2024 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 22–25, 2024, LONDON, UK

NOVEL GRADIENT SPARSIFICATION ALGORITHM VIA BAYESIAN INFERENCE

Ali Bereyhi and Ben Liang

ECE Department
University of Toronto
ali.bereyhi@utoronto.ca, liang@ece.utoronto.ca

Gary Boudreau and Ali Afana

Ericsson Canada
Ottawa, Canada
{gary.boudreau,ali.afana}@ericsson.com

ABSTRACT

Error accumulation is an essential component of the TOP-k sparsification method in distributed gradient descent. It implicitly scales the learning rate and prevents the slow-down of lateral movement, but it can also deteriorate convergence. This paper proposes a novel sparsification algorithm called *regularized* TOP-k (REGTOP-k) that controls the learning rate scaling of error accumulation. The algorithm is developed by looking at the gradient sparsification as an inference problem and determining a Bayesian optimal sparsification mask via maximum-a-posteriori estimation. It utilizes past aggregated gradients to evaluate posterior statistics, based on which it prioritizes the local gradient entries. Numerical experiments with ResNet-18 on CIFAR-10 show that at 0.1% sparsification, REGTOP-k achieves about 8% higher accuracy than standard TOP-k.

Index Terms— Gradient sparsification, TOP-k algorithm, Bayesian inference, distributed stochastic gradient descent, communication-efficient distributed learning

1. INTRODUCTION

Consider a distributed stochastic gradient descent (SGD) setting [1], where N workers compute gradients and share them with a server to estimate a *global* gradient. In iteration t, worker n starts from a common model $\mathbf{w}^t \in \mathbb{R}^J$ and computes its *local* gradient using a local surrogate loss function determined by averaging the loss over a stochastically-selected mini-batch. Let $\mathbf{g}_n^t \in \mathbb{R}^J$ denote the gradient computed by worker n. After receiving $\mathbf{g}_1^t, \ldots, \mathbf{g}_N^t$, the server estimates the global gradient $\mathbf{g}^t$ by weighted averaging, i.e.,

$$\mathbf{g}^t = \sum_{n=1}^{N} \omega_n \mathbf{g}_n^t \tag{1}$$

for some non-negative weights ω_n. The estimated gradient is used to update the common model as $\mathbf{w}^{t+1} = \mathbf{w}^t - \eta^t \mathbf{g}^t$ with some learning rate η^t.

With realistic models, the scale of communication in such settings can be prohibitive, e.g., for ResNet-110 $J \approx 1.7\times10^6$ [2]. Assuming 1000 mini-batches at each worker, the network exchanges 1.7×10^9 symbols per epoch for each worker. A classical solution to this issue is *gradient sparsification* [1, 3]: in each iteration, the workers only send *important gradient entries* along with their indices. The term *sparsification* refers to the fact that this approach can be interpreted as if the sever approximates local gradients with their *sparsified version*. Typical order of sparsity in practice is less than 1%, i.e., fewer than $0.01J$ entries are sent by each worker [3, 4, 5, 6, 7].

The key point in gradient sparsification is the design of a mechanism which can efficiently find *important gradient entries*. This can be challenging, as each worker has no explicit information about the gradients of other workers, and hence decides *locally*. In this work, we address this challenge by developing a new algorithm called REGTOP-k, which extracts *global* information from earlier iterations. REGTOP-k can be seen as the classical TOP-k with regularization that controls the *learning rate scaling property* of error accumulation. We illustrate this property next.

1.1. Error Accumulation and Learning Rate Scaling

The standard approach for sparsification is TOP-k, which selects the k largest *accumulated* gradient entries in each iteration. In iteration t, worker n computes its local gradient $\mathbf{g}_n^t$ and the accumulated gradient as $\mathbf{a}_n^t = \boldsymbol{\epsilon}_n^t + \mathbf{g}_n^t$, where $\boldsymbol{\epsilon}_n^t$ is the *sparsification error* from the previous iteration. It then selects the k entries of $\mathbf{a}_n^t$ with the largest amplitude which results in the sparsified gradient $\hat{\mathbf{g}}_n^t \in \mathbb{R}^J$. The sparsification error is then updated as $\boldsymbol{\epsilon}_n^{t+1} = \mathbf{a}_n^t - \hat{\mathbf{g}}_n^t$.

As one may notice, beyond naive selection of the k largest gradient entries, a key procedure in TOP-k is *error accumulation*, i.e., the computation of sparsification error $\boldsymbol{\epsilon}_n^t$. This way, the initially unselected entries get the chance of being selected after their errors become large enough. After such an entry is eventually selected, SGD moves a large step in the entry's direction, whose length is proportional to the gradient of that entry *accumulated* in previous iterations. This behavior is known as *learning rate scaling* [8]. Though this scaling is fairly effective for smooth losses, for other losses it can result in either alternation around the optimum or divergence. This behavior is best understood through a toy example that is given in the sequel.

1.2. A Motivational Example

Consider logistic regression with $J = 2$, where $N = 2$ workers employ distributed gradient descent to minimize the cross-entropy. Let worker 1 and 2 have single data-points $(\mathbf{x}_1, 1)$ and $(\mathbf{x}_2, 1)$, respectively, where $\mathbf{x}_1 = [100, 1]$, and $\mathbf{x}_2 =$

This work was funded in part by Ericsson Canada and by the Natural Sciences and Engineering Research Council of Canada.

