Statistically Preconditioned Accelerated Gradient Method for Distributed Optimization

Hadrien Hendrikx 1 Lin Xiao 2 Sébastien Bubeck 2 Francis Bach 1 Laurent Massoulié 1

Abstract
We consider the setting of distributed empirical risk minimization where multiple machines compute the gradients in parallel and a centralized server updates the model parameters. In order to reduce the number of communications required to reach a given accuracy, we propose a preconditioned accelerated gradient method where the preconditioning is done by solving a local optimization problem over a subsampled dataset at the server. The convergence rate of the method depends on the square root of the relative condition number between the global and local loss functions. We estimate the relative condition number for linear prediction models by studying uniform concentration of the Hessians over a bounded domain, which allows us to derive improved convergence rates for existing preconditioned gradient methods and our accelerated method. Experiments on real-world datasets illustrate the benefits of acceleration in the ill-conditioned regime.

1. Introduction
We consider empirical risk minimization problems of the form

$$\min_{x \in \mathbb{R}^d} \Phi(x) \triangleq F(x) + \psi(x),$$

where $F$ is the empirical risk over a dataset $\{z_1, \ldots, z_N\}$:

$$F(x) = \frac{1}{N} \sum_{i=1}^{N} \ell(x, z_i),$$

and $\psi$ is a convex regularization function. We incorporate smooth regularizations such as squared Euclidean norms $(\lambda/2)\|x\|^2$ into the individual loss functions $\ell(x, z_i)$, and leave $\psi$ mainly for non-smooth regularizations such as the $\ell_1$-norm or the indicator function of a constraint set.

In modern machine learning applications, the dataset is often very large and has to be stored at multiple machines. For simplicity of presentation, we assume $N = mn$, where $m$ is the number of machines and $n$ is the number of samples stored at each machine. Let $D_j = \{z^{(j)}_1, \ldots, z^{(j)}_n\}$ denote the dataset at machine $j$ and define the local empirical risk

$$f_j(x) = \frac{1}{n} \sum_{i=1}^{n} \ell(x, z^{(j)}_i), \quad j = 1, \ldots, m.$$ (3)

The overall empirical risk of Equation (2) can then be written as

$$F(x) = \frac{1}{m} \sum_{j=1}^{m} f_j(x) = \frac{1}{nm} \sum_{j=1}^{m} \sum_{i=1}^{n} \ell(x, z^{(j)}_i).$$

We assume that $F$ is $L_F$-smooth and $\sigma_F$-strongly convex over $\text{dom} \psi$, in other words,

$$\sigma_F I_d \preceq \nabla^2 F(x) \preceq L_F I_d, \quad \forall x \in \text{dom} \psi, \quad (4)$$

where $I_d$ is the $d \times d$ identity matrix. The condition number of $F$ is defined as $\kappa_F = L_F/\sigma_F$.

We focus on a basic setting of distributed optimization where the $m$ machines (workers) compute the gradients in parallel and a centralized server updates the variable $x$. Specifically, during each iteration $t = 0, 1, 2, \ldots$,

(i) the server broadcasts $x_t$ to all $m$ machines;
(ii) each machine $j$ computes the gradient $\nabla f_j(x_t)$ and sends it back to the server;
(iii) the server forms $\nabla F(x_t) = \frac{1}{m} \sum_{j=1}^{m} \nabla f_j(x_t)$ and uses it to compute the next iterate $x_{t+1}$.

A standard way for solving problem (1) in this setting is to implement the proximal gradient method at the server:

$$x_{t+1} = \arg\min_{x \in \mathbb{R}^d} \left\{\nabla F(x_t)^\top x + \psi(x) + \frac{1}{2\eta_t} \|x - x_t\|^2\right\}, \quad (5)$$

where $\|\cdot\|$ denotes the Euclidean norm and $\eta_t > 0$ is the step size. Setting $\eta_t = 1/L_F$ leads to linear convergence:

$$\Phi(x_t) - \Phi(x_*) \leq (1 - \kappa_F^{-1})^t \frac{L_F}{2\kappa_F} \|x_0 - x_*\|^2, \quad (6)$$
where \( x_\ast = \arg \min \Phi(x) \) (e.g., Beck, 2017, Section 10.6). In other words, in order to reach \( \Phi(x_t) - \Phi(x_\ast) \leq \epsilon \), we need \( O(\kappa_F \log(1/\epsilon)) \) iterations, which is also the number of communication rounds between the workers and the server. If we use accelerated proximal gradient methods (e.g., Nesterov, 2004; Beck and Teboulle, 2009; Nesterov, 2013) at the server, then the iteration/communication complexity can be improved to \( O(\sqrt{\kappa_F} \log(1/\epsilon)) \).

### 1.1. Statistical Preconditioning

In general, for minimizing \( F(x) = (1/m) \sum_{j=1}^{m} f_j(x) \) with first-order methods, the communication complexity of \( O(\sqrt{\kappa_F} \log(1/\epsilon)) \) cannot be improved (Arjevani and Shamir, 2015; Scaman et al., 2017). However, for distributed empirical risk minimization (ERM), the additional finite-sum structure of each \( f_j \) in (3) allows further improvement. A key insight here is that the datasets \( D_j \) at different workers are i.i.d. samples from the same source distribution, then the local empirical losses \( f_j \) are statistically very similar to each other and to their average \( F \), especially when \( n \) is large. Statistical preconditioning is a technique to further reduce communication complexity based on this insight.

An essential tool for preconditioning in first-order methods is the Bregman divergence. The Bregman divergence of a strictly convex and differentiable function \( \phi \) is defined as

\[
D_\phi(x,y) \triangleq \phi(x) - \phi(y) - \nabla \phi(y)^\top (x-y).
\]

We also need the following concepts of relative smoothness and strong convexity (Bauschke et al., 2017; Lu et al., 2018).

