+m)$ due to the singular vectors. We compare the optimal second moment from Theorem 5.

Table 1: Communication cost versus second moment of singular value sparsification and vectorized matrix sparsification of a $n \times m$ matrix.

	COMMUNICATION COST	SECOND MOMENT
ENTRY-WISE	s	$\dfrac{1}{s}\|X\|_{1,1}^2$
SVD	$s(n+m)$	$\dfrac{1}{s}\|X\|_*^2$

To compare the second moment of these two methods under the same communication cost, we s and suppose X is s-balanced entry-wise. By Theorem 5 and Lemma 3, the second moment in Table 1 is achieved iff

$$s \leq \frac{\|X\|_{1,1}}{\|X\|_{\max}}.$$

To achieve the same communication cost with $\widehat{X}_\sigma$, we take a sparsity budget of $s' = s/(n+m)$. By Theorem 5 and Lemma 3, the second moment in Table 1 is achieved iff

$$s' = \frac{s}{n+m} \leq \frac{\|X\|_*}{\|X\|_2}.$$

This leads to the following theorem.

Theorem 7. *Suppose $X \in \mathbb{R}^{n \times m}$ and*

$$s \leq \min \left\{ \frac{\|X\|_{1,1}}{\|X\|_{\max}}, (n+m) \frac{\|X\|_*}{\|X\|_2} \right\}.$$

Then $\widehat{X}_\sigma$ with sparsity $s' = s/(n+m)$ incurs the same communication cost as $\widehat{X}$ with sparsity s, and $\mathbb{E}[\|\widehat{X}_\sigma\|^2] \leq \mathbb{E}[\|\widehat{X}\|^2]$ if and only if

$$(n+m)\|X\|_*^2 \leq \|X\|_{1,1}^2.$$

To better understand this condition, we will make use of the following lemma concerning the equivalence of the $\ell_{1,1}$ and spectral norms.

Lemma 8. *For any $n \times m$ matrix X over $\mathbb{R}$, $\frac{1}{\sqrt{nm}}\|X\|_{1,1} \leq \|X\|_* \leq \|X\|_{1,1}$.*

We give a proof of this fact in Appendix B and show that these bounds are the best possible. In other words, there are matrices for which both of the inequalities are equalities. If the first inequality is tight, then $\mathbb{E}[\|\widehat{X}_\sigma\|^2] \leq \mathbb{E}[\|\widehat{X}\|^2]$, while if the second is tight then $\mathbb{E}[\|\widehat{X}_\sigma\|^2] \geq \mathbb{E}[\|\widehat{X}\|^2]$. As we show in the next section, using singular value sparsification can translate in to significantly reduced distributed training time.

6 Experiments

We present an empirical study of Spectral-ATOMO and compare it to the recently proposed QSGD [4], on a different neural network models and data sets, under real distributed environments. Our main findings are as follows:

- We observe that spectral-ATOMO provides a useful alternative to entry-wise sparsification methods, it reduces communication compared to vanilla mini-batch SGD, and can reduce training time compared to QSGD by up to a factor of $2\times$. For instance, on VGG11-BN trained on CIFAR-10, spectral-ATOMO with sparsity budget 3 achieves $3.96\times$ speedup over vanilla SGD, while 4-bit QSGD achieves $1.68\times$ on a cluster of 16, g2.2xlarge instances.

- We observe that spectral-ATOMO in distributed settings leads to models with negligible accuracy loss when combined with parameter tuning.

Implementation and setup We compare spectral-ATOMO with different sparsity budgets to b-bit QSGD across a distributed cluster with a parameter server (PS), implemented in mpi4py [12] and PyTorch [38] and deployed on multiple types of instances in Amazon EC2 (*e.g.*, m5.4xlarge, m5.2xlarge, and g2.2xlarge), both PS and compute nodes are of the same type of instance. The PS implementation is standard, with a few important modifications. At the most basic level, it receives gradients from the compute nodes and broadcasts the updated model once a batch has been received.

In our experiments, we use data augmentation (random crops, and flips), and tuned the step size for every different setup as shown in Table 3. Momentum and regularization terms are switched off to make the hyperparameter search tractable and the results more legible. Tuning the step sizes for this distributed network for three different datasets and eight different coding schemes can be computationally intensive. As such, we only used small networks so that multiple networks

could fit into GPU memory. To emulate the effect of larger networks, we use synchronous message communication, instead of asynchronous.

Each compute node evaluates gradients sampled from its partition of data. Gradients are then sparsified through QSGD or spectral-ATOMO, and then are sent back to the PS. Note that spectral-ATOMO transmits the weighted singular vectors sampled from the true gradient of a layer. The PS then combines these, and updates the model with the average gradient. Our entire experimental pipeline is implemented in PyTorch [38] with mpi4py [12], and deployed on either g2.2xlarge, m5.2xlarge and m5.4xlarge instances in Amazon AWS EC2. We conducted our experiments on various models, datasets, learning tasks, and neural network models as detailed in Table 2.

Dataset	CIFAR-10	CIFAR-100	SVHN
# Data points	60,000	60,000	600,000
Model	ResNet-18 / VGG-11-BN	ResNet-18	ResNet-18
# Classes	10	100	10
# Parameters	11,173k / 9,756k	11,173k	11,173k

Table 2: The datasets used and their associated learning models and hyperparameters.

