Gradskip

Grik Tadevosyan, Maksim Osipenko

December 2024

1 Introduction

Federated Learning (FL) is an emerging distributed machine learning paradigm where diverse data holders or clients (e.g., smart watches, mobile devices, laptops, hospitals) collectively aim to train a single machine learning model without revealing local data to each other or the orchestrating central server. Training such models amounts to solving federated optimization problems of the form

$$\min_{x\in\mathbb{R}^d}\left\{f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x)\right\}, \tag{1}$$

where d is the (typically large) number of parameters of the model $x \in \mathbb{R}^d$ we aim to train, n is the (potentially large) total number of devices in the federated environment. The loss is denoted by $f_i(x)$, associated with the data $\mathcal{D}_i$ stored on client $i \in [n] := \{1, 2, \ldots, n\}$. Formally, our goal is to minimize the overall loss denoted by $f(x)$.

Due to their efficiency, gradient-type methods with its numerous extensions is by far the most dominant method for solving (1) in practice.

The simplest implementation of gradient descent for federated setup requires all workers $i \in [n]$ in each time step $t \geq 0$ to *(i)* compute local gradient $\nabla f_i(x_t)$ at the current model x_t, *(ii)* update the current global model x_t using locally computed gradient $\nabla f_i(x_t)$ via (2) with some step size $\gamma > 0$, *(iii)* average the updated local models $\hat{x}_{i,t+1}$ via (3) to get the new global model x_{t+1}.

$$\hat{x}_{i,t+1} = x_t - \gamma\nabla f_i(x_t), \tag{2}$$

$$x_{t+1} = \frac{1}{n}\sum_{i=1}^{n}\hat{x}_{i,t+1}. \tag{3}$$

Distinguishing challenges that characterize FL as a separate distributed training setup and dictate adjustments to the training algorithm include *high communication costs*, *heterogeneous data distribution* and *system heterogeneity* across clients. Next, we briefly discuss these challenges and possible algorithmic workarounds.

1.1 Communication costs

It has been repeatedly observed and advocated that in federated optimization, communication costs dominate and can be a primary bottleneck because of slow and unreliable wireless links between clients and the central server. The communication step (3) between clients cannot be taken out entirely as otherwise, all clients would keep training on local data only, resulting in a poor model due to limited local data.

A simple trick to reduce communication costs is to perform the costly synchronization step (3) infrequently, allowing multiple local gradient steps (2) in each communication round. This trick will be called as *local gradient methods* further. However, until very recently, theoretical guarantees on the convergence rates of local gradient methods were worse than the rate of classical gradient descent, which synchronizes after every gradient step.

In some works a novel local gradient method, called ProxSkip, was proposed. It performs a random number of local gradient steps before each communication (alternation between local training and synchronization is probabilistic) and guarantees strong communication acceleration properties. First, they reformulate the problem (1) into an equivalent regularized consensus problem of the form

$$\min_{x_1,\ldots,x_n\in\mathbb{R}^d}\left\{\frac{1}{n}\sum_{i=1}^{n} f_i(x_i)+\psi(x_1,\ldots,x_n)\right\}, \quad \psi(x_1,\ldots,x_n):=\begin{cases}0 & \text{if } x_1=\cdots=x_n\\ +\infty & \text{otherwise}\end{cases}, \tag{4}$$

where communication between the clients and averaging local models $x_1,\ldots,x_n$ is encoded as taking the proximal step with respect to ψ, i.e., $\text{prox}_{\psi}([x_1\ldots x_n]^\top)=[\bar{x}\ldots\bar{x}]^\top$, where $\bar{x}=:\frac{1}{n}\sum_{i=1}^{n}x_i$. With this reformulation, ProxSkip method of performs the proximal (equivalently averaging) step with small probability $p=1/\sqrt{\kappa}$, where κ is the condition number of the problem. Then the key result of the method for smooth and strongly convex setup is $\mathcal{O}(\kappa\log 1/\epsilon)$ iteration complexity with $\mathcal{O}\left(\sqrt{\kappa}\log 1/\epsilon\right)$ communication rounds to achieve $\epsilon>0$ accuracy. Follow-up works extend the method to variance reduced gradient methods, randomized application of proximal operator, and accelerated primal-dual algorithms. Our work was inspired by development of this new generation of local gradient methods, which we detail shortly.