979-8-3503-7225-0/24/$31.00 ©2024 IEEE

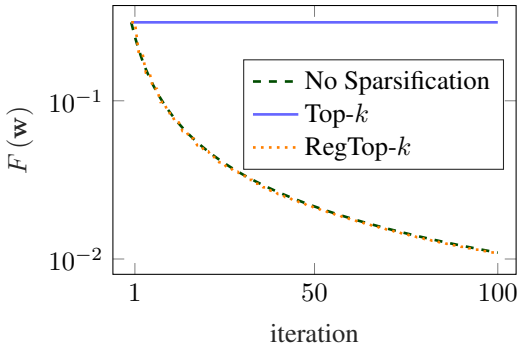

Fig. 1. Example of large learning rate scaling in TOP-k.

$[-100, 1]$. The workers agree on a model with weight vector $\mathbf{w} = [\theta_1, \theta_2]$ and zero bias. The loss of worker $n \in \{1, 2\}$ in this case is $F_n(\mathbf{w}) = \log(1 + \exp\{-\langle \mathbf{w}; \mathbf{x}_n \rangle\})$, and the empirical risk used for distributed training is $F(\mathbf{w}) = (F_1(\mathbf{w}) + F_2(\mathbf{w}))/2$. Worker n shares its gradient

$$\mathbf{g}_n = -\frac{\exp\{-\langle \mathbf{w}; \mathbf{x}_n \rangle\} \mathbf{x}_n}{1 + \exp\{-\langle \mathbf{w}; \mathbf{x}_n \rangle\}}, \tag{2}$$

with the server who computes $\mathbf{g} = 0.5(\mathbf{g}_1 + \mathbf{g}_2)$.

Figure 1 shows training loss against iterations for learning rate $\eta = 0.9$ and $\mathbf{w}^0 = [0, 1]$ for both TOP-1 sparsification and non-sparsified cases. It is observed that TOP-1 is not able to reduce the empirical risk even after 100 iterations. This behavior can be explained as follows: at $\mathbf{w}^0$, the gradients are $\mathbf{g}_1 = 0.736\,[-100, 1]$ and $\mathbf{g}_2 = 0.736\,[100, 1]$, and TOP-1 selects the first entry at both workers. One can however see that despite their significantly larger amplitudes, the first entries do not contribute in the training, as they cancel out after averaging. With TOP-1, the aggregated sparsified gradient remains zero, and hence the distributed gradient descent remains at $\mathbf{w}^0$ for several iterations. It starts to move from the initial point only after a large number of iterations when the accumulated sparsification error at the second entry starts to surpass the first entry in amplitude. For comparison, we further show the results for REGTOP-1: our proposed algorithm tracks non-sparsified training consistently.

One can readily extend this toy-example to a setting where learning rate scaling hinders the convergence. For instance, let the loss of worker n be $\tilde{F}_n(\mathbf{w}) = F_n(\mathbf{w}) + G(\theta_2)$, for some $G(\theta_2)$ whose derivative at $\theta_2 = 1$ is 1. Starting from $\mathbf{w}^0 = [0, 1]$, TOP-1 aggregates zero gradients in the first 50 iterations and moves from $\mathbf{w}^0$ only at $t = 51$ when the accumulation error at both workers read $\boldsymbol{\epsilon}_n^t = [0, 100]$. At this iteration, the workers send their sparsified accumulated gradients $\hat{\mathbf{g}}_n^t = [0, 100]$, which leads to $\mathbf{g}^t = [0, 100]$. Comparing this gradient with the non-sparsified aggregation in the first iteration, one can see that TOP-1 scales the learning rate with factor 50. Depending on $G(\theta_2)$, this large scaling of the learning rate can deteriorate the convergence of the optimizer.

1.3. Related Work

Several lines of work have extended distributed SGD with TOP-k: the study in [9] proposes an adaptive sparsification technique based on TOP-k aiming to reduce the computational complexity. The authors of [8] develop the *deep gradient compression scheme* that incorporates the ideas of momentum correction into TOP-k. Online adaptation of TOP-k with the goal of having minimum training time is discussed in [10]. In [11], the TOP-k extension *Atomo* is introduced, which sparsifies the gradients in an arbitrary atomic decomposition space. Layer-wise gradient sparsification for deep neural networks (DNNs) is further investigated in [12]. In [13], the authors propose a scalable sparsified gradient compression that uses the similarity of local gradients to enhance the scalability of TOP-k. The above lines of work substantially differ from our study as *they mainly focus on adapting the classical* TOP-k *to a wider range of distributed settings*, e.g., various optimizers. Unlike these lines of work, our study aims to develop a new sparsification scheme that *controls the learning rate scaling*.

