Contents lists available at ScienceDirect

EURO Journal on Computational Optimization

journal homepage: www.elsevier.com/locate/ejco

Hyperfast second-order local solvers for efficient statistically preconditioned distributed optimization

Pavel Dvurechensky [a,*], Dmitry Kamzolov [b,h], Aleksandr Lukashevich [c], Soomin Lee [d], Erik Ordentlich [d], César A. Uribe [e], Alexander Gasnikov [b,f,g]

[a] *Weierstrass Institute for Applied Analysis and Stochastics, Berlin, Germany*
[b] *Moscow Institute of Physics and Technology, Dolgoprudny, Russia*
[c] *Center for Energy Science and Technology, Skolkovo Institute of Science and Technology, Moscow, Russia*
[d] *Yahoo! Research, Sunnyvale, CA, United States of America*
[e] *Rice University, Houston, TX, United States of America*
[f] *Institute for Information Transmission Problems RAS, Moscow, Russia*
[g] *HSE University, Moscow, Russia*
[h] *Mohamed bin Zayed University of Artificial Intelligence, Masdar City, Abu Dhabi, United Arab Emirates*

ARTICLE INFO

Keywords:
Empirical risk minimization
Distributed optimization
Statistical preconditioning
Tensor optimization methods

ABSTRACT

Statistical preconditioning enables fast methods for distributed large-scale empirical risk minimization problems. In this approach, multiple worker nodes compute gradients in parallel, which are then used by the central node to update the parameter by solving an auxiliary (preconditioned) smaller-scale optimization problem. The recently proposed Statistically Preconditioned Accelerated Gradient (SPAG) method [1] has complexity bounds superior to other such algorithms but requires an exact solution for computationally intensive auxiliary optimization problems at every iteration. In this paper, we propose an Inexact SPAG (InSPAG) and explicitly characterize the accuracy by which the corresponding auxiliary subproblem needs to be solved to guarantee the same

* Corresponding author.
E-mail addresses: pavel.dvurechensky@wias-berlin.de (P. Dvurechensky), kamzolov.dmitry@phystech.edu (D. Kamzolov), aleksandr.lukashevich@skoltech.ru (A. Lukashevich), soominl@yahooinc.com (S. Lee), eord@yahooinc.com (E. Ordentlich), cauribe@rice.edu (C.A. Uribe), gasnikov.av@mipt.ru (A. Gasnikov).

https://doi.org/10.1016/j.ejco.2022.100045

convergence rate as the exact method. We build our results by first developing an inexact adaptive accelerated Bregman proximal gradient method for general optimization problems under relative smoothness and strong convexity assumptions, which may be of independent interest. Moreover, we explore the properties of the auxiliary problem in the InSPAG algorithm assuming Lipschitz third-order derivatives and strong convexity. For such problem class, we develop a linearly convergent Hyperfast second-order method and estimate the total complexity of the InSPAG method with hyperfast auxiliary problem solver. Finally, we illustrate the proposed method's practical efficiency by performing large-scale numerical experiments on logistic regression models. To the best of our knowledge, these are the first empirical results on implementing high-order methods on large-scale problems, as we work with data where the dimension is of the order of 3 million, and the number of samples is 700 million.

1. Introduction

The efficient parallelization of large-scale learning is one of the most challenging problems in modern machine learning. Among several approaches, distributed computation and preconditioning have been shown effective in accelerating optimization algorithms, especially with increasing amounts of data [2,1,3]. In this paper, we propose an efficient distributed optimization algorithm for solving the empirical risk minimization (ERM) problem:

$$\min_{x\in\mathbb{R}^d} \left\{ f(x) \triangleq F(x) + h(x) \right\}, \tag{1}$$

where $h(x)$ is a convex regularizer and $F(x)$ is the empirical loss

$$F(x) \triangleq \frac{1}{N}\sum_{i=1}^{N} \ell(x;\zeta_i). \tag{2}$$

Here $\mathcal{D} \triangleq \{\zeta_i = (\xi_i, \eta_i)\}_{i=1}^{N}$ is a set of N training data samples, and ℓ is a convex loss function with respect to x. We assume that F is L_F-smooth and μ_F-strongly convex, i.e.,

$$\mu_F I_d \preceq \nabla^2 F(x) \preceq L_F I_d, \tag{3}$$

where I_d is the d-dimensional identity matrix. The condition number of F is denoted as $\kappa_F = L_F/\mu_F$, and the solution to (1) is denoted as x_*.