An orthogonal approach utilizes communication compression strategies on the transferred information. Informally, instead of communicating full precision models infrequently we might communicate compressed version of the local model in each iteration via an application of lossy compression operators. Such strategies include sparsification, quantization, sketching and low-rank approximation.

1.2 Statistical heterogeneity

Because of the decentralized nature of the training data, distributions of local datasets can vary from client to client. This heterogeneity in data distributions poses an additional challenge since allowing multiple local steps would make the local models deviate from each other, an issue widely known as *client drift*. On

the other hand, if training datasets are identical across the clients (commonly referred to as homogeneous setup), then the mentioned drifting issue disappears and the training can be done without any communication whatsoever. Now, if we interpolate between these two extremes, then under some data similarity conditions (which typically expressed as gradient similarity conditions) multiple local gradient steps should be useful. In fact, initial theoretical guarantees of local gradient methods utilize such assumptions.

In the fully heterogeneous setup, client drift reduction techniques were designed and analyzed to mitigate the adverse effect of local model deviations. A very close analogy is variance reduction techniques called error feedback mechanisms for the compression noise added to lessen the number of bits required to transfer.

1.3 System heterogeneity

Lastly, system heterogeneity refers to the diversity of clients in terms of their computation capabilities or the amount of resources they are willing to use during the training. In a typical FL setup, all participating clients must perform the same amount of local gradient steps before each communication. Consequently, a highly heterogeneous cluster of devices results in significant and unexpected delays due to slow clients or stragglers.

One approach addressing system heterogeneity or dealing with slow clients is client selection strategies. Basically, client sampling can be organized in such a way that slow clients do not delay the global synchronization and clients with similar computational capabilities are sampled in each communication round.

In contrast to the above strategy, we propose that clients perform as few or as many local steps as their local resources allow. In other words, we consider the full participation setup where each client decides how much local computation to perform before communication. Informally, slow clients do less local work than fast clients, and during the synchronization of local trained models, the slowdown caused by the stragglers will be minimized.

2 Algorithms brief description

2.1 Gradskip brief description

The authors design a new local gradient-type method for distributed optimization with communication and computation constraints. The proposed GradSkip (see Algorithm 1) is an extension of recently developed ProxSkip method, which was the first method showing communication acceleration property of performing multiple local steps without any data similarity assumptions. GradSkip method inherits the same accelerated communication complexity from ProxSkip, while further improving computational complexity allowing clients to terminate their local gradient computations independently from each other.

The key technical novelty of the proposed algorithm is the construction of auxiliary shifts $\hat{h}_{i,t}$ to handle gradient skipping for each client $i \in [n]$. GradSkip also maintains shifts $h_{i,t}$ initially introduced in ProxSkip to handle communication skipping across the clients. GradSkip converges linearly in strongly convex and smooth setup, has the same $\mathcal{O}(\sqrt{\kappa_{\max}} \log 1/\epsilon)$ accelerated communication complexity as ProxSkip, and requires clients to compute (in expectation) at most $\min(\kappa_i, \sqrt{\kappa_{\max}})$ local gradients in each communication round, where κ_i is the condition number for client $i \in [n]$ and $\kappa_{\max} = \max_i \kappa_i$. Thus, for GradSkip, clients with well-conditioned problems $\kappa_i < \sqrt{\kappa_{\max}}$ perform much less local work to achieve the same convergence rate of ProxSkip, which assumes $\sqrt{\kappa_{\max}}$ local steps on average for all clients.

2.2 GradSkip+ brief description

Next, the authors generalize the construction and the analysis of GradSkip by extending it in two directions: handling optimization problems with arbitrary proximal regularizer and incorporating general randomization procedures using unbiased compression operators with custom variance bounds. With such enhancements, there is the second method, GradSkip+, which recovers several methods in the literature as a special case, including the standard proximal gradient descent (ProxGD), ProxSkip, RandProx-FB and GradSkip.