Developing sparsification algorithms based on model statistics has been recently investigated in [7] and [14]. In [7], the authors revisit TOP-k sparsification and give an alternative interpretation as the optimal sparsifier for per-iteration communication budget. The notion of optimality is then extended to the entire training leading to a new TOP-k based sparsification scheme. The study in [14] proposes a statistical approach for gradient sparsification by treating local gradients as random variables distributed with empirically-validated sparsity-inducing distributions. It is worth mentioning that, these studies do *not* propose any mechanism for controlling the learning rate scaling and mainly focus on extending TOP-k sparsification under a new set of design constraints. To our best knowledge, *our work is the first study that develops a sparsification algorithm based on error accumulation that controls learning rate scaling.*

1.4. Contributions

In this work, we develop a Bayesian framework for gradient sparsification. Unlike earlier studies, we focus on the learning rate scaling of TOP-k and propose a regularization technique to control this property. In a nutshell, our main contributions are as follows: (1) We formulate gradient sparsification as an inference problem. Invoking this formulation, we represent the Bayesian optimal sparsifier as a maximum-a-posterior (MAP) estimator. (2) We construct a prior belief on the gradient entries by interpreting TOP-k as a *mismatched* MAP sparsifier with postulated uniform likelihood. We call this prior distribution the TOP-k prior belief. (3) Using the TOP-k prior belief, we determine the optimal sparsification mask and show that it is a regularized form of the TOP-k sparsifier. We validate our derivations through numerical experiments, which suggest that while TOP-k oscillates at a fixed optimality gap, REGTOP-k can converge to the global optimum with significantly sparser local gradients.

Notation Vectors are shown in bold, e.g., $\mathbf{x}$. The j-th entry of vector $\mathbf{x}$ is represented as $x_{[j]}$. Entry-wise multiplication and division are denoted by $\odot$ and $\oslash$, respectively. The inner product of $\mathbf{x}$ and $\mathbf{y}$ is represented by $\langle \mathbf{x}; \mathbf{y} \rangle$. The ℓ_p-norm of $\mathbf{x}$ is denoted by $\|\mathbf{x}\|_p$. For an integer N, the set $\{1, \ldots, N\}$ is abbreviated as $[N]$. We denote the cardinality of set $\mathbb{S}$ by $|\mathbb{S}|$.

2. PRELIMINARIES

We consider the distributed setting described in Section 1. Let $\mathbb{D}_n = \{\mathbf{x}_{n,i} \text{ for } i \in [D_n]\}$ denote the training batch of size D_n at worker $n \in [N]$ whose entries are sampled independent and identically distributed (i.i.d.) from $p(\mathbf{x})$. The distributed training in this setting is formulated as

$$\min_{\mathbf{w}\in\mathbb{R}^J} \sum_{n=1}^{N} \omega_n F_n(\mathbf{w}) \tag{3}$$

for some ω_n proportional to D_n, where $F_n(\mathbf{w})$ is the empirical loss computed by worker n, i.e.,

$$F_n(\mathbf{w}) = \frac{1}{D_n} \sum_{i=1}^{D_n} f(\mathbf{w}|\mathbf{x}_{n,i}) \tag{4}$$

for some loss function $f(\mathbf{w}|\mathbf{x})$. The goal is to solve this optimization in a distributed fashion with minimal communication overhead. We consider a gradient-based optimizer, where the server in iteration t is interested in computing $\mathbf{g}^t = \sum_n \omega_n \mathbf{g}_n^t$ with $\mathbf{g}_n^t$ denoting the gradient of $F_n(\mathbf{w})$ at $\mathbf{w}^t$.

TOP-k Sparsification With gradient sparsification, worker n sends the sparsified gradients $\hat{\mathbf{g}}_n^t = \mathbf{s}_n^t \odot \mathbf{a}_n^t$ to the sever, where $\mathbf{s}_n^t \in \{0,1\}^J$ is the sparsification mask and $\mathbf{a}_n^t$ denotes the accumulated gradient computed by adding the local gradient $\mathbf{g}_n^t$ to the sparsification error $\boldsymbol{\epsilon}_n^t$ as defined in Section 1. The TOP-k mask is determined by selecting the k largest entries of $\mathbf{a}_n^t$, i.e., $\mathbf{s}_n^t = \mathrm{Top}_k(\mathbf{a}_n^t)$, where the *top k selector* $\mathrm{Top}_k(\cdot)$ is defined as follows: let the entries of $\boldsymbol{x} \in \mathbb{R}^J$ be sorted as $|x_{i_1}| \geq \ldots |x_{i_J}|$; then, the i-th entry of $\mathrm{Top}_k(\boldsymbol{x})$ is

$$\mathrm{Top}_k(\boldsymbol{x})_{[i]} = \begin{cases} 1 & \text{if } i \in \{i_1, \ldots, i_k\} \\ 0 & \text{elsewhere.} \end{cases} \tag{5}$$

Upon receiving the sparsified gradients $\hat{\mathbf{g}}_n^t = \mathbf{s}_n^t \odot \mathbf{a}_n^t$, the server estimates the global gradient as $\mathbf{g}^t = \sum_n \omega_n \hat{\mathbf{g}}_n^t$. Note that the communication reduction by gradient sparsification is achieved at the expense of an extra index transmission per symbol. However, the index can be losslessly represented by $\log J$ bits, so its communication can be neglected.