Experiments	CIFAR-10 & ResNet-18	SVHN & ResNet-18	CIFAR-10 & VGG-11-BN
SVD rank 1	0.0625	0.1	0.125
SVD rank 2	0.0625	0.125	0.125
SVD rank 3	0.125	0.125	0.0625
SVD rank 4	0.0625	0.125	0.15
QSGD 1bit	0.0078125	0.0078125	0.0009766
QSGD 2bit	0.0078125	0.0078125	0.0009766
QSGD 4bit	0.125	0.046875	0.015625
QSGD 8bit	0.125	0.125	0.0625

Table 3: Tuned step sizes for experiments

Scalability   We study the scalability of these sparsification methods on clusters of different sizes. We used clusters with one PS and $n = 2, 4, 8, 16$ compute nodes. We ran ResNet-34 on CIFAR-10 using mini-batch SGD with batch size 512 split among compute nodes. The experiment was run on m5.4xlarge instances of AWS EC2 and the results are shown in Figure 2.

While increasing the size of the cluster, decreases the computational cost per worker, it causes the communication overhead to grow. We denote as computational cost, the time cost required by each worker for gradient computations, while the communication overhead is represented by the amount time the PS waits to receive the gradients by the slowest worker. This increase in communication cost is non-negligible, even for moderately-sized networks with sparsified gradients. We observed a trade-off in both sparsification approaches between the information retained in the messages after sparsification and the communication overhead.
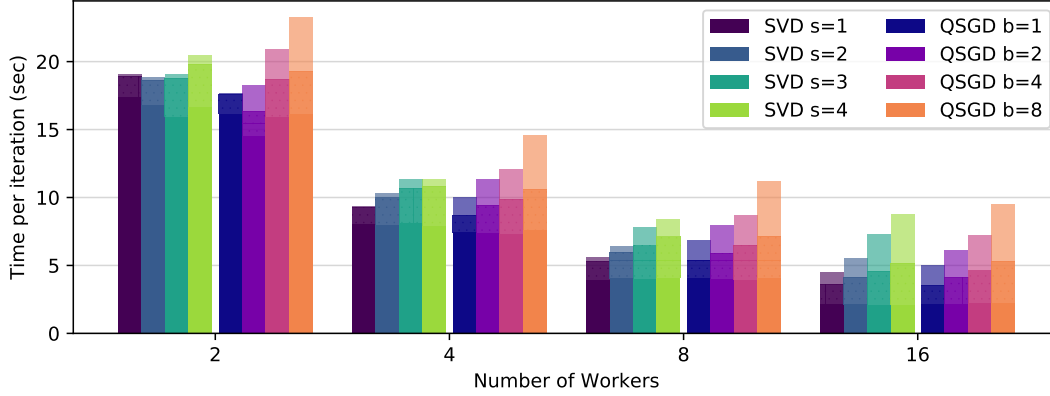
Figure 2: The timing of the gradient coding methods (QSGD and spectral-Atomo) for different quantization levels, b bits and s sparsity budget respectively for each worker when using a ResNet-34 model on CIFAR10. For brevity, we use SVD to denote spectral-Atomo. The bars represent the total iteration time and are divided into computation time (bottom, solid), encoding time (middle, dotted) and communication time (top, faded).

Runtime analysis We empirically study runtime costs of spectral-Atomo with sparsity budget set at 1, 2, 3, 6 and made comparisons among b-bit QSGD and TernGrad. We deployed distributed training on ResNet-18 with batch size $B = 256$ on the CIFAR-10 dataset run with m5.2xlarge instances. As shown in Figure 3, there is a trade-off between the amount of communication per iteration and the running time for both singular value sparsification and QSGD. In some scenarios, spectral-Atomo attains a higher compression ratio than QSGD and TernGrad. For example, singular value sparsification with sparsity budget 1 may communicate smaller messages than $\{2, 4\}$-bit QSGD and TernGrad.
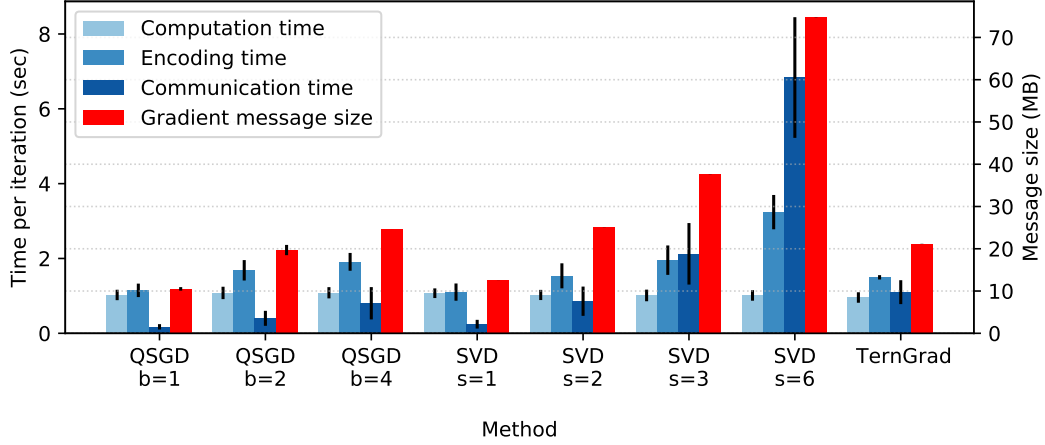

Figure 3: Runtime analysis of different sparsification methods (singular value sparsification, QSGD, and TernGrad) for ResNet-18 trained on CIFAR-10. The values shown are computation, encoding and communication time as well as the size of the message required to send gradients between workers.

Table 4: Speedups of spectral-ATOMO with sparsity budget s and b-bit QSGD using ResNet-18 on CIFAR10 over vanilla SGD.