3 Gradskip alogrithm

3.1 Algorithm structure

For the sake of presentation, the progress of the algorithm is described by using two variables $x_{i,t}, \hat{x}_{i,t}$ for the local models and two variables $h_{i,t}, \hat{h}_{i,t}$ for the local gradient shifts. Essentially, we want to maintain two variables for the local models since clients get synchronized infrequently. The shifts $h_{i,t}$ are designed to reduce the client drift caused by the statistical heterogeneity. Finally, we introduce an auxiliary shifts $\hat{h}_{i,t}$ to take care of the different number of local steps. The GradSkip method is formally presented in Algorithm 1.

As an initialization step, we choose probability $p > 0$ to control communication rounds, probabilities $q_i > 0$ for each client $i \in [n]$ to control local gradient steps and initial control variates (or shifts) $h_{i,0} \in \mathbb{R}^d$ to control the client drift. Besides, we fix the stepsize $\gamma > 0$ and assume that all clients commence with the same local model, namely $x_{1,0} = \cdots = x_{n,0} \in \mathbb{R}^d$. Then, each iteration of the method comprises two stages, the local stage and the communication stage, operating probabilistically. Specifically, the probabilistic nature of these stages is the following. The local stage requires computation only with some predefined probability; otherwise, the stage is void. Similarly, the communication stage requires synchronization between all clients only with probability p; otherwise, the stage is void. In the local stage (lines 5–7), all clients $i \in [n]$ in parallel

Algorithm 1 GradSkip

Input: stepsize $\gamma > 0$, synchronization probability p, probabilities $q_i > 0$ controlling local steps, initial local iterates $x_{1,0} = \cdots = x_{n,0} \in \mathbb{R}^d$, initial shifts $h_{1,0}, \ldots, h_{n,0} \in \mathbb{R}^d$, total number of iterations $T \geq 1$
for $t = 0, 1, \ldots, T-1$ **do**
  server: Flip a coin $\theta_t \in \{0, 1\}$ with $\text{Prob}(\theta_t = 1) = p$ ◇ Decide when to skip communication
  for all devices $i \in [n]$ **in parallel do**
    Flip a coin $\eta_{i,t} \in \{0, 1\}$ with $\text{Prob}(\eta_{i,t} = 1) = q_i$ ◇ Decide when to skip gradient steps.
    $\hat{h}_{i,t+1} = \eta_{i,t} h_{i,t} + (1 - \eta_{i,t}) \nabla f_i(x_{i,t})$ ◇ Update the local auxiliary shifts $\hat{h}_{i,t}$
    $\hat{x}_{i,t+1} = x_{i,t} - \gamma(\nabla f_i(x_{i,t}) - \hat{h}_{i,t+1})$ ◇ Update the local auxiliary iterate $\hat{x}_{i,t}$ via shifted gradient step
    if $\theta_t = 1$ **then**
      $x_{i,t+1} = \frac{1}{n} \sum_{j=1}^{n} \left(\hat{x}_{j,t+1} - \frac{\gamma}{p} \hat{h}_{j,t+1} \right)$ ◇ Average shifted iterates, but only very rarely!
    else
      $x_{i,t+1} = \hat{x}_{i,t+1}$ ◇ Skip communication!
    end if
    $h_{i,t+1} = \hat{h}_{i,t+1} + \frac{p}{\gamma}(x_{i,t+1} - \hat{x}_{i,t+1})$ ◇ Update the local shifts $h_{i,t}$
  end for
end for

update their local variables $(\hat{x}_{i,t+1}, \hat{h}_{i,t+1})$ using values $(x_{i,t}, h_{i,t})$ from previous iterate either by computing the local gradient $\nabla f_i(x_{i,t})$ or by just copying the previous values. Afterwards, in the communication stage (lines 8–13), all clients in parallel update their local variables $(x_{i,t+1}, h_{i,t+1})$ from $(\hat{x}_{i,t+1}, \hat{h}_{i,t+1})$ by either averaging across the clients or copying previous values.

3.2 Reduced local computation

Clearly, communication costs are reduced as the averaging step occurs only when $\theta_t = 1$ with probability p of our choice. However, it is not directly apparent how the computational costs are reduced during the local stage. Indeed, both options $\eta_{i,t} = 1$ and $\eta_{i,t} = 0$ involve the expression $\nabla f_i(x_{i,t})$ as if local gradients need to be evaluated in every iteration. And this is not the case.

The smaller probability q_i implies the sooner coin toss $\eta_{i,t} = 0$ in expectation, hence, the less amount of local computation for client i. Therefore, we can relax computational requirements of clients by adjusting these probabilities q_i and controlling the amount of local gradient computations.