3. BAYESIAN GRADIENT SPARSIFICATION

Though intuitive, there is no existing study that examines the optimality of TOP-k sparsification. In this section, we develop a stochastic framework for gradient sparsification by interpreting it as a Bayesian inference problem. We then show that, despite its effectiveness, TOP-k is not the optimal approach in this stochastic framework. We derive the REG-TOP-k algorithm by characterizing the optimal sparsification scheme in the Bayesian sense. Due to the lack of space, we only present the key steps in this section. Detailed derivations are skipped and left for the extended version of the paper.

3.1. Bayesian-Optimal Sparsification

Let us start with a thought experiment in which a genie provides each worker information about the aggregated gradient of the other workers. For entry $j \in [J]$, worker n knows in advance the weighted sum gradient of the other workers, denoted by $z_{n[j]}^t$, which satisfies $a_{[j]}^t = \omega_n a_{n[j]}^t + z_{n[j]}^t$, where $a_{[j]}^t$ is the j-th aggregated entry *when the workers apply no sparsification*, i.e., $a_{[j]}^t = \sum_n \omega_n a_{n[j]}^t$. Worker n decides to transmit $a_{n[j]}^t$, only if $a_{[j]}^t$ is within the top k gradient entries. In other words, given that the workers know $z_{n[j]}^t$, they apply TOP-k directly on the average gradient. We refer to this idealized approach as *global* TOP-k. It is readily seen that global TOP-k is in practice infeasible, as worker n does not have access to $z_{n[j]}^t$.

Statistical Global TOP-k Although worker n does not know $z_{n[j]}^t$, it has partial access to $z_{n[j]}^\ell$ for $\ell < t$ through the global gradients collected in previous iterations. This can be used to estimate $z_{n[j]}^t$, i.e., the workers use the *information collected through time* to apply the global TOP-k *statistically*. In the Bayesian framework, we can formulate a *statistical global* TOP-k via the following principle MAP problem:

Definition 1 (Principle MAP Problem)**.** *Let $\mathbb{T}_k^t$ denote the set of k largest entries of $|\mathbf{a}^t|$. Worker n determines its posterior probabilities*

$$P_{n[j]}^t = \Pr\left\{ j \in \mathbb{T}_k^t \,\middle|\, \mathbf{a}_n^t, \left\{ \mathbf{a}_n^\ell, \mathbf{g}^\ell : \ell < t-1 \right\} \right\} \tag{6}$$

for $j \in [J]$, where $\mathbf{g}^\ell$ is the aggregated gradient in iteration $\ell < t$ that is already known to all workers.[1] *Worker n then selects the k entries with the largest posteriors.*

To solve the principle MAP problem, we invoke the Bayes rule. The posterior probability $P_{n[j]}^t$ is expanded as

$$P_{n[j]}^t = \Pr\left\{ j \in \mathbb{T}_k^t \,\middle|\, \mathbf{a}_n^t, \left\{ \mathbf{a}_n^\ell, \mathbf{g}^\ell : \ell < t \right\} \right\} \tag{7a}$$

$$= \mathcal{L}_{n[j]}^t \Pr\left\{ j \in \mathbb{T}_k^t \,\middle|\, \mathbf{a}_n^t \right\} \tag{7b}$$

where $\Pr\{ j \in \mathbb{T}_k^t \,|\mathbf{a}_n^t\}$ is the *prior belief*, i.e., the prior probability of $j \in \mathbb{T}_k^t$ based on the local gradient of worker n, and $\mathcal{L}_{n[j]}^t$ denotes the *likelihood* given by

$$\mathcal{L}_{n[j]}^t \propto p\left(\mathbf{a}_n^\ell, \mathbf{g}^\ell : \ell < t \middle| \, j \in \mathbb{T}_k^t, \mathbf{a}_n^t \right). \tag{8}$$

Here, we use notation $p(\cdot)$ to refer to the probability density function (PDF).

TOP-k in Bayesian Framework TOP-k can be seen as a mismatched form of the principle MAP sparsifier under a specific prior belief. To see this, let us consider the following prior probability, to which we refer to as TOP-k *prior belief*.

Definition 2 (TOP-k Prior Belief)**.** *Given the accumulated gradient $\mathbf{a}_n^t$, the probability of entry j being among the top k entries of $\mathbf{g}^t$ is proportional to $|a_{n[j]}^t|$, i.e.,*

$$\Pr\left\{ j \in \mathbb{T}_k^t \,\middle|\, \mathbf{a}_n^t \right\} = \frac{|a_{n[j]}^t|}{\|\mathbf{a}_n^t\|_1}. \tag{9}$$

With this prior belief, the MAP sparsifier reduces to

$$\underset{j}{\mathrm{argmax}^k}\, P_{n[j]}^t = \underset{j}{\mathrm{argmax}^k}\, \mathcal{L}_{n[j]}^t |a_{n[j]}^t|, \tag{10}$$

where $\mathrm{argmax}^k{}_j\, x_j$ returns the k largest entries of the sequence $\{x_j : j \in [J]\}$ for $J \geq k$.

[1] Note that in distributed SGD, the server broadcasts either $\mathbf{g}^t$ or $\mathbf{w}^{t+1}$ to all workers at iteration t. In the latter case, the workers can recover the gradient as $\mathbf{g}^t = \left(\mathbf{w}^{t+1} - \mathbf{w}^t\right)/\eta^t$.