Sum-type optimization problems of the form (1) are used to model various statistical learning problems, including least squares regression, logistic regression, and support vector machines. One characteristic of modern uses of models like (1) is the so-called large-scale regime, i.e., when N is very large. Large N poses additional challenges related to the storage and processing of data, which in turn drives the need for modern distributed/federated architectures [4] that take advantage of parallel processing capabilities [5], e.g., Apache Spark [6], Parameter Server [7] and MapReduce [8].

In practice, when N is very large, the complete set of data points $\mathcal{D}$ cannot be stored or is not accessible at a single machine. Instead, data is distributed among m computing units/nodes/agents such that $\mathcal{D} = \{\mathcal{D}_1, \dots, \mathcal{D}_m\}$. Moreover, the distributed setup assumes there is a central node, that is able to communicate with all the worker nodes. Without loss of generality we assume that $N = mn$, i.e., machine $j \in \{1, \dots, m\}$ locally stores n samples $\mathcal{D}_j = \{\xi_i^{(j)}, \eta_i^{(j)}\}_{i=1}^n$. Specifically, each agent j has a local empirical risk, denoted as $F_j(x) \triangleq (1/n) \sum_{i=1}^n \ell(x; \xi_i^{(j)}, \eta_i^{(j)})$. Thus,

$$F(x)=\frac{1}{m}\sum_{j=1}^{m} F_j(x)=\frac{1}{nm}\sum_{j=1}^{m}\sum_{i=1}^{n}\ell(x;\xi_i^{(j)},\eta_i^{(j)}). \tag{4}$$

The centralized distributed optimization architecture described above, with a central node and a number of workers, typically involves two resources: communication and computation. Communication is usually regarded as the most valuable resource [9]. Thus, recent efforts [2,1,3] have been focused on the efficiency of communications, where one seeks to minimize (4) with a minimal number of communication rounds between the workers and the central node.

Recent distributed optimization approaches: The distributed approximate Newton-type method (DANE) [2] has been one of the most popular second-order methods for communication-efficient distributed machine learning. DANE improves the polynomial dependency of the iteration complexity on the condition number κ_F of first-order methods for distributed empirical risk minimization problems, compared to the geometric rates available for centralized, i.e., non-distributed, methods [10]. Particularly, DANE has an iteration (communication) complexity of $\widetilde{O}(\kappa_F^2/n)$[1] for quadratic functions, and $\widetilde{O}(\kappa_F)$ for convex non-quadratic functions. However, DANE requires the exact solution of a carefully constructed subproblem, which can be impractical [2]. An inexact version of DANE, termed InexactDANE [11], and its accelerated variant, termed AIDE [11], achieve an iteration complexity of $\widetilde{O}(\kappa_F)$, and $\widetilde{O}(\sqrt{\kappa_F})$ respectively, without requiring exact solutions of the auxiliary subproblem. For quadratic functions InexactDANE and AIDE have an iteration complexity of $\widetilde{O}(\kappa_F^2/n)$ and $\widetilde{O}(\sqrt{\kappa_F}/n^{1/4})$ respectively. Nevertheless, the advantage of preconditioning, where the condition number is effectively reduced as n increases, was only shown for quadratic problems. Recently,

[1] The $\widetilde{O}$-notation means non-asymptotic inequality up to constant and poly-logarithmic factors. More precisely, $A = \widetilde{O}(B)$ if there exist constants $C, a > 0$ such that $A \leq CB \ln^a \frac{1}{\varepsilon}$.