End-to-end convergence performance We evaluate the end-to-end convergence performance on different datasets and neural networks, training with spectral-ATOMO(with sparsity budget $s = 1, 2, 3, 4$), QSGD (with $n = 1, 2, 4, 8$ bits), and ordinary mini-batch SGD. The datasets and models are summarized in Table 2. We use ResNet-18 [23] and VGG11-BN [46] for CIFAR-10 [31] and SVHN [37]. Again, for each of these methods we tune the step size. The experiments were run on a cluster of 16 compute nodes instantiated on g2.2xlarge instances.

The gradients of convolutional layers are 4 dimensional tensors with shape of $[x, y, k, k]$ where x, y are two spatial dimensions and k is the size of the convolutional kernel. However, matrices are required to compute the SVD for spectral-ATOMO, and we choose to reshape each layer into a matrix of size $[xy/2, 2k^2]$. This provides more flexibility on the sparsity budget for the SVD sparsification. For QSGD, we use the bucketing and Elias recursive coding methods proposed in [4], with bucket size equal to the number of parameters in each layer of the neural network.

Figure 4 and 5 shows how the testing accuracy varies with wall clock time and number of iterations respectively. Tables 4, 5, and 6 give a detailed account of speedups of singular value sparsification compared to QSGD. In these tables, each method is run until a specified accuracy.

(a) CIFAR-10, ResNet-18, Best of QSGD and SVD

(b) SVHN, ResNet-18, Best of QSGD and SVD
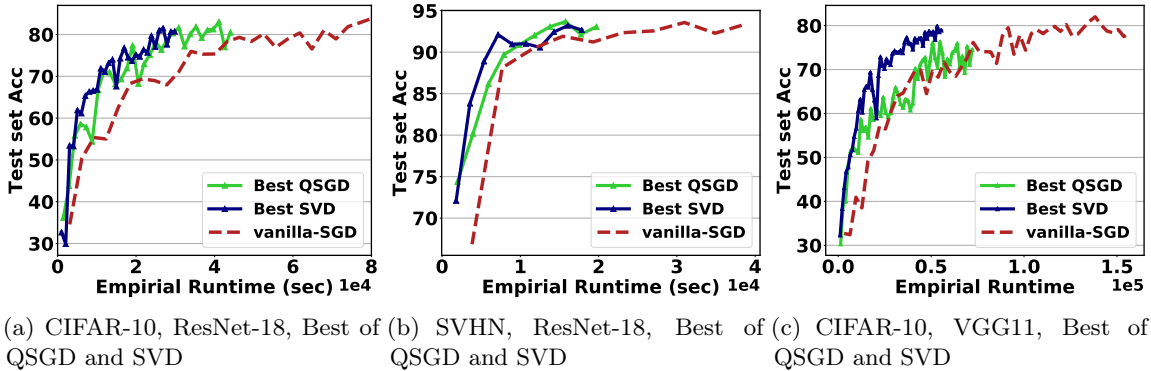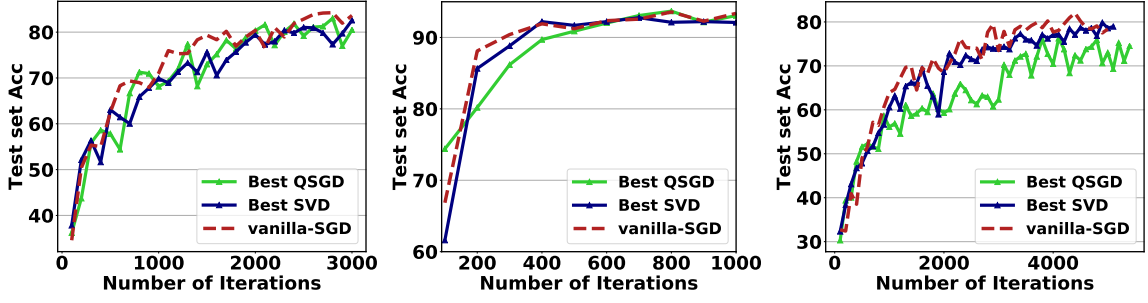
(c) CIFAR-10, VGG11, Best of QSGD and SVD

Figure 4: Convergence rates for the best performance of QSGD and spectral-ATOMO, alongside vanilla SGD. (a) uses ResNet-18 on CIFAR-10, (b) uses ResNet-18 on SVHN, and (c) uses VGG-11-BN on CIFAR-10. For brevity, we use SVD to denote spectral-ATOMO.

We observe that both QSGD and ATOMO speed up model training significantly and achieve

13

(a) CIFAR-10, ResNet-18, Best of QSGD and SVD (b) SVHN, ResNet-18, Best of QSGD and SVD (c) CIFAR-10, VGG11, Best of QSGD and SVD

Figure 5: Convergence rates with respect to number of iterations on: (a) CIFAR-10 on ResNet-18 of best performances from QSGD and SVD (b) SVHN on ResNet-18 of best performances from QSGD and SVD, (c) CIFAR-10 on VGG-11-BN best of performances from QSGD and SVD

Test accuracy	SVD s=1	SVD s=2	SVD s=3	SVD s=4	QSGD b=4	QSGD b=8
65%	3.66x	3.02x	2.78x	2.71x	1.25x	1.23x
68%	2.71x	2.48x	2.53x	2.38x	1.13x	0.99x
71%	3.16x	2.96x	2.55x	2.5x	1.57x	1.29x
74%	2.89x	2.82x	3.96x	1.96x	1.68x	1.43x

Table 5: Speedups of spectral-ATOMO with sparsity budget s and b-bit QSGD using VGG11-BN on CIFAR-10 over vanilla SGD.