Comparing (10) with TOP-k, one can readily conclude that TOP-k *is a MAP sparsifier whose likelihood* $\mathcal{L}^t_{n[j]}$ *is uniform*. This is however a mismatched assumption. In fact, TOP-k simply ignores the information collected in previous iterations and infers the dominant entries solely based on the local accumulated gradients.

REGTOP-k Sparsification It is not straightforward to determine the exact expression for likelihood $\mathcal{L}^t_{n[j]}$, since the statistical model of its forward probability problem is not completely known. In the sequel, we approximate it in the large-system limit $J \to \infty$ under some simplifying assumptions.

We start the derivations by finding an alternative expression for the posterior probability $P^t_{n[j]}$:

Proposition 1. *The posterior* $P^t_{n[j]}$ *is computed as*

$$P^t_{n[j]} = \int_{\mathbb{F}^k_j} q_n(\mathbf{a}^t) \mathrm{d}\mathbf{a}^t,$$

where $\mathbb{F}^k_j = \left\{ \boldsymbol{x} \in \mathbb{R}^J : x_j \in \mathrm{argmax}^k{}_i \, x_i \right\}$ *with* x_i *denoting the* i*-th entry of* $\boldsymbol{x}$*, and* $q_n(\mathbf{a}^t)$ *is*

$$q_n(\mathbf{a}^t) = p\left(\mathbf{a}^t \left| \mathbf{a}^t_n, \left\{ \mathbf{a}^\ell_n, \mathbf{g}^\ell : \ell < t \right\} \right.\right). \tag{11}$$

Proof. The proof is given by standard marginalization and use of the fact that $\mathbf{a}^t_n, \left\{ \mathbf{a}^\ell_n, \mathbf{g}^\ell : \ell < t \right\} \to \mathbf{a}^t \to \mathbb{T}^t_k$ form a Markov chain.[2] Details are left for the extended version. □

Using this alternative form, we can write

$$\mathcal{L}^t_{n[j]} \propto \frac{1}{|a^t_{n[j]}|} \int_{\mathbb{F}^k_j} q_n(\mathbf{a}^t) \mathrm{d}\mathbf{a}^t. \tag{12}$$

The likelihood calculation is hence reduced to the characterization of the conditional distribution $q_n\left(\mathbf{a}^t\right)$. To describe $q_n(\mathbf{a}^t)$, we need to specify the stochastic model that describe the relation between $\mathbf{a}^t$ and $\mathbf{a}^t_n, \left\{ \mathbf{a}^\ell_n, \mathbf{g}^\ell : \ell < t \right\}$. Considering the gradient aggregation strategy at the server, we can write $\mathbf{a}^t = \omega_n \mathbf{a}^t_n + \mathbf{z}^t_n$, where $\mathbf{z}^t_n$ denotes the vector form of $z^t_{n[j]}$ as defined above. We now describe the time evolution of $\mathbf{z}^t_n$ via an additive model, i.e., $\mathbf{z}^t_n = \mathbf{z}^{t-1}_n + \boldsymbol{\xi}^t_n$ for some *innovation* $\boldsymbol{\xi}^t_n$, which we treat as a random variable. It is worth mentioning that the innovation describes the difference between two consecutive local gradients, and hence its distribution depends on the dataset.

We now divide the index set $[J]$ into two subsets, namely $\mathbb{S}^{t-1}_n$ and its complement, where $\mathbb{S}^{t-1}_n$ denotes the support of the previous sparsification mask of worker n $\mathbf{s}^{t-1}_n$. We now focus on $j \in \mathbb{S}^{t-1}_n$. For this set, worker n has already received the aggregated gradient in the last iteration, i.e., $a^t_{[j]} = \mathrm{g}^t_{[j]}$. We can hence write

$$z^{t-1}_{n[j]} = \mathrm{g}^{t-1}_{[j]} - \omega_n a^{t-1}_{n[j]} = \omega_n a^t_{n[j]} \Delta^t_{n[j]} \tag{13}$$

where we define $\Delta^t_{n[j]} = (\mathrm{g}^{t-1}_{[j]} - \omega_n a^{t-1}_{n[j]}) / \omega_n a^t_{n[j]}$, and refer to it as the *posterior distortion*. We now write $a^t_{[j]} = \bar{\mathrm{g}}^t_{n[j]} + \xi^t_{n[j]}$, where $\bar{\mathrm{g}}^t_{n[j]} = \omega_n a^t_{n[j]}(1 + \Delta^t_{n[j]})$.

[2]Note that $\mathbb{T}^t_k$ is defined in Definition 1.

To proceed with the exact computation of likelihood $\mathcal{L}^t_{n[j]}$, we require an explicit expression of the distribution of $\xi^t_{n[j]}$. This is however analytically infeasible, as it depends on several problem-specific factors, e.g., data distribution, learning rate, and loss function. We hence invoke large-deviations arguments and the method of types to asymptotically approximate the likelihood for a class of settings in which $\boldsymbol{\xi}^t_n$ is distributed symmetrically around zero and is fast decaying. We skip details due to lack of space.