**Definition 1.** Suppose \( \phi : \mathbb{R}^d \to \mathbb{R} \) is convex and twice differentiable. The function \( F \) is said to be \( L_{F/\phi} \)-smooth and \( \sigma_{F/\phi} \)-strongly convex with respect to \( \phi \) if for all \( x \in \mathbb{R}^d \),

\[
\sigma_{F/\phi} \nabla^2 \phi(x) \preceq \nabla^2 F(x) \preceq L_{F/\phi} \nabla^2 \phi(x). \tag{8}
\]

The classical definition in (4) can be viewed as relative smoothness and strong convexity where \( \phi(x) = (1/2)\|x\|^2 \). Moreover, it can be shown that (8) holds if and only if for all \( x, y \in \mathbb{R}^d \)

\[
\sigma_{F/\phi} D_\phi(x,y) \leq D_F(x,y) \leq L_{F/\phi} D_\phi(x,y). \tag{9}
\]

Consequently, we define the relative condition number of \( F \) with respect to \( \phi \) as \( \kappa_{F/\phi} = L_{F/\phi}/\sigma_{F/\phi} \).

Following the Distributed Approximate Newton (DANE) method by Shamir et al. (2014), we construct the reference function \( \phi \) by adding some extra regularization to one of the local loss functions (say \( f_1 \), without loss of generality):

\[
\phi(x) = f_1(x) + \frac{\mu}{2} \|x\|^2. \tag{10}
\]

Then we replace \((1/2)\|x-x_t\|^2\) in the proximal gradient method (5) with the Bregman divergence of \( \phi \), i.e.,

\[
x_{t+1} = \arg \min_{x \in \mathbb{R}^d} \left\{ \nabla F(x_t)^\top x + \psi(x) + \frac{1}{\eta_t} D_\phi(x,x_t) \right\}. \tag{11}
\]

In this case, worker 1 acts as the server to compute \( x_{t+1} \), which requires solving a nontrivial optimization problem involving the local loss function \( f_1 \).

According to Shamir et al. (2014) and Lu et al. (2018), with \( \eta_t = 1/L_{F/\phi} \), the sequence \( \{x_t\} \) generated by (11) satisfies

\[
\Phi(x_t) - \Phi(x_\ast) \leq \left( 1 - \kappa_{F/\phi}^{-1} \right) L_{F/\phi} D_\phi(x_\ast,x_0), \tag{12}
\]

which is a direct extension of (6). Therefore, the effectiveness of preconditioning hinges on how much smaller \( \kappa_{F/\phi} \) is compared to \( \kappa_F \). Roughly speaking, the better \( f_1 \) or \( \phi \) approximates \( F \), the smaller \( \kappa_{F/\phi} \geq 1 \) is. In the extreme case of \( f_1 \equiv F \) (with only one machine \( m = 1 \)), we can choose \( \mu = 0 \) and thus \( \phi \equiv F \), which leads to \( \kappa_{F/\phi} = 1 \), and we obtain the solution within one step.

In general, we choose \( \mu \) to be an upper bound on the spectral norm of the matrix difference \( \nabla^2 f_1 - \nabla^2 F \). Specifically, we assume that with high probability, for the operator norm between matrices (i.e., the largest singular value),

\[
\|\nabla^2 f_1(x) - \nabla^2 F(x)\| \leq \mu, \quad \forall x \in \text{dom } \psi, \tag{13}
\]

which implies (Zhang and Xiao, 2018, Lemma 3),

\[
\frac{\sigma_F}{\sigma_F + 2\mu} \nabla^2 \phi(x) \preceq \nabla^2 F(x) \preceq \nabla^2 \phi(x). \tag{14}
\]

Now we invoke a statistical argument based on the empirical average structure in (3). Without loss of generality, we assume that \( D_1 \) contains the first \( n \) samples of \( \{z_1, \ldots, z_N\} \) and thus \( \nabla^2 f_1(x) = \frac{1}{n} \sum_{i=1}^n \nabla^2 \ell(x,z_i) \). For any fixed \( x \), we can use Hoeffding’s inequality for matrices (Tropp, 2015) to obtain, with probability \( 1 - \delta \),

\[
\left\| \frac{1}{n} \sum_{i=1}^n \nabla^2 \ell(x,z_i) - \nabla^2 F(x) \right\| \leq \sqrt{\frac{32L^2_d \log(d/\delta)}{n}}, \tag{15}
\]

where \( L_d \) is the uniform upper bound on \( \|\nabla^2 \ell(x,z_i)\| \).

If the losses \( \ell(x,z_i) \) are quadratic in \( x \), then the Hessians are constant and (13) holds with \( \mu = \tilde{O}(L_d/\sqrt{n}) \), hiding the factor \( \log(d/\delta) \). In this case, we derive from (14) that

\[
\kappa_{F/\phi} = 1 + \frac{2\mu}{\sigma_F} = 1 + \tilde{O} \left( \frac{\kappa_F}{\sqrt{n}} \right), \tag{16}
\]

where we assume \( \sigma_F \approx \sigma_t \), where \( \nabla^2 \ell(x,z_i) \geq \sigma_t I_d \) for all \( x \). Therefore, for large \( n \), whenever we have \( \kappa_{F/\phi} < \kappa_F \), the communication complexity \( O(\kappa_{F/\phi} \log(1/\epsilon)) \) is better than without preconditioning.
For non-quadratic loss functions, we need to ensure that (13) holds uniformly over a compact domain with high probability. Standard ball-packing arguments encounter an additional factor of $\sqrt{d}$ (e.g., Zhang and Xiao, 2018, Lemma 6). In this case, we have $\mu = \tilde{O}(L_d \sqrt{d}/n)$ and

$$\kappa_{F/\phi} = 1 + \frac{2\mu}{\sigma_F} = 1 + \tilde{O}\left(\frac{\kappa_L \sqrt{d}}{\sqrt{\mu}}\right), \quad (17)$$

which suggests that the benefit of preconditioning may degrade or disappear in high dimension.