in [3], the authors showed that the preconditioning effect holds locally for a variation of DANE termed DANE-HB with inexact solutions to the local subproblem. Specifically, an iteration complexity of $\widetilde{O}(d^{1/4}\sqrt{\kappa_F}/n^{1/4})$ was shown to hold in a neighborhood around the optimal point for non-quadratic convex functions. Additionally, for linear prediction models, an improved global bound of $\widetilde{O}(\sqrt{\kappa_F}/n^{1/4})$ was shown [3] to be achieved by the D^{2}ANE Algorithm. In [12] the authors propose the DiSCO algorithm with global bounds $\widetilde{O}(\sqrt{\kappa_F}/n^{1/4})$ for quadratic functions and $\widetilde{O}(d^{1/4}\sqrt{\kappa_F}/n^{1/4})$ for self-concordant functions which are a different class than functions satisfying (3). One of the main observations in [3] is that the looseness in the bounds of DANE and AIDE came from the reduce (model aggregation) step done by the central node. Thus, DANE-HB and D^{2}ANE build their results from a modified structure. The worker nodes compute gradients and communicate them back to the central node, which solves the preconditioned auxiliary subproblem. Such algorithmic structure was used in [1] recently, where the authors proposed the Statistically Preconditioned Accelerated Gradient (SPAG) method. SPAG has an iteration complexity of $\widetilde{O}(\sqrt{\kappa_F}/n^{1/4})$ for quadratic functions with direct acceleration, instead of using the Catalyst framework [13]. SPAG was also shown to have an asymptotic iteration complexity of $\widetilde{O}(\sqrt{\kappa_F}/n^{1/4})$, with empirical evidence that such rate behavior holds non-asymptotically in practice. However, exact solvers for the auxiliary subproblem on the central node are required. Such convergence rates match complexity lower bounds [14,15]. In a more challenging setup (which we do not consider in this paper) of decentralized distributed optimization [16] propose an algorithm with iteration complexity $\widetilde{O}(\kappa_F/\sqrt{n})$ and similar up to a network-dependent factor communication complexity.

Although SPAG obtains the near-optimal iteration complexity for distributed algorithms applied to (1) and (4), it strongly depends on the ability to exactly solve an intermediate auxiliary optimization subproblem (usually in the form of a non-Euclidean Bregman projection), whose complexity was not explicitly taken into account in [1]. More importantly, as pointed out in [1], such an intermediate problem is computationally hard, and the accuracy of its solution dramatically affects the performance of the whole method. *We solve this issue in this paper.*

Our solution's key innovation is explicitly considering the auxiliary subproblem's inexactness and quantifying how it affects the convergence rate of the whole algorithm. Moreover, for the case of functions with high-order bounded derivatives (e.g., logistic regression or softmax problems [17]), we provide a Hyperfast second-order method that efficiently computes the approximate solution of the subproblem. This approach builds upon the line of works on *implementable* tensor methods for *convex* problems recently initiated[2] by Yu. Nesterov [22], where it was shown that the third-order method for convex problems with Lipschitz third-order derivative could have a convex subproblem

[2] We underline that the main words here are *implementable* and *convex*. Adaptive tensor methods with optimal complexity guarantees for non-convex problems were proposed earlier in [18–20], and previously known tensor methods for convex problems [21] did not necessarily have convex auxiliary problem in each iteration.

and its solution can be efficiently implemented. Later, [23] proposed near-optimal tensor methods with complexity bounds which match up to a logarithmic factor the lower bounds for highly-smooth convex optimization. [24] proposes a third-order tensor method with third-order derivative approximated by finite-difference of gradients, which leads to a Superfast second-order method with convergence rate $O(1/k^4)$ for convex functions with Lipschitz third-order derivative. As a next step, [25] proposes an inexact accelerated high-order proximal point method which allows improving, compared to Superfast second-order method, the convergence rate to $O(1/k^5)$ up to logarithmic factors. In parallel to the previous work and inspired by [24], the authors of [26] proposed a Hyperfast second-order method with the same convergence rate, but based on another accelerated high-order method developed in [23]. In this paper, we extend both methods to the setting of strongly convex minimization problems and apply them to solve the intermediate auxiliary optimization subproblem in each iteration of our inexact version of SPAG.