Proposition 2. *Let* $p_j\left(\xi\right)$ *denote the distribution of the* j*-th entry in* $\boldsymbol{\xi}^t_n$*, i.e.,* $\xi^t_{n[j]}$*. Assume that* $p_j\left(\xi\right)$ *represents a symmetric zero-mean distribution, and that for* $j \in [J]$*, given a small* δ*, there exists a small* ε*, such that*

$$\int_{-\varepsilon|a^t_{n[j]}|}^{\varepsilon|a^t_{n[j]}|} p_j\left(\xi\right) \mathrm{d}\xi \geq 1 - \delta. \tag{14}$$

Moreover, assume that the entries of $\boldsymbol{\xi}^t_n$ *are independent. Then, as* $J \to \infty$*, we have*

$$\int_{\mathbb{F}^k_j} q_n(\mathbf{a}^t) \mathrm{d}\mathbf{a}^t \propto |a^t_{n[j]}| \begin{cases} u_\mu(|1 + \Delta^t_{n[j]}|) & j \in \mathbb{S}^{t-1}_n \\ C & j \notin \mathbb{S}^{t-1}_n \end{cases},$$

for some constant C*, and a non-decreasing* $u_\mu : \mathbb{R}^+ \mapsto \mathbb{R}^+$ *that approximates the sign function and is tuned by a positive parameter* μ*.*

Proof. The proof follows large-deviations arguments. We skip the details here due to lack of space and leave them for the extended version of the paper. □

From Proposition 2, we can conclude that under the given assumptions, the likelihood for large J is approximated by

$$\mathcal{L}^t_{n[j]} = u_\mu(|1 + \Delta^t_{n[j]}|). \tag{15}$$

where, for $j \notin \mathbb{S}^{t-1}_n$, we define $\Delta^t_{n[j]} = Q$ for some Q satisfying $u_\mu(|1 + Q|) = C$. The Bayesian-optimal sparsification is hence approximated by substituting this expression into the principle MAP sparsifier.

3.2. REGTOP-k Algorithm

The asymptotic expression (15) for the likelihood implies that for Bayesian-optimal sparsification, the top k selector should be applied on a *regularized accumulated gradient*, i.e.,

$$\tilde{\mathbf{a}}^t_n = \mathbf{a}^t_n \odot u_\mu(|1 + \boldsymbol{\Delta}^t_n|), \tag{16}$$

where $\boldsymbol{\Delta}^t_n \in \mathbb{R}^J$ collects the posterior distortions $\Delta^t_{n[j]}$ for $j \in [J]$. We now follow the definition of $u_\mu\left(\cdot\right)$ in Proposition 2 and set[3] $u_\mu\left(x\right) = \tanh\left(x/\mu\right)$, where we can treat μ as a hyperparameter. This concludes the REGTOP-k algorithm, whose pseudo code is given in Algorithm 1.

Discussions on Algorithm 1 REGTOP-k starts by applying standard TOP-k in the initial iteration. From $t = 1$, worker n after calculating its accumulated gradient $\mathbf{a}^t_n$ determines the *posterior distortion* $\boldsymbol{\Delta}^t_n$ for those entries that were sent

[3]Note that other choices, e.g., sigmoid function, are also valid.

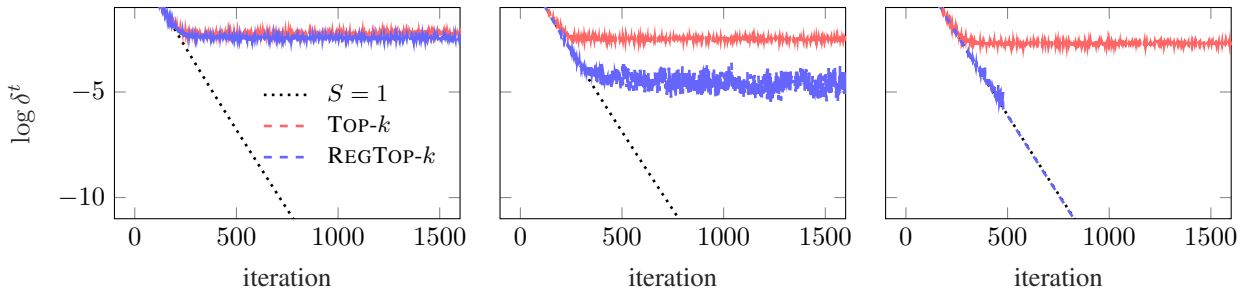

Fig. 2. REGTOP-k versus TOP-k sparsification for three sparsity factors. Left: $S = 0.4$; middle: $S = 0.5$; right: $S = 0.6$.