Next, let us find out how the expected number of local gradient steps depends on probabilities p and q_i. Let Θ and H_i be random variables representing the number of coin tosses (Bernoulli trials) until the first occurrence of $\theta_t = 1$ and

Algorithm 2 GradSkip+

Parameters: stepsize $\gamma > 0$, compressors $\mathcal{C}_\omega \in \mathbb{B}^d(\omega)$ and $\mathcal{C}_{\mathbf{\Omega}} \in \mathbb{B}^d(\mathbf{\Omega})$.
Input: initial iterate $x_0 \in \mathbb{R}^d$, initial control variate $h_0 \in \mathbb{R}^d$, number of iterations $T \geq 1$.
for $t = 0, 1, \ldots, T-1$ **do**
  $\hat{h}_{t+1} = \nabla f(x_t) - (\mathbf{I} + \mathbf{\Omega})^{-1}\mathcal{C}_{\mathbf{\Omega}}\left(\nabla f(x_t) - h_t\right)$ ◇ Update the shift $\hat{h}_t$ via shifted compression
  $\hat{x}_{t+1} = x_t - \gamma(\nabla f(x_t) - \hat{h}_{t+1})$ ◇ Update the iterate $\hat{x}_t$ via shifted gradient step
  $\hat{g}_t = \frac{1}{\gamma(1+\omega)}\mathcal{C}_\omega\left(\hat{x}_{t+1} - \mathrm{prox}_{\gamma(1+\omega)\psi}\left(\hat{x}_{t+1} - \gamma(1+\omega)\hat{h}_{t+1}\right)\right)$ ◇ Estimate the proximal gradient
  $x_{t+1} = \hat{x}_{t+1} - \gamma\hat{g}_t$ ◇ Update the main iterate x_t
  $h_{t+1} = \hat{h}_{t+1} + \frac{1}{\gamma(1+\omega)}(x_{t+1} - \hat{x}_{t+1})$ ◇ Update the main shift h_t
end for

$\eta_{i,t} = 0$ respectively. Equivalently, $\Theta \sim \mathrm{Geo}(p)$ is a geometric random variable with parameter p, and $H_i \sim \mathrm{Geo}(1-q_i)$ are geometric random variables with parameter $1-q_i$ for $i \in [n]$. Notice that, within one communication round, i^{th} client performs $\min(\Theta, H_i)$ number of local gradient computations, which is again a geometric random variable with parameter $1-(1-(1-q_i))(1-p) = 1-q_i(1-p)$. The expected number of local gradient steps is $\mathbb{E}\left[\min(\Theta, H_i)\right] = 1/(1-q_i(1-p))$.

4 Gradskip+ alogrithm

4.1 Algorithm description

The first direction is the formulation of the optimization problem. As we discussed earlier, distributed optimization (1) with consensus constraints can be transformed into regularized optimization problem (4) in the lifted space. Thus, following, we consider the (lifted) problem[1]

$$\min_{x \in \mathbb{R}^d} f(x) + \psi(x), \tag{5}$$

where $f(x)$ is strongly convex and smooth loss, while $\psi(x)$ is closed, proper and convex regularizer (e.g., see (4)). The requirement we impose on the regularizer is that the proximal operator of ψ is a single-valued function that can be computed.

The second extension in GradSkip+ is the generalization of the randomization procedure of probabilistic alternations in GradSkip by allowing arbitrary unbiased compression operators with certain bounds on the variance. Let us formally define the class of compressors we will be working with.

[1]To be precise, the lifted problem is in $\mathbb{R}^{nd}$ as we stack all local variables $x_1, \ldots, x_n \in \mathbb{R}^d$ into one.