Test accuracy	SVD s=1	SVD s=2	SVD s=3	SVD s=4	QSGD b=1	QSGD b=2	QSGD b=4	QSGD b=8
80%	1.45x	2.61x	2.63x	1.94x	2.09x	2.89x	2.54x	2.56x
86%	1.45x	2.61x	1.75x	1.3x	2.09x	1.92x	1.69x	1.43x
90%	1.45x	1.98x	1.98x	1.17x	1.88x	2.16x	1.91x	1.62x
92%	2.17x	3.96x	3.96x	1.96x	1.91x	2.16x	3.81x	3.24x

Table 6: Speedups of SVD and QSGD using ResNet-18 on SVHN over vanilla SGD.

similar accuracy to vanilla mini-batch SGD. We also observe that the best performance is not achieve with the most sparsified, or quantized method, but the optimal method lies somewhere in the middle where enough information is preserved during the sparsification. For instance, 8-bit QSGD converges faster than 4-bit QSGD, and spectral-ATOMO with sparsity budget 3, or 4 seems to be the fastest. Higher sparsity can lead to a faster running time, but extreme sparsification can adversely affect convergence. For example, for a fixed number of iterations, 1-bit QSGD has the smallest time cost, but may converge much more slowly to an accurate model.

7   Conclusion

In this paper, we present and analyze ATOMO, a general sparsification method for distributed stochastic gradient based methods. ATOMO applies to any atomic decomposition, including the entry-wise and the SVD of a matrix, and can be applied to whichever is most beneficial. ATOMO generalizes 1-bit QSGD and TernGrad, and provably minimizes the variance of the sparsified gradient subject to a sparsity constraint on the atomic decomposition. We focus on the use ATOMO for sparsifying matrices, especially the gradients in neural network training. We show that applying ATOMO to the singular values of these matrices can lead to faster training than both vanilla SGD or QSGD, for the same communication budget. We present extensive experiments showing that ATOMO can lead to up to a $2\times$ speed-up in training time over QSGD.

In the future, we plan to explore the use of ATOMO with Fourier decompositions, due to its utility and prevalence in signal processing. We also plan to examine how we can sparsify and compress gradients in a joint fashion to further reduce communication costs. Finally, when sparsifying the SVD of a matrix, we only sparsify the singular values. We also note that it would be interesting to explore jointly sparsification of the SVD and and its singular vectors, which we leave for future work.

References

[1] 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 *OSDI*, volume 16, pages 265–283, 2016.

[2] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*, 2017.

[3] S Alexander. Transient weight misadjustment properties for the finite precision LMS algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(9):1250–1258, 1987.

[4] 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*, pages 1707–1718, 2017.

[5] José Carlos M Bermudez and Neil J Bershad. A nonlinear analytical model for the quantized LMS algorithm-the arbitrary step size case. *IEEE Transactions on Signal Processing*, 44(5):1175–1183, 1996.

[6] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signSGD: compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*, 2018.

[7] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[8] Chia-Yu Chen, Jungwook Choi, Daniel Brand, Ankur Agrawal, Wei Zhang, and Kailash Gopalakrishnan. Adacomp: Adaptive residual gradient compression for data-parallel distributed training. *arXiv preprint arXiv:1712.02679*, 2017.

[9] Jianmin Chen, Xinghao Pan, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous SGD. *arXiv preprint arXiv:1604.00981*, 2016.

[10] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

[11] Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. Better mini-batch algorithms via accelerated gradient methods. In *Advances in neural information processing systems*, pages 1647–1655, 2011.

[12] Lisandro D Dalcin, Rodrigo R Paz, Pablo A Kler, and Alejandro Cosimo. Parallel distributed computing using python. *Advances in Water Resources*, 34(9):1124–1139, 2011.

[13] Soham De, Abhay Yadav, David Jacobs, and Tom Goldstein. Big Batch SGD: Automated inference using adaptive batch sizes. *arXiv preprint arXiv:1610.05792*, 2016.

[14] Christopher De Sa, Matthew Feldman, Christopher Ré, and Kunle Olukotun. Understanding and optimizing asynchronous low-precision stochastic gradient descent. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pages 561–574. ACM, 2017.

[15] Christopher De Sa, Megan Leszczynski, Jian Zhang, Alana Marzoev, Christopher R Aberger, Kunle Olukotun, and Christopher Ré. High-accuracy low-precision training. *arXiv preprint arXiv:1803.03383*, 2018.

[16] Christopher De Sa, Christopher Re, and Kunle Olukotun. Global convergence of stochastic gradient descent for some non-convex matrix problems. In *International Conference on Machine Learning*, pages 2332–2341, 2015.

[17] Christopher M De Sa, Ce Zhang, Kunle Olukotun, and Christopher Ré. Taming the wild: A unified analysis of hogwild-style algorithms. In *Advances in neural information processing systems*, pages 2674–2682, 2015.

[18] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231, 2012.

[19] John C Duchi, Sorathan Chaturapruek, and Christopher Ré. Asynchronous stochastic convex optimization. *arXiv preprint arXiv:1508.00882*, 2015.

[20] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[21] R Gitlin, J Mazo, and M Taylor. On the design of gradient algorithms for digitally implemented adaptive filters. *IEEE Transactions on Circuit Theory*, 20(2):125–136, 1973.