Algorithm 1 REGTOP-k Sparsification at Worker n

Initialization: Set $\boldsymbol{\epsilon}_n^0 = \{0\}^J$ and some $Q, \mu > 0$

1: **for** $t = 0$ **do** Sparsify via TOP-k and collect $\mathbf{g}^0$ **end for**
2: **for** $t \geq 1$ **do**
3: Determine local gradient $\mathbf{g}_n^t$ at global model $\mathbf{w}^t$
4: Determine *accumulated gradient* as $\mathbf{a}_n^t = \boldsymbol{\epsilon}_n^t + \mathbf{g}_n^t$
5: Determine the *posterior distortion* as

$$\boldsymbol{\Delta}_n^t = \mathbf{s}_n^{t-1}\left[\left(\mathbf{g}^{t-1} - \omega_n \mathbf{a}_n^{t-1}\right) \oslash \omega_n \mathbf{a}_n^t\right] + Q\left(1 - \mathbf{s}_n^{t-1}\right)$$

6: Find the sparsification mask as

$$\mathbf{s}_n^t = \mathrm{Top}_k\left(\mathbf{a}_n^t \odot \tanh\left(\frac{|1 + \boldsymbol{\Delta}_n^t|}{\mu}\right)\right)$$

7: Sparsify as $\hat{\mathbf{g}}_n^t = \mathbf{s}_n^t \odot \mathbf{a}_n^t$ and send to server
8: Update the *sparsification error* as $\boldsymbol{\epsilon}_n^{t+1} = \mathbf{a}_n^t - \hat{\mathbf{g}}_n^t$
9: **end for**

in the previous iterations, i.e., j for which $s_{n[j]}^t = 1$. The impact of the regularization applied by REGTOP-k can be intuitively illustrated by considering the following extreme cases: (1) As $\mu \to 0$, the regularizer converges to 1, and hence REGTOP-k reduces to standard TOP-k. TOP-k is hence a special case of REGTOP-k with no regularization. (2) Assume the j-th local gradient entries of all workers are large in amplitude but cancel out after aggregation.[4] In this case, after the initial aggregation, worker n determines its posterior distortion as $\Delta_{n[j]}^t = 0 - a_{n[j]}^{t-1}/a_{n[j]}^t = -1$. This leads to the j-th regularized accumulated gradient entry damped to zero and prevents its selection in iteration t. This way the selection frequency of constructively-aggregated gradient entries with small amplitudes is increased, and hence large scaling of learning rate is avoided.

4. NUMERICAL VALIDATION

We validate REGTOP-k through numerical experiments. We start with the problem of linear regression. For this learning problem, we can track the optimal solution and hence evaluate the *distance between the converging point and the global optimum*, which is a strong notion of convergence. We further give the results for training ResNet-18 on CIFAR-10. Throughout the numerical experiments, REGTOP-k is compared against TOP-k and the distributed gradient descent without sparsification. As mentioned, the existing extensions to TOP-k, e.g., [7, 8, 9], do not revise the derivation of sparsification mask. This means that with respect to impact of large learning rate scaling, these approaches perform identically to TOP-k, and thus our comparison with TOP-k suffices.

4.1. Linear Regression

We consider a distributed setting with 20 workers that solve a distributed linear regression problem via distributed SGD. Each worker has 500 labeled data-points of dimension 100. The workers employ the method of least-squares (LS). The global loss is determined by arithmetic averaging, i.e., $\omega_n = 1/N$. The training is performed via full-batch gradient descent. The learning rate is kept fixed at $\eta = 10^{-2}$.

Synthetic Dataset Generation The local datasets are generated independently via a Gaussian linear model. For worker n, data-points are sampled independently from an i.i.d. zero-mean and unit-variance Gaussian process. To label these data-points, we generate the ground truth model $\mathbf{t}_n \in \mathbb{R}^J$ i.i.d. according to a Gaussian distribution with mean u_n and variance h^2. The mean u_n is further sampled from a Gaussian process with mean U and variance σ^2. The labels are determined via the linear model as $y_{n,i} = \mathbf{x}_{n,i}^\mathsf{T}\mathbf{t}_n + \varepsilon_{n,i}$, where $\varepsilon_{n,i}$ is a zero-mean Gaussian perturbation with variance ϵ.

Numerical Results We evaluate performance by tracking the *optimality gap*: in iteration t, we compute the difference between the globally updated model parameter, i.e., $\mathbf{w}^t$, and the global minimizer $\mathbf{w}^\star$, i.e., $\delta^t = \|\mathbf{w}^t - \mathbf{w}^\star\|$. We denote the sparsification factor by $S = k/J$.

Figure 2 sketches the optimality gap δ^t in the logarithmic scale against the number of iterations for the three algorithms. Here, we set $U = 0$, $\sigma^2 = 5$, $h^2 = 1$ and $\epsilon = 0.5$. The figure shows the convergence for three sparsity factors, namely $S = 0.4$, $S = 0.5$, and $S = 0.6$. As observed, the REGTOP-k algorithm starts to track the (non-sparsified) distributed SGD at $S = 0.6$ while TOP-k remains at a certain distance from the optimal solution. This behavior can be intuitively explained as follows: for both sparsification approaches, the aggregation of large local gradient entries gradually moves the initial point towards the global optimum. At a certain vicinity of the optimum, the impact of smaller gradient entries in convergence becomes more dominant. TOP-k selects these entries only after large error accumulation, which due to learning rate scaling leads to oscillation around the optimum at a fixed distance. To avoid such oscillation, TOP-k would need signifi-

[4] Recall the toy-example in Section 1.