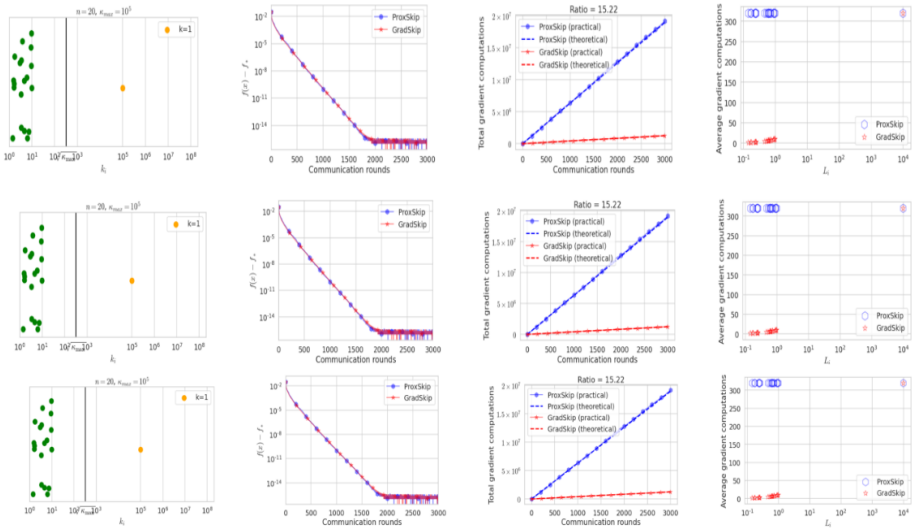

Figure 1: Comparision the algorithms

Similar to GradSkip, we maintain two variables x_t, $\hat{x}_t$ for the model, and two variables h_t, $\hat{h}_t$ for the gradient shifts in GradSkip+. Initial values $x_0 \in \mathbb{R}^d$ and $h_0 \in \mathbb{R}^d$ can be chosen arbitrarily. In each iteration, GradSkip+ first updates the auxiliary shift $\hat{h}_{t+1}$ using the previous shift h_t and gradient $\nabla f(x_t)$ (line 4). This shift $\hat{h}_{t+1}$ is then used to update the auxiliary iterate x_t via shifted gradient step (line 5). Then we estimate the proximal gradient $\hat{g}_t$ (line 6) in order to update the main iterate x_{t+1} (line 7). Lastly, we complete the iteration by updating the main shift h_t (line 8). See Algorithm 2 for the formal steps.

5 Experiments

To test the performance of GradSkip and illustrate theoretical results, the authors use the classical logistic regression problem. The loss function for this model has the following form:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{m_i}\sum_{j=1}^{m_i}\log\left(1+\exp\left(-b_{ij}a_{ij}^{\top}x\right)\right) + \frac{\lambda}{2}\|x\|^2,$$

where n is the number of clients, m_i is the number of data points per worker, $a_{ij} \in \mathbb{R}^d$ and $b_{ij} \in -1, +1$ are the data samples, and λ is the regularization parameter. The results of the comparison between ProxSkip and GradSkip and ProxSkip and Gradskip+ algorithms are depicted in figures 1 and 2 respectively.

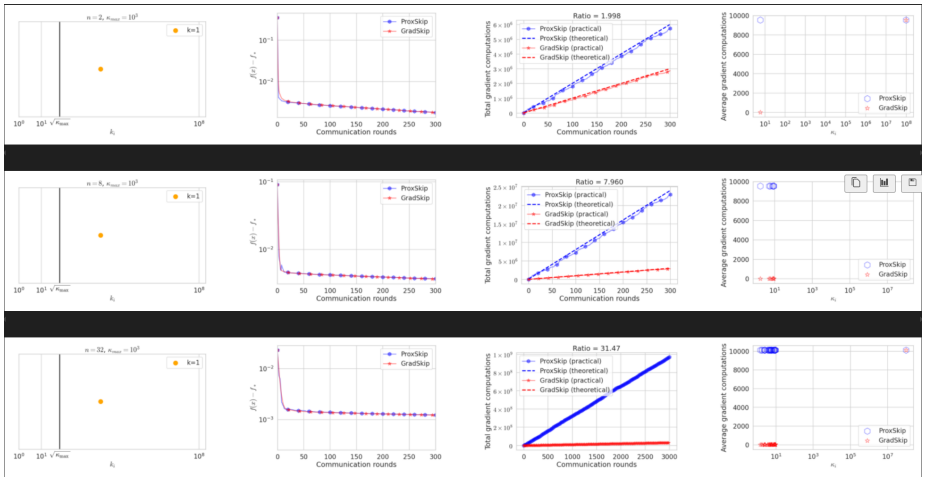

Figure 2: Comparision the algorithms