[22] Demjan Grubic, Leo Tam, Dan Alistarh, and Ce Zhang. Synchronous multi-GPU deep learning with low-precision communication: An experimental study. 2018.

[23] 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*, pages 770–778, 2016.

[24] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017.

[25] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

[26] Martin Jaggi, Virginia Smith, Martin Takác, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 3068–3076, 2014.

[27] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[28] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016.

[29] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[30] Jakub Konečnỳ and Peter Richtárik. Randomized distributed mean estimation: Accuracy vs communication. *arXiv preprint arXiv:1611.07555*, 2016.

[31] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[32] Rémi Leblond, Fabian Pedregosa, and Simon Lacoste-Julien. ASAGA: asynchronous parallel SAGA. *arXiv preprint arXiv:1606.04809*, 2016.

[33] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

[34] Ji Liu, Stephen J Wright, Christopher Ré, Victor Bittorf, and Srikrishna Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. *The Journal of Machine Learning Research*, 16(1):285–322, 2015.

[35] Horia Mania, Xinghao Pan, Dimitris Papailiopoulos, Benjamin Recht, Kannan Ramchandran, and Michael I Jordan. Perturbed iterate analysis for asynchronous stochastic optimization. *arXiv preprint arXiv:1507.06970*, 2015.

[36] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[37] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[38] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017.

[39] Hang Qi, Evan R. Sparks, and Ameet Talwalkar. Paleo: A performance model for deep neural networks. In *Proceedings of the International Conference on Learning Representations*, 2017.

[40] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[41] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701, 2011.

[42] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pages 314–323, 2016.

[43] Cèdric Renggli, Dan Alistarh, and Torsten Hoefler. SparCML: high-performance sparse communication for machine learning. *arXiv preprint arXiv:1802.08021*, 2018.

[44] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6655–6659. IEEE, 2013.

[45] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[46] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[47] Nikko Strom. Scalable distributed DNN training using commodity gpu cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[48] Ananda Theertha Suresh, Felix X Yu, Sanjiv Kumar, and H Brendan McMahan. Distributed mean estimation with limited communication. *arXiv preprint arXiv:1611.00429*, 2016.

[49] Yusuke Tsuzuku, Hiroto Imachi, and Takuya Akiba. Variance-based gradient compression for efficient distributed deep learning. *arXiv preprint arXiv:1802.06058*, 2018.

[50] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. *arXiv preprint arXiv:1710.09854*, 2017.

[51] 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 *Advances in Neural Information Processing Systems*, pages 1508–1518, 2017.

[52] Simon Wiesler, Alexander Richard, Ralf Schluter, and Hermann Ney. Mean-normalized stochastic gradient for large-scale deep learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 180–184. IEEE, 2014.

[53] Jian Xue, Jinyu Li, and Yifan Gong. Restructuring of deep neural network acoustic models with singular value decomposition. In *Interspeech*, pages 2365–2369, 2013.

[54] Dong Yin, Ashwin Pananjady, Max Lam, Dimitris Papailiopoulos, Kannan Ramchandran, and Peter Bartlett. Gradient diversity: a key ingredient for scalable distributed learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1998–2007, 2018.

[55] Hantian Zhang, Jerry Li, Kaan Kara, Dan Alistarh, Ji Liu, and Ce Zhang. Zipml: Training linear models with end-to-end low precision, and a little bit of deep learning. In *International Conference on Machine Learning*, pages 4035–4043, 2017.

[56] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. DoReFa-Net: training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

A   Proof of results

A.1   Proof of Lemma 2

Proof. Suppose we have some p satisfying the conditions in (4). We define two auxiliary vectors $\alpha, \beta \in \mathbb{R}^n$ by

$$\alpha_i = \frac{|\lambda_i|}{\sqrt{p_i}}.$$

$$\beta_i = \sqrt{p_i}.$$

Then note that using the fact that $\sum_i p_i = s$, we have

$$f(p) = \sum_{i=1}^n \frac{\lambda_i^2}{p_i} = \left(\sum_{i=1}^n \frac{\lambda_i^2}{p_i}\right)\left(\frac{1}{s}\sum_{i=1}^n p_i\right) = \frac{1}{s}\left(\sum_{i=1}^n \frac{\lambda_i^2}{p_i}\right)\left(\sum_{i=1}^n p_i\right) = \frac{1}{s}\|\alpha\|_2^2\|\beta\|_2^2.$$

By the Cauchy-Schwarz inequality, this implies

$$f(p) = \frac{1}{s}\|\alpha\|_2^2\|\beta\|_2^2 \geq \frac{1}{s}|\langle\alpha,\beta\rangle|^2 = \frac{1}{s}\left(\sum_{i=1}^n |\lambda_i|\right)^2 = \frac{1}{s}\|\lambda\|_1^2. \tag{9}$$

This proves the first part of Lemma 2. In order to have $f(p) = \frac{1}{s}\|\lambda\|_1^2$, (9) implies that we need

$$|\langle\alpha,\beta\rangle| = \|\alpha\|_2\|\beta\|_2.$$

By the Cauchy-Schwarz inequality, this occurs iff α and β are linearly dependent. Therefore, $c\alpha = \beta$ for some constant c. Solving, this implies $p_i = c|\lambda_i|$. Since $\sum_{i=1}^n p_i = s$, we have

$$c\|\lambda\|_1 = \sum_{i=1}^n c|\lambda_i| = \sum_{i=1}^n p_i = s.$$

Therefore, $c = \|\lambda\|_1/s$, which implies the second part of the theorem. $\square$