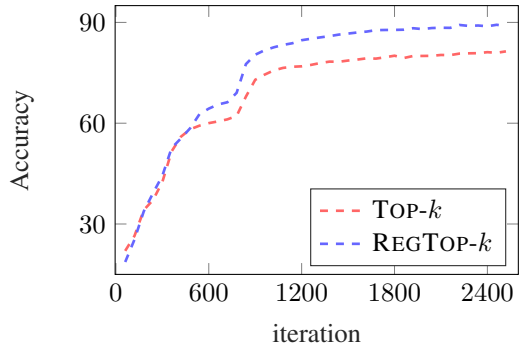

Fig. 3. ResNet-18 on CIFAR-10 with 0.1% sparsification.

cant damping of the learning rate that can slow convergence. REGTOP-k, however, selects the small (but dominant) gradient entries at lower error accumulation levels, which prevents large learning rate scaling.

4.2. Training ResNet-18 on CIFAR-10

We now employ REGTOP-k to train ResNet-18 on CIFAR-10, with data-points distributed evenly among $N = 8$ workers. The clients compute their local gradients over mini-batches of size 20. The aggregation is performed by arithmetic averaging. The learning rate is set to $\eta = 0.01$, and the clients sparsify with $S = 0.001$, i.e., 0.1% sparsification.

Numerical Results Figure 3 shows the validation accuracy against the number of iterations for both TOP-k and REGTOP-k. To keep the comparison fair, we have considered the same initialization of the global model for both algorithms and identical batch samplers. As the figure shows, after the first 600 iterations, the model trained by REGTOP-k sparsification starts to give strictly higher accuracy as compared with the one trained by standard TOP-k. As the number of iterations surpasses 1500, the difference between the accuracy values exceeds 8%. This considerable gain indicates that REGTOP-k can substantially improve the efficiency of sparsification in real-world applications.

5. CONCLUSIONS

Information collected during training can be used for efficient compression of local gradients in distributed learning. We have invoked this idea and developed a Bayesian framework to regularize the TOP-k sparsification algorithm. Numerical investigations validate our derivations. REGTOP-k can track the performance of non-sparsified distributed learning at significantly lower sparsity factors than TOP-k. Interestingly, this gain is achieved with no considerable increase in computation complexity. This work can be extended in various respects. Most naturally, the proposed scheme can be extended to adaptive sparsification frameworks.

6. REFERENCES

[1] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli, "The convergence of sparsified gradient methods," in *Proc. NeurIPS*, 2018.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proc. Conf. on Computer Vis. and Pattern Recog.*, 2016.

[3] Nikko Ström, "Scalable distributed DNN training using commodity GPU cloud computing," in *Proc. An. Conf. Int. Speech Commun. Association*, 2015.

[4] Nikoli Dryden, Tim Moon, Sam Ade Jacobs, and Brian Van Essen, "Communication quantization for data-parallel training of deep neural networks," in *Proc. Worksh. Machine Learning HPC Environ.*, 2016.

[5] Alham Fikri Aji and Kenneth Heafield, "Sparse communication for distributed gradient descent," in *Proc. Conf. Empirical Meth. Natural Language Process.*, 2017.

[6] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *Proc. ICLR*, 2018.

[7] Atal Sahu, Aritra Dutta, Ahmed M Abdelmoniem, Trambak Banerjee, Marco Canini, and Panos Kalnis, "Rethinking gradient sparsification as total error minimization," in *Proc. NeurIPS*, 2021.

[8] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *Proc. ICLR*, 2018.

[9] Chia-Yu Chen, Jungwook Choi, Daniel Brand, Ankur Agrawal, Wei Zhang, and Kailash Gopalakrishnan, "Adacomp: Adaptive residual gradient compression for data-parallel distributed training," in *Proc. AAAI Conf. AI*, 2018.

[10] Pengchao Han, Shiqiang Wang, and Kin K Leung, "Adaptive gradient sparsification for efficient federated learning: An online learning approach," in *Proc. Int. Conf. Dist. Computing Systems (ICDCS)*, 2020.

[11] Hongyi Wang, Scott Sievert, Shengchao Liu, Zachary Charles, Dimitris Papailiopoulos, and Stephen Wright, "Atomo: Communication-efficient learning via atomic sparsification," in *Proc. NeurIPS*, 2018.

[12] Zhaorui Zhang and Choli Wang, "MIPD: An adaptive gradient sparsification framework for distributed DNNs training," *IEEE Trans. Parallel and Dist. Sys.*, vol. 33, no. 11, pp. 3053–3066, 2022.

[13] Chia-Yu Chen, Jiamin Ni, Songtao Lu, Xiaodong Cui, Pin-Yu Chen, Xiao Sun, Naigang Wang, Swagath Venkataramani, Vijayalakshmi Viji Srinivasan, Wei Zhang, et al., "Scalecom: Scalable sparsified gradient compression for communication-efficient distributed training," in *Proc. NeurIPS*, 2020.

[14] Ahmed M Abdelmoniem, Ahmed Elzanaty, Mohamed-Slim Alouini, and Marco Canini, "An efficient statistical-based gradient compression technique for distributed training systems," *Proc. Machine Learnig and Systems (MLSys)*, vol. 3, 2021.