Contributions SPAG is one of the fastest distributed methods (in terms of communication steps number) for the minimization of (1) and (4) with i.i.d. samples [1]. Moreover, the Hyperfast second-order method is the best known (near-optimal) second-order method to minimize convex functions with Lipschitz third-order derivatives. We argue that the extended combination of the proposed inexact SPAG and the new Hyperfast second-order method provides a useful approach to construct new efficient distributed algorithms. Specifically, in SPAG, the central node solves a problem with a similar structure as (1), but with a smaller number n of data samples. Therefore, with a reduced number of samples, the complexity of calculating the Hessian is comparable (due to the sum-type structure of F) with its inversion by the matrix inversion lemma [27] and modern practical versions of Strassen-type algorithm [28]. In this regime, at the central node, Hyperfast second-order methods outperform existing variance-reduced stochastic first-order schemes. *We extend the theoretical analysis of inexact statistical preconditioning methods alongside high-order methods and show that they jointly provide an efficient second-order method that outperforms (from theoretical and practical points of view) well-known (stochastic) first-order schemes.*

The main contributions of this paper are as follows:

- Since SPAG is based on the accelerated Bregman proximal gradient method for relatively smooth and strongly-convex problems, we first propose an inexact accelerated Bregman proximal gradient method for general convex optimization problems. Our algorithm is based on an inexact model for the objective, which subsumes the setting of relatively smooth and (strongly-)convex problems and the setting of inexact first-order oracles. Our algorithm also allows for approximate Bregman projections. We estimate the convergence rate and rates of inexactnesses accumulation.
- We propose an Inexact Statistically Preconditioned Accelerated Gradient (InSPAG) method for distributed optimization problem (1), (4), and explicitly characterize the accuracy by which the corresponding auxiliary subproblem needs to be solved to

guarantee the same convergence rate as the exact method, i.e., $\widetilde{O}(\sqrt{\kappa_F}/n^{1/4})$. Our method is not a direct extension and has a slightly simpler structure than the method in [1].

- We extend and generalize the Hyperfast second-order method [25,26], recently proposed for smooth and convex problems, to the class of uniformly, and especially strongly, convex functions. We show a linear convergence rate for this problem class.
- Based on an example of sparse logistic regression, we discuss the distributed optimization problem regime, for which Hyperfast second-order optimization methods provide a theoretical advantage over classical first-order methods for the problem size, dimension, and desired accuracy of the solution.
- We provide experimental results in application to large-scale machine learning problems that show the efficiency of the use of high-order methods in practice. To the authors' best knowledge, this is one of the first attempts to apply near-optimal tensor methods for real data and applications. Specifically, we test the proposed algorithm on a proprietary data set with 710 million entries and a dimension of 3.2 million.

Outline In Section 2, we introduce the inexact accelerated Bregman proximal gradient method for general convex optimization problems. This includes defining the concept of the inexact model of the objective, illustrating it by examples, presenting the algorithm and its convergence rate theorem together with its proof. Section 3 presents the setting for statistically preconditioned distributed algorithms, introduces InSPAG algorithm and its convergence rate theorem. After that, we present the Hyperfast second-order method for the auxiliary subproblem of the InSPAG, estimate its complexity and combine the building blocks to obtain the total complexity of the whole approach. We finish this section by discussing the regime in which our approach is superior to applying stochastic variance-reduced algorithms. Section 4 presents our experimental results. For the sake of completeness in Section 5 we present Hyperfast second-order method for uniformly convex functions. We finalize with conclusions in Section 6.

2. Accelerated gradient method under inexactness and relative smoothness

In this section, we propose a general accelerated first-order algorithm that will be used in the next section to propose our InSPAG method for distributed optimization. We believe that the results of this section may be of independent interest. This section is, to an extent, independent of the other sections and the reader interested in the distributed optimization may skip this section since in what follows only the main result of this section (Theorem 2.6) will be used. We consider the following general optimization problem

$$\min_{x\in Q} f(x), \tag{5}$$

where Q is a convex subset of finite-dimensional vector space E. Our goal is to develop a general accelerated inexact gradient method capable to work under relative smoothness and strong convexity assumptions [29,30]. We consider two types of inexactness: inexact information on the objective function and inexact generalized projection.