1.2. Contributions and Outline

In this paper, we make the following two contributions.

First, we propose a Statistically Preconditioned Accelerated Gradient (SPAG) method that can further reduce the communication complexity. Accelerated methods with $O(\sqrt{\kappa_{F/\phi}} \log(1/\epsilon))$ complexity have been developed for quadratic loss functions (see related work in Section 2). However, Dragomir et al. (2019) have shown that acceleration is not possible in general in the relatively smooth and strongly convex setting, and that more assumptions are needed. Here, by leveraging the fact the reference function $\phi$ itself is smooth and strongly convex, we obtain

$$\Phi(x_t) - \Phi(x_*) \leq \prod_{t=1}^{\tau} \left(1 - \frac{1}{\sqrt{\kappa_{F/\phi}} G_t}\right) L_{F/\phi} D_\phi(x_*, x_0),$$

where $1 \leq G_t \leq \kappa_\phi$ and $G_t \to 1$ geometrically. Moreover, $G_t$ can be calculated at each iteration and serve as a numerical certificate of the actual convergence rate. In all of our experiments, we observe $G_t \approx 1$ even in early iterations, which results in $O(\sqrt{\kappa_{F/\phi}} \log(1/\epsilon))$ iterations empirically.

Second, we derive refined bounds on the relative condition number for linear prediction models. Linear models such as logistic regression have the form $f(x) = f_j(x) + \mu/2||x||^2$ for each worker $j$, compute $m$ separate local updates using (11), and then use their average as $x_{t+1}$. For quadratic losses, they obtain the communication complexity $\tilde{O}(\kappa_L^2/d) \log(1/\epsilon)$, which is roughly $O(\kappa_{F/\phi}^2 \log(1/\epsilon))$ in our notation, which is much worse than their result without averaging of $O(\kappa_{F/\phi} \log(1/\epsilon))$ given in Section 1.1. We further improve this to $O(\sqrt{\kappa_{F/\phi}} \log(1/\epsilon))$ using acceleration.

Zhang and Xiao (2015) proposed DiSCO, an inexact damped Newton method, where the Newton steps are computed by a distributed conjugate gradient method with a similar preconditioner as (10). They obtain a communication complexity of $O((\sqrt{\kappa_L} d/n^{1/4}) \log(1/\epsilon))$ for quadratic losses and $\tilde{O}(\sqrt{\kappa_L} (d/n)^{1/4} \log(1/\epsilon))$ for self-concordant losses. Comparing with (16) and (17), in both cases they correspond to $O(\sqrt{\kappa_{F/\phi}} \log(1/\epsilon))$ in our notation. Reddi et al. (2016) use the Catalyst framework (Lin et al., 2015) to accelerate DANE; their method, called AIDE, achieves the same improved complexity for quadratic functions. We obtain similar results for smooth convex functions using direct acceleration.

Yuan and Li (2019) revisited the analysis of DANE and found that the worse complexity of $\tilde{O}(\kappa_{F/\phi}^2 \log(1/\epsilon))$ is due to the lost statistical efficiency when averaging $m$ different updates computed by (11). They propose to use a single local preconditioner at the server and obtain a communication complexity of $O((1 + \kappa_{F/\phi}) \log(1/\epsilon))$ for quadratic functions. In addition, they propose a variant of DANE with heavy-ball momentum (DANE-HB), and show that it has communication complexity $O((\sqrt{\kappa_L} d/n^{1/4}) \log(1/\epsilon))$ for quadratic loss functions, matching that of DiSCO and AIDE. For non-quadratic functions, they show DANE-HB has accelerated local convergence rate near the solution.

Wang et al. (2018) proposed GIANT, an approximate New-
ton method that approximates the overall Hessian by the harmonic mean of the local Hessians. It is equivalent to DANE in the quadratic case. They obtain a communication complexity that has logarithmic dependence on the condition number but requires local sample size $n > d$. Mahajan et al. (2018) proposed a distributed algorithm based on local function approximation, which is related to the preconditioning idea of DANE. Wang and Zhang (2019) apply statistical preconditioning to speed up a mini-batch variant of SVRG (Johnson and Zhang, 2013), but they rely on generic Catalyst acceleration and their convergence results only hold for a very small ball around the optimum.

Distributed optimization methods that use dual variables to coordinate solutions to local subproblems include ADMM (Boyd et al., 2010) and CoCoA (Jaggi et al., 2014; Ma et al., 2015; 2017). Numerical experiments demonstrate that they benefit from statistical similarities of local functions in the early iterations (Xiao et al., 2019), but their established communication complexity is no better than $O(\kappa_F \log(1/\epsilon))$.

3. The SPAG Algorithm

Although our main motivation in this paper is distributed optimization, the SPAG algorithm works in the general setting of minimizing relatively smooth and strongly convex functions. In this section, we first present SPAG in the more general setting (Algorithm 1), then explain how to run it for distributed empirical risk minimization.

In the general setting, we consider convex optimization problems of the form (1), where $\psi$ is a closed convex function and $F$ satisfies the following assumption.

**Assumption 1.** $F$ is $L_F$-smooth and $\sigma_F$-strongly convex. In addition, it is $L_{F/\phi}$-smooth and $\sigma_{F/\phi}$-strongly convex with respect to a differentiable function $\phi$, and $\phi$ itself is $L_\phi$-smooth and $\sigma_\phi$-strongly convex.