A.2   Proof of Lemma 4

Fix q that is feasible in (3). To prove Lemma 4 we will require a lemma. Given the atomic decomposition $g = \sum_{i=1}^n \lambda_i a_i$, we say that λ is s-unbalanced at i if $|\lambda_i|s > \|\lambda\|_1$, which is equivalent to g being unbalanced in this atomic decomposition at i. For notational simplicity, we will assume that λ is s-unbalanced at $i = 1$. Let $A \subseteq \{2, \ldots, n\}$. We define the following notation:

$$s_A = \sum_{i\in A} q_i.$$

$$f_A(q) = \sum_{i\in A} \frac{\lambda_i^2}{q_i}.$$

$$(\lambda_A)_i = \begin{cases} \lambda_i, & \text{for } i \in A, \\ 0, & \text{for } i \notin A. \end{cases}$$

Note that under this notation, Lemma 2 implies that for all $p > 0$,

$$f_A(p) \geq \frac{1}{s_A}\|\lambda_A\|_1^2. \tag{10}$$

Lemma 9. *Suppose that q is feasible and that there is some set $A \subseteq \{2, \ldots, n\}$ such that*

1. λ_A *is $(s_A + q_1 - 1)$-balanced.*

2. $|\lambda_1|(s_A + q_1 - 1) > \|\lambda_A\|_1$.

Then there is a vector p that is feasible satisfying $f(p) \le f(q)$ and $p_1 = 1$.

Proof. Suppose that such a set A exists. Let $B = \{2, \ldots, n\} \backslash A$. Note that we have

$$f(q) = \sum_{i=1}^{n} \frac{\lambda_i^2}{q_i} = \frac{\lambda_1^2}{q_1} + f_A(q) + f_B(q).$$

By (10), this implies

$$f(q) \ge \frac{\lambda_1^2}{q_1} + \frac{1}{s_A}\|\lambda_A\|_1^2 + f_B(q). \tag{11}$$

Define p as follows.

$$p_i = \begin{cases} 1, & \text{for } i = 1, \\ \dfrac{|\lambda_i|(s_A + q_1 - 1)}{\|\lambda_A\|_1}, & \text{for } i \in A, \\ q_i, & \text{for } i \notin A. \end{cases}$$

Note that by Assumption 1 and Lemma 2, we have

$$f_A(p) = \frac{1}{s_A + q_1 - 1}\|\lambda_A\|_1^2.$$

Since $p_i = q_i$ for $i \in B$, we have $f_B(p) = f_B(q)$. Therefore,

$$f(p) = \lambda_1^2 + \frac{1}{s_A + q_1 - 1}\|\lambda_A\|_1^2 + f_B(q). \tag{12}$$

Combining (11) and (12), we have

$$f(q) - f(p) = \lambda_1^2\left(\frac{1}{q_1} - 1\right) + \|\lambda_A\|_1^2\left(\frac{1}{s_A} - \frac{1}{s_A + q_1 - 1}\right)$$

$$= \lambda_1^2\left(\frac{1 - q_1}{q_1}\right) + \|\lambda_A\|_1^2\left(\frac{q_1 - 1}{s_A(s_A + q_1 - 1)}\right).$$

Combining this with Assumption 2, we have

$$f(q) - f(p) \ge \frac{\|\lambda_A\|_1^2}{(s_A + q_1 - 1)^2}\left(\frac{1 - q_1}{q_1}\right) + \|\lambda_A\|_1^2\left(\frac{q_1 - 1}{s_A(r_A + q_1 - 1)}\right). \tag{13}$$

To show that the RHS of (13) is at most 0, it suffices to show

$$s_A \ge q_1(s_A + q_1 - 1). \tag{14}$$

However, note that since $0 < q_1 < 1$, the RHS of (14) satisfies

$$q_1(s_A + q_1 - 1) = s_A q_1 - q_1(1 - q_1) \le s_A q_1 \le s_A.$$

Therefore, (14) holds, completing the proof. $\qquad\square$

We can now prove Lemma 4. In the following, we will refer to Conditions 1 and 2, relative to some set A, as the conditions required by Lemma 9.

Proof. We first show this in the case that $n = 2$. Here we have the atomic decomposition

$$g = \lambda_1 a_1 + \lambda_2 a_2.$$

The condition that λ is s-unbalanced at $i = 1$ implies

$$|\lambda_1|(s - 1) > |\lambda_2|.$$

In particular, this implies $s > 1$. For $A = \{2\}$, Condition 1 is equivalent to

$$|\lambda_2|(s_A + q_1 - 2) \leq 0.$$

Note that $s_A = q_2$ and that $q_1 + q_2 - 2 = s - 2$ by assumption. Since $q_i \leq 1$, we know that $s - 2 \leq 0$ and so Condition 1 holds. Similarly, Condition 2 becomes

$$|\lambda_1|(s - 1) > |\lambda_2|$$

which holds by assumption. Therefore, Lemma 4 holds for $n = 2$.

Now suppose that $n > 2$, q is some feasible probability vector, and that λ is s-unbalanced at index 1. We wish to find an A satisfying Conditions 1 and 2. Consider $B = \{2, \ldots, n\}$. Note that for such B, $s_B + q_1 - 1 = s - 1$. By our unbalanced assumption, we know that Condition 2 holds for $B = \{2, \ldots, n\}$. If λ_B is $(s - 1)$-balanced, then Lemma 9 implies that we are done.