Before we give more details, we introduce some general notations. Let E be an d-dimensional real vector space and E^* be its dual. We denote the value of a linear function $g \in E^*$ at $x \in E$ by $\langle g, x\rangle$. Let $\|\cdot\|$ be some norm on E, $\|\cdot\|_*$ be its dual, defined by $\|g\|_* = \max\limits_{x} \big\{\langle g, x\rangle, \|x\| \leq 1\big\}$. Let ϕ be a convex function on Q, which is continuously differentiable on the relative interior ri$Q$ of Q. Let $D_\phi[y](x) = \phi(x) - \phi(y) - \langle \nabla\phi(y), x - y\rangle$, $x \in Q, y \in \text{ri}Q$ be the corresponding Bregman divergence. Based on the Bregman divergence we introduce the following two definitions of inexactness.

Definition 2.1 *(Inexact model [31]).* Let $\delta, L, \mu, m \geq 0$. We say that $(f_\delta(y), \psi_\delta(x, y))$ is a $(\delta, L, \mu, m, \phi)$-model of the function f at a given point y iff, for all $x \in Q$,

$$\mu D_\phi[y](x) \leq f(x) - (f_\delta(y) + \psi_\delta(x, y)) \leq L D_\phi[y](x) + \delta, \tag{6}$$

$\psi_\delta(x, y)$ is convex in x, satisfies $\psi_\delta(x, x) = 0$ for all $x \in Q$ and

$$\psi(x) \geqslant \psi(z) + \langle g, x - z\rangle + m D_\phi[z](x), \quad \forall x, z \in Q,\ \forall g \in \partial\psi(z), \tag{7}$$

where for fixed $y \in Q$ and any $x \in Q$ we denote $\psi(x) = \psi_\delta(x, y)$.

Definition 2.2 *(Inexact generalized projection [32]).* For a convex optimization problem $\min_{x\in Q} \Psi(x)$ and $\widetilde{\delta} \geq 0$, we denote by $\text{Arg}\min^{\widetilde{\delta}}_{x\in Q} \Psi(x)$ a set of points $\widetilde{x}$ such that

$$\exists h \in \partial\Psi(\widetilde{x}) : \forall x \in Q \ \rightarrow\ \langle h, x - \widetilde{x}\rangle \geq -\widetilde{\delta}. \tag{8}$$

We denote by $\arg\min^{\widetilde{\delta}}_{x\in Q} \Psi(x)$ some element of $\text{Arg}\min^{\widetilde{\delta}}_{x\in Q} \Psi(x)$.

Optimization algorithms with inexact model of the objective were extensively studied in [31] and are generalizations of first-order algorithms with inexact oracle [33,34]. We now give two particular examples that are covered by the inexact model framework and refer to [31] for further examples.

Example 2.3. Relative smoothness and relative strong convexity, [29,30]. Assume that $\phi(x)$ is differentiable, and in (5), the objective f is differentiable, relatively smooth [29,30] and strongly convex [30] relative to ϕ, i.e., for some $\mu \geq 0$ and $L > 0$,

$$\mu D_\phi[y](x) \leq f(x) - f(y) - \langle \nabla f(y), x - y\rangle \leq L D_\phi[y](x),\ \forall x, y \in Q.$$

Then, clearly, Definition 2.1 holds with $m = 0$, $\delta = 0$, $f_\delta(y) = f(y)$, $\psi_\delta(x, y) = \langle \nabla f(y), x - y\rangle$. Importantly, the function ϕ is not required to be strongly convex. Our InSPAG relies on this particular example.