Algorithm 1 requires an initial point $x_0 \in \text{dom } \psi$ and two parameters $L_{F/\phi}$ and $\sigma_{F/\phi}$. During each iteration, Line 6 finds $\alpha_{t+1} > 0$ by solving a quadratic equation, then Line 7 calculates three scalars $\alpha_t$, $\beta_t$, and $\eta_t$, which are used in the later updates for the three vectors $y_t$, $v_{t+1}$ and $x_{t+1}$. The function $V_t(\cdot)$ being minimized in Line 10 is defined as

$$V_t(x) = \eta_t (\nabla F(y_t)^T x + \psi(x)) + (1 - \beta_t)D_\phi(x, v_t) + \beta_t D_\phi(x, y_t).$$

(18)

The inequality that needs to be satisfied in Line 12 is

$$D_\phi(x_{t+1}, y_t) \leq \alpha_t^2 G_t \left((1 - \beta_t)D_\phi(v_{t+1}, v_t) + \beta_t D_\phi(v_{t+1}, y_t)\right),$$

where $G_t$ is a scaling parameter depending on the properties of $D_\phi$. It is a more flexible version of the triangle scaling gain introduced by Hanzely et al. (2018).

As we will see in Theorem 1, smaller $G_t$'s correspond to faster convergence rate. Algorithm 1 implements a gain-search procedure to automatically find a small $G_t$. At the beginning of each iteration, the algorithm always tries to set $G_t = G_{t-1}/2$ as long as $G_{t-1} \geq 2 (G_{t-1} \text{ is divided by 4 in Line 3 since it is always multiplied by 2 in Line 5).}$ Whenever (19) is not satisfied, $G_t$ is multiplied by 2. When the inequality (19) is satisfied, $G_t$ is within a factor of 2 from its smallest possible value. The following lemma guarantees that the gain-search loop always terminates within a small number of steps (see proof in Appendix A).

**Lemma 1.** If Assumption 1 holds, then the inequality (19) holds with $G_t = \kappa_\phi = L_\phi/\sigma_\phi$.

Therefore, if $\phi = (1/2)\| \cdot \|^2$, then we can set $G_t = 1$ and there is no need to check (19). In general, Algorithm 1 always produces $G_t < 2\kappa_\phi$ for all $t \geq 0$. Following the argument from Nesterov (2013, Lemma 4), the total number of gain-searches performed up to iteration $t$ is bounded by

$$2(t + 1) + \log_2(G_t),$$

which also bounds the total number of gradient evaluations. Thus the overhead is roughly twice as if there were no gain-search. Next we present a convergence theorem for SPAG.

**Theorem 1.** Suppose Assumption 1 holds. Then the sequences generated by SPAG satisfy for all $t \geq 0$,

$$(\Phi(x_t) - \Phi(x_*)) + \sigma_{F/\phi} D_\phi(x_*, v_t) \leq \frac{1}{A_t} D_\phi(x_*, v_0),$$

where $A_t = \frac{1}{\kappa_{F/\phi} G_t} \left(\prod_{\tau=0}^{t-1} (1 + \gamma_\tau) - \prod_{\tau=0}^{t-1} (1 - \gamma_\tau)\right)^2$, and $\gamma_t = \frac{1}{2\sqrt{\kappa_{F/\phi} G_t}}$.

---

**Algorithm 1** SPAG($L_{F/\phi}, \sigma_{F/\phi}, x_0$)

1: $v_0 = x_0$, $A_0 = 0$, $B_0 = 1$, $G_1 = 1$
2: for $t = 0, 1, 2, \ldots$ do
3: $G_t = \max\{1, G_{t-1}/2\}$
4: repeat
5: $G_t \leftarrow 2G_t$
6: Find $\alpha_{t+1}$ such that $a_{t+1}^2 L_{F/\phi} G_t = A_{t+1} B_{t+1}$ where $A_{t+1} = A_t + \alpha_{t+1}, B_{t+1} = B_t + \alpha_{t+1} \sigma_{F/\phi}$
7: $\alpha_t = \frac{a_{t+1}}{A_t + 1}$, $\beta_t = \frac{a_{t+1}^2}{B_t + 1} \sigma_{F/\phi}$, $\eta_t = \frac{1}{A_t + 1}$
8: $y_t = \frac{1}{1 - \sigma_t \beta_t} ((1 - \alpha_t)x_t + \alpha_t (1 - \beta_t)v_t)$
9: Compute $\nabla F(y_t)$ (requires communication)
10: $v_{t+1} = \arg \min_x V_t(x)$
11: $x_{t+1} = (1 - \alpha_t)x_t + \alpha_t v_{t+1}$
12: until Inequality (19) is satisfied
13: end for
The proof of Theorem 1 relies on the techniques of Nesterov and Stich (2017), and the details are given in Appendix A. We can estimate the convergence rate as follows:

\[
\frac{1}{A_t} = O\left(\prod_{r=0}^{t-1} \left(1 - \frac{1}{\sqrt{\kappa_{F/\phi} G_t^r}}\right)\right) = O\left(\left(1 - \frac{1}{\sqrt{\kappa_{F/\phi} G_t}}\right)^t\right),
\]

where \( G_t \) is such that \( \widetilde{G}_t^{-1/2} = (1/t) \sum_{r=0}^{t-1} G_t^{-1/2} \), that is, \( \widetilde{G}_t^{-1/2} \) is the harmonic mean of \( G_t^0, \ldots, G_t^{t-1} \). In addition, it can be shown that \( A_t \geq t^2/(4L_{F/\phi}G_t) \). Therefore, if \( \sigma_{F/\phi} \to 0 \), Theorem 1 gives an accelerated sublinear rate:

\[
\Phi(x_t) - \Phi(x^*) \leq \frac{4L_{F/\phi}G_t}{t^2} D_{\phi}(x^*, x_0).
\]

To estimate the worst case when \( \sigma_{F/\phi} > 0 \), we replace \( G_t \) by \( \kappa_\phi \) to obtain the iteration complexity \( O(\sqrt{\kappa_{F/\phi}}\sigma_\phi \log(1/\epsilon)) \). Since \( \kappa_{F/\phi} \kappa_\phi \approx \kappa_F \), this is roughly \( O(\sqrt{\kappa_F} \log(1/\epsilon)) \), the same as without preconditioning. However, the next lemma shows that under a mild condition, we always have \( G_t \to 1 \) geometrically.