Assume that λ_B is not $(s - 1)$-balanced. After relabeling, we can assume it is unbalanced at $i = 2$. Let $C = \{3, \ldots, n\}$. Therefore,

$$|\lambda_2|(s - 2) > \|\lambda_C\|_1. \tag{15}$$

Combining this with the s-unbalanced assumption at $i = 1$, we find

$$\begin{aligned}
|\lambda_1| &> \frac{\|\lambda_B\|_1}{s - 1} \\
&= \frac{|\lambda_2|}{s - 1} + \frac{\|\lambda_C\|_1}{s - 1} \\
&> \frac{\|\lambda_C\|_1}{(s - 1)(s - 2)} + \frac{\|\lambda_C\|_1}{s - 1} \\
&= \frac{\|\lambda_C\|_1}{s - 2}.
\end{aligned}$$

Therefore,

$$|\lambda_1|(s - q_2 - 1) \geq |\lambda_1|(s - 2) > \|\lambda_C\|_1. \tag{16}$$

Let $D = \{1, 3, 4, \ldots, n\} = \{1, \ldots, n\}\setminus\{2\}$. Then note that (16) implies that λ_D is $(s - q_2)$-unbalanced at $i = 1$. Inductively applying this theorem, this means that we can find a vector $p' \in \mathbb{R}^{|D|}$ such that $p'_1 = 1$ and $f_D(p') \leq f_D(q)$. Moreover, $s_D(p') = s - q_2$. Therefore, if we let p be the vector that equals p' on D and with $p_2 = q_2$, we have

$$f(p_2) = f_C(p') + \frac{\lambda_2^2}{q_2} \leq f_D(q) + \frac{\lambda_2^2}{q_2} = f(q).$$

This proves the desired result. $\qquad\square$

B   Equivalence of norms

We are often interested in comparing norms on vectors spaces. This naturally leads to the following definition.

Definition 3. *Let V be a vector space over $\mathbb{R}$ or $\mathbb{C}$. Two norms $\|\cdot\|_a, \|\cdot\|_b$ are equivalent if there are positive constants C_1, C_2 such that*

$$C_1\|x\|_a \leq \|x\|_b \leq C_2\|x\|_a$$

for all $x \in V$.

As it turns out, norms on finite-dimensional vector spaces are always equivalent.

Theorem 10. *Let V be a finite-dimensional vector space over $\mathbb{R}$ or $\mathbb{C}$. Then all norms are equivalent.*

In order to compare norms, we often wish to determine the tightest constants which give equivalence between them. In Section 5, we are particularly interested in comparing the $\|X\|_*$ and $\|X\|_{1,1}$ on the space of $n \times m$ matrices. We have the following lemma.

Lemma 11. *For all $n \times m$ real matrices,*

$$\frac{1}{\sqrt{nm}}\|X\|_{1,1} \leq \|X\|_* \leq \|X\|_{1,1}.$$

Proof. Suppose that X has the singular value decomposition

$$X = \sum_{i=1}^{r} \sigma_i u_i v_i^T.$$

We will first show the left inequality. First, note that for any $n \times m$ matrix A, $\|A\|_{1,1} \leq \sqrt{nm}\|A\|_F$. This follows directly from the fact that for a n-dimensional vector v, $\|v\|_1 \leq \sqrt{n}\|v\|_2$. We will also use the fact that for any vectors $u \in \mathbb{R}^n, v \in \mathbb{R}^m$, $\|uv^T\|_F = \|u\|_2\|v\|_2$. We then have

$$
\begin{aligned}
\|X\|_{1,1} &= \left\|\sum_{i=1}^{r} \sigma_i u_i v_i^T\right\|_{1,1} \\
&\leq \sum_{i=1}^{r} \sigma_i \|u_i v_i^T\|_{1,1} \\
&= \sum_{i=1}^{r} \sigma_i \sqrt{nm}\|u_i v_i^T\|_F \\
&= \sum_{i=1}^{r} \sigma_i \sqrt{nm}\|u_i\|_2\|v_i\|_2 \\
&= \|X\|_*.
\end{aligned}
$$

For the right inequality, note that we have

$$X = \sum_{i,j} X_{i,j} e_i e_j^T$$

23

where $e_i \in \mathbb{R}^n$ is the i-th standard basis vector, while $e_j \in \mathbb{R}^m$ is the j-th standard basis vector. We then have

$$\|X\|_* \leq \sum_{i,j} |X_{i,j}| \|e_i e_j^T\|_* = \sum_{i,j} |X_{i,j}| = \|X\|_{1,1}.$$

$\square$

In fact, these are the best constants possible. To see this, first consider the matrix X with a 1 in the upper-left entry and 0 elsewhere. Clearly, $\|X\|_* = \|X\|_{1,1} = 1$, so the right-hand inequality is tight. For the left-hand inequality, consider the all-ones matrix X. This has one singular value, $\sqrt{nm}$, so $\|X\|_* = \sqrt{nm}$. On the other hand, $\|X\|_{1,1} = nm$. Therefore, $\|X\|_{1,1} = \sqrt{nm}\|X\|_*$ in this case.

C   Analysis of ATOMO via the KKT Condtions

In this section we show how to derive Algorithm 1 using the KKT conditions. Recall that we wish to solve the following optimization problem:

$$\min_p \ f(p) := \sum_{i=1}^n \frac{\lambda_i^2}{p_i} \quad \text{subject to} \ \ \forall i, \ 0 < p_i \leq 1, \ \ \sum_{i=1}^n p_i = s. \tag{17}$$

We first note a few immediate consequences.

1. If $s > n$ then the problem is infeasible. Note that when $s \geq n$, the optimal thing to do is to set all $p_i = 1$, in which case no sparsification takes place.