Example 2.4. Composite optimization, [35,36]. Assume that in (5), $f(x) = g(x) + h(x)$ with convex L-smooth w.r.t. norm $\|\cdot\|$ term $g(x)$ and simple convex term $h(x)$ which is usually called composite. In this case we assume that $\phi(x)$ is 1-strongly-convex w.r.t $\|\cdot\|$, and define $f_\delta(y) = g(y)+h(y)$ and $\psi_\delta(x, y) = \langle \nabla g(y), x-y\rangle+h(x)-h(y)$. Then, clearly,

$$f(x) - (f_\delta(y) + \psi_\delta(x, y)) = g(x) - (g(y) + \langle \nabla g(y), x - y\rangle).$$

By convexity of g, we have $0 \leq g(x) - (g(y) + \langle \nabla g(y), x - y\rangle)$. At the same time, by the L-smoothness of g and 1-strong-convexity of $\phi(x)$,

$$g(x) - (g(y) + \langle \nabla g(y), x - y\rangle) \leq \frac{L}{2}\|x - y\|^2 \leq LD_\phi[y](x).$$

From the combination of the above two relations, it is clear that (6) holds with $\delta = 0$ and $\mu = 0$ and we are in the situation of Definition 2.1 with $m = 0$ since $\psi_\delta(x, y)$ is convex in x.

In [31], to develop an accelerated algorithm, the authors use a different assumption where in the r.h.s. of (6) the Bregman divergence $D_\phi[y](x)$ is substituted with $\frac{1}{2}\|x-y\|^2$, and assume that ϕ is 1-strongly-convex w.r.t. $\|\cdot\|$. This, unfortunately, restricts the range of applications of the algorithm, and we use a weaker set of assumptions in Definition 2.1. At the same time, [14] showed that it is not possible to develop an accelerated algorithm in the relative smoothness setting without additional assumptions. Thus, we introduce the following assumption on the Bregman divergence $D_\phi[y](x)$ and note that the range of applications is still wider than for the approach of [31]. We also note that this assumption is simpler than the one in [1] and is a version of triangle scaling gain introduced in [37] and triangle lower bound property of [38].

Assumption 2.5. There exists a constant $G \geq 1$ such that for all $x, y, u, u_+ \in \mathrm{ri}\,\mathrm{dom}\phi$ such that $x - y = \tau(u_+ - u)$ for some $\tau \in [0, 1]$ it holds that

$$D_\phi[y](x) \leq G\tau^2 D_\phi[u](u_+). \tag{9}$$

This assumption can be seen as a relaxation of homogeneity of degree 2. The simplest example when this property holds is when $D_\phi[y](x) = \frac{1}{2}\|y-x\|^2$. In this case $G = 1$. We also note that our algorithm is adaptive to constant G which means that the property (9) is sufficient to hold only locally.

The proposed accelerated gradient method with inexact model is listed below as Algorithm 1. Unlike [1,37,38], our algorithm is simultaneously adaptive to the "Lipschitz" constant L (see Definition 2.1) and constant G in Assumption 2.5, which is expressed in constant M that plays the role of the product LG. Also, unlike [1,37,38], our algorithm allows two types of inexactness covered by Definitions 2.1 and 2.2. Finally, unlike [37,38], our algorithm has linear convergence when $\mu > 0$. We also note that we allow the accuracies $\delta, \widetilde{\delta}$ in Definition 2.1 and 2.2 to depend on the iteration counter k, which is expressed by the sequences $\{\delta_k, \widetilde{\delta}_k\}_{k\geq 0}$.