**Lemma 2.** Suppose Assumption 1 holds and in addition, \( \nabla^2 \phi \) is \( M \)-Lipschitz-continuous, i.e., for all \( x, y \in \text{dom } \psi \),

\[
\|\nabla^2 \phi(x) - \nabla^2 \phi(y)\| \leq M \|x - y\|.
\]

Then the inequality (19) holds with

\[
G_t = \min\{\kappa_\phi, 1 + (M/\sigma_\phi) d_t\},
\]

where \( d_t = \|v_{t+1} - v_t\| + \|v_{t+1} - y_t\| + \|x_{t+1} - y_t\| \).

In particular, if \( \phi \) is quadratic, then we have \( M = 0 \) and \( G_t = 1 \) always satisfies (19). In this case, the convergence rate in Theorem 1 satisfies \( 1/A_t = O\left(1 - 1/\sqrt{\kappa_{F/\phi}}\right)^t \).

In general, \( M \neq 0 \), but it can be shown that the sequences generated by Algorithm 1, \( \{x_i\}, \{y_i\} \) and \( \{v_i\} \) all converge to \( x^* \) at the rate \( (1 - 1/\sqrt{\kappa_{F/\phi}})^t \) (see, e.g., Lin and Xiao, 2015, Theorem 1). As a result, \( d_t \to 0 \) and thus \( G_t \to 1 \) at the same rate. Consequently, the convergence rate established in Theorem 1 quickly approaches \( O\left(1 - 1/\sqrt{\kappa_{F/\phi}}\right)^t \).

The asymptotic nature of the preconditioned convergence rate in the nonquadratic case (\( G_t \) converges to 1 instead of being a small constant) seems to be unavoidable, given the recent work on lower bounds for mirror descent methods by Dragomir et al. (2019).

### 3.1. Implementation for Distributed Optimization

In distributed optimization, Algorithm 1 is implemented at the server. During each iteration, communication between the server and the workers only happens when computing \( \nabla F(y_i) \). Checking if the inequality (19) holds locally requires that the server has access to the preconditioner \( \phi \).

If the datasets on different workers are i.i.d. samples from the same source distribution, then we can use any \( f_j \) in the definition of \( \phi \) in (10) and assign worker \( j \) as the server. However, this is often not the case in practice and obtaining i.i.d. datasets on different workers may involve expensive shuffling and exchanging large amount of data among the workers. In this case, a better alternative is to randomly sample small portions of the data on each worker and send them to a dedicated server. We call this sub-sampled dataset \( D_0 \) and the local loss at the server \( f_0 \), which is defined the same way as in (3). Then the server implements Algorithm 1 with \( \phi(x) = f_0(x) + (\mu/2)\|x\|^2 \). Here we only need \( D_0 \) to be a uniform sub-sample of \( \cup_{j=1}^n D_j \), which is critical for effective preconditioning. On the other hand, it is not a problem at all if the datasets at the workers, \( D_1, \ldots, D_m \), are not shuffled to be i.i.d., because it does not change the average gradients \( \nabla F(y_i) \). In the rest of the paper, we omit the subscript to simply use \( f \) to represent the local empirical loss function. As discussed in Section 1.1, if

\[
\|\nabla^2 f(x) - \nabla^2 F(x)\| \leq \mu, \quad \forall x \in \text{dom } \psi
\]

with high probability, then according to (14), we can choose

\[
L_{F/\phi} = 1, \quad \sigma_{F/\phi} = \frac{\sigma_F}{\sigma_F + 2\mu}
\]

as the input to Algorithm 1. In the next section, we leverage matrix concentration bounds to estimate how \( \mu \) varies with the number of subsamples \( n \). With sufficiently large \( n \), we can make \( \mu \) small so that the relative condition number \( \kappa_{F/\phi} = 1 + 2\mu/\sigma_F \) is much smaller than \( \kappa_F \).

### 4. Bounding the Relative Condition Number

In this section, we derive refined matrix concentration bounds for linear prediction models. Suppose the overall dataset consists of \( N \) samples \( \{z_1, \ldots, z_N\} \), where each \( z_i = (a_i, b_i) \) with \( a_i \in \mathbb{R}^d \) being a feature vector and \( b_i \) the corresponding label or regression target. Linear models (including logistic and ridge regression) have the form \( \ell(x, z_i) = \ell_i(a_i^T x) + \frac{1}{2}\|x\|^2 \), where \( \ell_i \) is twice differentiable and may depend on \( b_i \) and \( \lambda > 0 \). We further assume that \( \ell''_i = \ell''_j \) for all \( i \) and \( j \), which is valid for logistic and ridge regression as well. Since \( f(x) = (1/n) \sum_{i=1}^n \ell_i(x, z_i) \), we have

\[
\nabla^2 f(x) = \frac{1}{n} \sum_{i=1}^n \ell''_i(a_i^T x) a_i a_i^T + \lambda I_d.
\]

Here we omit the subscript \( j \) in \( f_j \) since we only need one subsampled dataset at the server, as explained in Section 3.1. For the overall loss function defined in (2), the Hessian \( \nabla^2 F(x) \) is defined similarly by replacing \( n \) with \( N \).

We assume for simplicity that the strong convexity of \( F \) mainly comes from regularization, that is, \( \sigma_F = \sigma_\lambda = \lambda \), but
the results can be easily extended to account for the strong
convexity from data. We start by showing tight results for
quadratics, and then provide uniform concentration bounds
of Hessians for more general loss functions. Finally, we
give a refined bound when the $a_i$’s are sub-Gaussian.