2. If $\lambda_i = 0$, then $p_i = 0$. This follows from the fact that this p_i does not change the value of $f(p)$, and the objective could be decreased by allocating more to the p_j associated to non-zero λ_j. Therefore we can assume that all $\lambda_i \neq 0$.

3. If $|\lambda_i| \geq |\lambda_j| > 0$, then we can assume $p_i \geq p_j$. Otherwise, suppose $p_j > p_i$ but $|\lambda_i| \geq |\lambda_j|$. Let p' denote the vector with p_i, p_j switched. We then have

$$f(p) - f(p') = \frac{\lambda_i^2 - \lambda_j^2}{p_i} + \frac{\lambda_j^2 - \lambda_i^2}{p_j}$$

$$= \lambda_i^2 \left(\frac{1}{p_i} - \frac{1}{p_j}\right) - \lambda_j^2 \left(\frac{1}{p_i} - \frac{1}{p_j}\right)$$

$$\geq 0.$$

We therefore assume $0 < s \leq n$ and $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n| > 0$. As above we define $\lambda := [\lambda_1, \ldots, \lambda_n]^T$. While the formulation of (17) does not allow direct application of the KKT conditions, since we have a strict inequality of $0 < p_i$, this is fixed with the following lemma.

Lemma 12. *The minimum of* (17) *is achieved by some p^* satisfying*

$$p_i^* \geq \frac{s\lambda_i^2}{n\|\lambda\|_2^2}.$$

Proof. Define $\bar{p}$ by $\bar{p}_i = s/n$. This vector is clearly feasible in (17). Let p be any feasible vector. If $f(p) \leq f(q)$ then for any $i \in [n]$ we have

$$\frac{\lambda_i^2}{p_i} \leq f(p) \leq f(\bar{p}).$$

Therefore, $p_i \geq \lambda_i^2/f(\bar{p})$. A straightforward computations shows that $f(\bar{p}) = n\|\lambda\|_2^2/s$. Note that this implies that we can restrict to the feasbile set

$$\frac{s\lambda_i^2}{n\|\lambda\|_2^2} \leq p_i \leq 1.$$

This defines a compact region C. Since f is continuous on this set, its maximum value is obtained at some p^*. $\qquad\square$

The KKT conditions then imply that at any point p solving (17), we must have

$$0 \leq 1 - p_i \perp \mu - \frac{\lambda_i^2}{p_i} \geq 0, \quad i = 1, 2, \ldots, n \tag{18}$$

for some $\mu \in \mathbb{R}$. Since $|\lambda_i| > 0$ for all i, we actually must have $\mu > 0$. We therefore have two conditions for all i.

1. $p_i = 1 \implies \mu \geq \lambda_i^2$.

2. $p_i < 1 \implies p_i = |\lambda_i|/\sqrt{\mu}$.

Note that in either case, to have p_1 feasible we must have $\mu \geq \lambda_1^2$. Combining this with the fact that we can always select $p_1 \geq p_2 \geq \ldots \geq p_n$, we obtain the following partial characterization of the solution to (17). For some $n_s \in [n]$, we have $p_1, \ldots, p_{n_s} = 1$ while $p_i = |\lambda_i|/\sqrt{\mu} \in (0, 1)$ for $i = n_s + 1, \ldots, n$. Combining this with the constraint that $\sum_{i=1}^n p_i = s$, we have

$$s = \sum_{i=1}^n p_i = n_s + \sum_{i=n_s+1}^n p_i = n_s + \sum_{i=n_s+1}^n \frac{|\lambda_i|}{\sqrt{\mu}}. \tag{19}$$

Rearranging, we obtain

$$\mu = \frac{\left(\sum_{i=n_s+1}^n |\lambda_i|\right)^2}{(s - n_s)^2} \tag{20}$$

which then implies that

$$p_i = 1, \quad i = 1, \ldots, n_s, \quad p_i = \frac{|\lambda_i|(s - n_s)}{\sum_{j=n_s+1}^n |\lambda_j|}, \quad i = n_s + 1, \ldots, n. \tag{21}$$

Thus, we need to select n_s such that the p_i in (21) are bounded above by 1. Let n_s^* denote the first element of $[n]$ for which this holds. Then the condition that $p_i \leq 1$ for $i = n_s^* + 1, \ldots, n$ is exactly the condition that $[\lambda_{n_s^*+1}, \ldots, \lambda_n]$ is $(s - n_s)$-balanced (see Definition 2. In particular, Lemma 2 implies that, fixing $p_i = 1$ for $i = 1, \ldots, n_s^*$, the optimal way to assign the remaining p_i is by

$$p_i = \frac{|\lambda_i|(s - n_s^*)}{\sum_{j=n_s^*+1}^n |\lambda_j|}.$$

This agrees with (21) for $n_s = n_s^*$. In particular, the minimal value of f occurs at the first value of n_s such that the p_i in (21) are bounded above by 1.

Algorithm 1 scans through the sorted λ_i and finds the first value of n_s for which the probabilities in (21) are in $[0, 1]$, and therefore finds the optimal p for (17). The runtime is dominated by the $O(n \log n)$ sorting cost. It is worth noting that we could perform the algorithm in $O(sn)$ time as well. Instead of sorting and then iterating through the λ_i in order, at each step we could simply select the next largest $|\lambda_i|$ not yet seen and perform an analogous test and update as in the above algorithm. Since we would have to do the selection step at most s times, this leads to an $O(sn)$ complexity algorithm.