4.1. Quadratic Case

We assume in this section that $\ell_i(a_i^\top x) = (a_i^\top x - b_i)^2/2$, and
that there exists a constant $R$ such that $\|a_i\| \leq R$ for all $i = 1, \ldots, N$. In this case we have $L_f = R^2$ and
$\kappa_f = R^2/\lambda$. Since the Hessians do not depend on $x$, we use the notation

\[ H_F = \nabla^2 F(x), \quad H_f = \nabla^2 f(x). \]

Previous works (Shamir et al., 2014; Reddi et al., 2016;
Yuan and Li, 2019) use the Hoeffding bound (15) to obtain

\[ (1 + \frac{2\mu}{\lambda})^{-1} (H_f + \mu I_d) \preceq H_F \preceq H_f + \mu I_d, \tag{23} \]

\[ \text{with} \quad \mu = \frac{R^2}{\sqrt{n}} \sqrt{32 \log(d/\delta)}. \tag{24} \]

Our result is given in the following theorem.

**Theorem 2.** Suppose $\ell_i$ is quadratic and $\|a_i\| \leq R$ for all $i$. For a fixed $\delta > 0$, if $n > \frac{28}{3} \log \left( \frac{2d}{\delta} \right)$, then the following inequality holds with probability at least $1 - \delta$:

\[ \left( \frac{3}{2} + \frac{2\mu}{\lambda} \right)^{-1} (H_f + \mu I_d) \preceq H_F \preceq 2 (H_f + \mu I_d), \tag{25} \]

\[ \text{with} \quad \mu = \frac{1}{2} \left( \frac{28R^2}{3n} \log \left( \frac{2d}{\delta} \right) - \lambda \right)^+ \tag{26} \]

Thus, for this choice of $\mu$, $\sigma_{F/\phi} = \left( \frac{3}{2} + \frac{2\mu}{\lambda} \right)^{-1}$, $L_{F/\phi} = 2$ and so $\kappa_{F/\phi} = O \left( 1 + \frac{\mu}{\lambda} \log \left( \frac{d}{\delta} \right) \right)$ with probability $1 - \delta$.

Theorem 2 improves on the result in (24) by a factor of $\sqrt{n}$. The reason is that matrix inequality (23) is derived from the additive bound $\|H_f - H_F\| \leq \mu$ (e.g., Shamir et al., 2014; Yuan and Li, 2019). We derive the matrix inequality (25) directly from a multiplicative bound using the matrix Bernstein inequality (see proof in Appendix B.1). Note that by using matrix Bernstein instead of matrix Hoeffding inequality (Tropp, 2015), one can refine the bound for $\mu$ in (23) from $L_f/\sqrt{n}$ to $\sqrt{L_f L_F/n}$, which can be as small as $L_f/n$ in the extreme case when all the $a_i$’s are orthogonal. Our bound in (26) states that $\mu = O(L_f/n)$ in general for quadratic problems, leading to $\kappa_{F/\phi} = O(1 + \kappa_f/n)$.

**Remark 1.** Theorem 2 is proved by assuming random sampling with replacement. In practice, we mostly use random sampling without replacement, which usually concentrates even more than with replacement (Hoeffding, 1963).

**Remark 2.** In terms of reducing $\kappa_{F/\phi}$, there is not much benefit to having $\mu < \lambda$. Indeed, higher values of $\mu$ regularize the inner problem of minimizing $V_i(x)$ in (18), because the condition number of $D_\phi(x, y) = D_f(x, y) + (\mu/2)||x - y||^2$ is $(L_f + \mu)/(\lambda + \mu)$. Increasing $\mu$ can thus lead to substantially easier subproblems when $\mu > \lambda$, which reduces the computation cost at the server, although this may sometimes affect the rate of convergence.

4.2. Non-quadratic Case

For non-quadratic loss functions, we need $\nabla^2 f(x)$ to be a good approximation of $\nabla^2 F(x)$ for all iterations of the SPAG algorithm. It is tempting to argue that concentration only needs to hold for the iterates of SPAG, and a union bound would then give an extra $\log T$ factors for $T$ iterations. Yet this only works for one step since $x_t$ depends on the points chosen to build $f$ for $t > 0$, so the $\ell_t''(a_i^\top x_t) a_i a_i^\top$ are not independent for different $i$ (because of $x_t$). Therefore, the concentration bounds need to be written at points that do not depend on $f$. In order to achieve this, we restrict the optimization variable within a bounded convex set and prove uniform concentration of Hessians over the set. Without loss of generality, we consider optimization problems constrained in $B(0, D)$, the ball of radius $D$ centered at 0. Correspondingly, we set the nonsmooth regularization function as $\psi(x) = 0$ if $x \in B(0, D)$ and infinity otherwise.

If the radius $D$ is small, it is then possible to leverage the quadratic bound by using the inequality

\[ \|H_f(x) - H_F(x)\| \leq \|H_f(x) - H_f(y)\| + \|H_f(y) - H_F(y)\| + \|H_F(y) - H_F(x)\|. \]

Thus, under a Lipschitz-continuous Hessian assumption (which we have), only concentration at point $y$ matters. Yet, such bounding is only meaningful when $x$ is close to $y$, thus leading to the very small convergence radius of Wang and Zhang (2019, Theorem 13), in which they use concentration at the optimal point $x_*$. Using this argument for several $y$’s that pave $B(0, D)$ leads to an extra $\sqrt{d}$ multiplicative factor since concentration needs to hold at exponentially (in $d$) many points, as discussed in Section 1.1. We take a different approach in this work, and proceed by directly bounding the supremum for all $x \in B(0, D)$, thus looking for the smallest $\mu$ that satisfies:

\[ \sup_{x \in B(0, D)} \|H_f(x) - H_F(x)\|_{op} \leq \mu. \tag{27} \]

Equation (23) can then be used with this specific $\mu$. We now introduce Assumption 2, which is for example verified for logistic regression with $B_t = 1/4$ and $M_t = 1$.

**Assumption 2.** There exist $B_t$ and $M_t$ such that $\ell_t''(a^\top x) \leq B_t$ almost surely for all $x \in B(0, D)$.
Theorem 3. If \( \ell_i \) satisfies Assumption 2, then Equation (27) is satisfied with probability at least \( 1 - \delta \) for
\[
\mu = \sqrt{4\pi R^2 \frac{R}{\sqrt{n}}} \left( B_\ell \left[ 2 + \sqrt{\frac{1}{2\pi} \log(\delta^{-1})} \right] + RM_{i}D \right).
\]

**Sketch of proof.** The high probability bound on the supremum is obtained using Mc Diarmid inequality (Boucheron et al., 2013). This requires a bound on its expectation, which is obtained using symmetrization and the Sudakov-Fernique Lemma (Boucheron et al., 2013). The complete proof can be found in Appendix B.2.

The bound of Theorem 3 is relatively tight as long as \( RM_{i}D < B_\ell \sqrt{\log(\delta^{-1})} \). Indeed, using the matrix Bernstein inequality for a fixed \( x \in B(0, D) \) would yield
\[
\mu = O \left( R \sqrt{\log(\delta^{-1})} / \sqrt{n} \right).
\]
Therefore, Theorem 3 is tight up to a factor \( R / \sqrt{\log(\delta^{-1})} \) in this case.

**Remark 3.** We consider \( D \) to be fixed in this work, although obtaining a meaningful solution to the ERM problem may require \( D \) to depend on the dimension \( d \). Yet, \( D \) would actually depend on the intrinsic dimension of the data, which can be much smaller than \( d \), especially when features are sparse.

### 4.3. Sub-Gaussian Bound

We show in this section that the bound of Theorem 3 can be improved under a stronger sub-Gaussian assumption on \( a \).

**Definition 2.** The random variable \( a \in \mathbb{R}^d \) is sub-Gaussian with parameter \( \rho > 0 \) if one has for all \( \epsilon > 0 \), \( x \in B(0, D) \):
\[
\mathbb{P}(\{a^T \geq \epsilon \}) \leq 2e^{-\frac{\epsilon^2}{2\rho^2}}.
\]

**Theorem 4.** If \( \ell_i \) satisfies Assumption 2 and the \( a_i \) are sub-Gaussian with constant \( \rho \), then denoting \( \tilde{B} = B_\ell / (M_{i}D) \), there exists \( C > 0 \) such that Equation (27) is satisfied with probability \( 1 - \delta \) for
\[
\mu = C \frac{R^2 M_i D}{\sqrt{n}} \left( d + \log(\delta^{-1}) \right) \left[ \frac{\rho + \tilde{B}}{\sqrt{d}} + \frac{\rho + (R^2 \tilde{B})^{1/2}}{\sqrt{n}} \right].
\]

Recall that this value of \( \mu \) can be plugged into Equation (23) to bound the relative condition number.

**Sketch of proof.** This bound is a specific instantiation of a more general result based on chaining, which is a standard argument for proving results on suprema of empirical processes (Boucheron et al., 2013). The complete proof can be found in Appendix B.3.

The sub-Gaussian assumption (28) always holds with \( \rho = R \), the almost sure bound on \( \|a_i\| \). However, Theorem 4 improves over Theorem 3 only with a stronger sub-Gaussian assumption, i.e., when \( \rho < R \). In particular for \( a_i \) uniform over \( B(0, R) \), one has \( \rho = R / \sqrt{d} \). Assuming further that the \( R^2 \tilde{B}^{1/2} / \sqrt{n} \) term dominates yields
\[
\mu = O \left( R^2 \tilde{B} / \sqrt{n} \right), \quad \sqrt{n} \text{ improvement over Theorem 3.}
\]
We expect tighter versions of Theorem 4, involving the effective dimension \( d_{eff} \) of vectors \( a_i \) instead of the full dimension \( d \), to hold.

### 5. Experiments

We have seen in the previous section that preconditioned gradient methods can outperform gradient descent by a large margin in terms of communication rounds, which was already observed empirically (Shamir et al., 2014; Reddi et al., 2016; Yuan and Li, 2019). We compare in this section the performances of SPAG with those of DANE and its heavy-ball acceleration, HB-DANE (Yuan and Li, 2019), as well as accelerated gradient descent (AGD). Due to its better convergence guarantees (Shamir et al., 2014; Yuan and Li, 2019), DANE refers in this section to the proximal gradient method with the Bregman divergence associated to \( f_1 = (\mu/2) \| \cdot \|^2 \) (without averaging over \( m \) workers).

We apply these algorithms to train linear prediction models over a dataset \( \{(a_i, b_i)\}_{i=1}^N \), where each \( a_i \in \mathbb{R}^d \) is a feature vector and \( b_i \) is the corresponding label or regression target. Specifically, we solve the problem
\[
\min_{x \in \mathbb{R}^d} F(x) = \frac{1}{N} \sum_{i=1}^n \ell_i(a_i^T x) + \frac{\lambda}{2} \|x\|^2,
\]
where \( \ell_i(a_i^T x) = \log(1 + \exp(-b_i(a_i^T x))) \) for logistic regression with \( b_i \in \{-1, +1\} \) and \( \ell_i(a_i^T x) = (a_i^T x - b_i)^2 \) for ridge regression with \( b_i \in \mathbb{R} \). We use two datasets from LibSVM\(^1\), RCV1 (Lewis et al., 2004) and the preprocessed version of KDD2010 (algebra) (Yu et al., 2010).

\(^1\)Accessible at https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html
Figure 2. Logistic regression on RCV1 dataset. The legend is the same for all figures. We use $\mu = 0.1/n$, except for $n = 10^5$ for which we use $\mu = 2 \cdot 10^{-6}$.

Note that, as mentioned in Section 3.1, the number of nodes used by SPAG does not affect its iteration complexity (but change the parallelism of computing $\nabla F(x_i)$). Only the size $n$ of the dataset used for preconditioning matters. We initialize all algorithms at the same point, which is the minimizer of the server’s entire local loss (with $10^5$ samples regardless of how many samples are used for preconditioning). Note that this is not mandatory but provides a good communication-free initialization, although it requires partially solving the server’s local loss. Not performing this would simply yield a worse initialization.

**Tuning $\mu$.** Although $\mu$ can be estimated using concentration results, as done in Section 4, these bounds are too loose to be used in practice. Yet, they show that $\mu$ depends very weakly on $\lambda$. This is verified experimentally, and we therefore use the same value for $\mu$ regardless of $\lambda$. To test the impact of $\mu$ on the iteration complexity, we fix a step-size of 1 and plot the convergence speed of SPAG for several values of $\mu$. We see on Figure 1 that the value of $\mu$ drastically affects convergence, actually playing a role similar to the inverse of a step-size. Indeed, the smaller the $\mu$ the faster the convergence, up to a point at which the algorithm is not stable anymore. Convergence could be obtained for smaller values of $\mu$ by taking a smaller step-size. Yet, the step-size needs to be tuned for each value of $\mu$, and we observed that this does not lead to significant improvements in practice. Thus, unless explicitly stated, we stick to the guidelines for DANE by Shamir et al. (2014), i.e., we choose $L_{F/\phi} = 1$ and tune $\mu$. The tuning strategy for $\mu$ is the following: we tune the base value of $\mu$ by starting from $0.1/n$ for the smallest $n$ and then decreasing it as long as it is stable, or increasing it as long as it is unstable. Then, we keep this base value and obtain $\mu$ for other values of $n$ by relying on the fact that $\mu$ should be proportional to $1/n$ (slightly adjusting when necessary if the algorithm becomes unstable). Therefore, even though some tuning is required to set $\mu$, this tuning is guided by the insight provided by Section 4.

**Setting acceleration parameters.** SPAG and HB-DANE require additional parameters compared with DANE. Yet, we use in our experiments the values given by the theory, i.e., we use SPAG with $\sigma^{-1}_{F/\phi} = 1 + 2\mu/\lambda$ and HB-DANE with $\beta = (1 - (1 + 2\mu/\lambda)^{-1/2})^2$. Fine tuning these parameters only leads to small improvements for both algorithms, as described in Appendix C. Therefore, SPAG and HB-DANE do not require more parameter tuning than DANE. We tune both the learning rate and the momentum of AGD.

**Line search for $G_t$.** As explained in Section 3, the optimal $G_t$ is obtained through a line search. Yet, we observed in all our experiments that $G_t = 1$ most of the time. This is due to the fact that we start at the minimizer of the local cost function, which can be close to the global solution. In addition, Equation (20) can actually be verified for $G_t < 1$, even in the quadratic. Therefore, the line search generally has no added cost (apart from checking that $G_t = 1$ works) and the effective rate in our experiments is $\kappa_{F/\phi}^{-1/2}$. Experiments for Figures 2 and 4 use $G_t = 1$ for simplicity.

**Local subproblems.** Local problems are solved using a sparse implementation of dual-free SDCA (Shalev-Shwartz, 2016). In practice, the ill-conditioned regime is very hard, especially when $\mu$ is small. Indeed, the local subproblems are very hard to solve, and it should be beneficial to use accelerated algorithms to solve the inner problems. In our experiments, we warm-start the local problems (initializing
on the solution of the previous one), and keep doing passes over the preconditioning dataset until \(|\nabla V_i(x_i)| \leq 10^{-9}\) (checked at each epoch).

**RCV1.** Figure 2 presents results for logistic regression on the RCV1 dataset with different regularization weights. All algorithms are run with \(N = 677399\) (split over 4 nodes) and \(d = 47236\). We see that in Figure 2(a), the curves can be clustered by values of \(n\), meaning that when regularization is relatively high (\(\lambda = 10^{-5}\)), increasing the preconditioning sample size has a greater effect than acceleration since the problem is already well-conditioned. In particular, acceleration does not improve the convergence rate when \(n = 10^5\) and \(\lambda = 10^{-5}\). When regularization is smaller (\(\lambda = 10^{-7}\)), SPAG and HB-DANE outperform DANE even when ten times less samples are used for preconditioning, as shown in Figure 2(b). As discussed in Appendix C, finer tuning (without using the theoretical parameters) of the momentum marginally improves the performances of SPAG and HB-DANE, at the cost of a grid search. SPAG generally outperforms HB-DANE in our experiments, but both methods have comparable asymptotic rates.

Figure 3 presents results for ridge regression on the same RCV1 dataset, using the class labels as the regression targets. In this case, \(\phi\) is quadratic so we do not need line search on \(G_t\) and always set \(G_t = 1\). The results are very similar to the logistic regression case.

**KDD2010.** Figure 4 presents the results of larger scale experiments on a random subset of the KDD2010 dataset with \(N = 7557074\) (split over 80 nodes), \(d = 20216830\) and \(\lambda = 10^{-7}\). The conclusions are similar to the experiments on RCV1, i.e., acceleration allows to use significantly less samples at the server for a given convergence speed. AGD competes with DANE when \(\lambda\) and \(n\) are small, but it is outperformed by SPAG in all our experiments. More experiments investigating the impact of line search, tuning and inaccurate local solutions are presented in Appendix C.

### 6. Conclusion

We have introduced SPAG, an accelerated algorithm that performs statistical preconditioning for large-scale distributed optimization. Although our motivation in this paper is distributed empirical risk minimization, SPAG applies to much more general settings that can benefit from statistical preconditioning. We have given tight bounds on the relative condition number, a crucial quantity to understand the convergence rate of preconditioned algorithms. We have also shown, both in theory and in experiments, that acceleration allows SPAG to efficiently leverage rough preconditioning when only limited number of local samples are available. Preliminary experiments suggest that SPAG is more robust to inaccurate solution of the inner problems than HB-DANE. Characterizing the effects of inaccurate inner solutions in the preconditioning setting would be an interesting extension of this work.

### Acknowledgements

This work was funded in part by the French government under management of Agence Nationale de la Recherche as part of the “Investissements d’avenir” program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute). We also acknowledge support from the European Research Council (grant SEQUOIA 724063) and from the MSR-INRIA joint centre.
References