Algorithm 1 Accelerated gradient method with $(\delta, L, \mu, m, \phi)$-model.

```
1: Input: $x_0$ is the starting point, $\mu \geq 0$, $m \geq 0$, $\{\delta_k\}_{k\geq 0}$ and $L_0 > 0$.
2: Set $y_0 := x_0$, $u_0 := x_0$, $\alpha_0 := 0$, $A_0 := \alpha_0$
3: for $k \geq 0$ do
4:     Find the smallest integer $i_k \geq 0$ such that
           $$f_{\delta_k}(x_{k+1}) \leq f_{\delta_k}(y_{k+1}) + \psi_{\delta_k}(x_{k+1}, y_{k+1}) + \frac{M_{k+1}\alpha_{k+1}^2}{A_{k+1}^2} D_\phi[u_k](u_{k+1}) + \delta_k, \quad (10)$$
       where $M_{k+1} = 2^{i_k - 1} M_k$, $\alpha_{k+1}$ is the largest root of the equation
           $$A_{k+1}(1 + A_k\mu + A_k m) = M_{k+1}\alpha_{k+1}^2, \quad A_{k+1} := A_k + \alpha_{k+1}, \text{ and} \quad (11)$$
           $$y_{k+1} := \frac{\alpha_{k+1}u_k + A_k x_k}{A_{k+1}}, \quad (12)$$
           $$\Phi_{k+1}(x) := \alpha_{k+1}\psi_{\delta_k}(x, y_{k+1}) + (1 + A_k(\mu + m))D_\phi[u_k](x) + \alpha_{k+1}\mu D_\phi[y_{k+1}](x),$$
           $$u_{k+1} := \arg\min_{x\in Q}{}^{\widetilde{\delta}_k} \Phi_{k+1}(x), \quad \text{for some } \widetilde{\delta}_k \geq 0 \quad (13)$$
           $$x_{k+1} := \frac{\alpha_{k+1}u_{k+1} + A_k x_k}{A_{k+1}}. \quad (14)$$
5:     Set $k := k + 1$.
6: end for
7: Output: $x_k$
```

The following is the convergence rate result for the proposed algorithm.

Theorem 2.6. *Assume that $(f_\delta(y), \psi_\delta(x, y))$ is a $(\delta, L, \mu, m, \phi)$-model according to Definition 2.1. Also assume that $D_\phi[y](x)$ satisfies Assumption 2.5. Then, after N iterations of Algorithm 1, we have*

$$f(x_N) - f(x_*) \leq \frac{D_\phi[u_0](x_*)}{A_N} + \frac{2\sum_{k=0}^{N-1} A_{k+1}\delta_k}{A_N} + \frac{\sum_{k=0}^{N-1}\widetilde{\delta}_k}{A_N}, \tag{15}$$

$$D_\phi[u_N](x_*) \leq \frac{D_\phi[u_0](x_*)}{(1 + A_N\mu + A_N m)} + \frac{2\sum_{k=0}^{N-1} A_{k+1}\delta_k}{(1 + A_N\mu + A_N m)} + \frac{\sum_{k=0}^{N-1}\widetilde{\delta}_k}{(1 + A_N\mu + A_N m)}. \tag{16}$$

In order to prove Theorem 2.6 we need the following technical Lemma.

Lemma 2.7 *([31, Lemma 3.5.]). Let $\psi(x)$ be a relatively m-strongly convex function relative to ϕ with $m \geq 0$, i.e. (7) holds, and*

$$y = \arg\min_{x\in Q}{}^{\widetilde{\delta}}\{\psi(x) + \beta D_\phi[z](x) + \gamma D_\phi[u](x)\},$$

where $\beta \geq 0$ and $\gamma \geq 0$. Then, for all $x \in Q$,

$$\psi(x)+\beta D_\phi[z](x)+\gamma D_\phi[u](x) \geq \psi(y)+\beta V[z](y)+\gamma D_\phi[u](y)+(\beta+\gamma+m)D_\phi[y](x)-\widetilde{\delta}.$$

Proof of Theorem 2.6. We start by proving the correctness of the algorithm, i.e. that if we fix iteration k, there exists $i_k \geq 0$ such that (10) holds. By Definition 2.1 with $x = y$, we have $f_{\delta_k}(y) \leq f(y)$. Thus, from (6)

$$f_{\delta_k}(x_{k+1}) \leq f_{\delta_k}(y_{k+1}) + \psi_{\delta_k}(x_{k+1}, y_{k+1}) + LD_\phi[y_{k+1}](x_{k+1}) + \delta_k. \tag{17}$$

Combining this with Assumption 2.5 and using (12), (14), we further obtain

$$f_{\delta_k}(x_{k+1}) \leq f_{\delta_k}(y_{k+1}) + \psi_{\delta_k}(x_{k+1}, y_{k+1}) + \frac{LG\alpha_{k+1}^2}{A_{k+1}^2} D_\phi[u_k](u_{k+1}) + \delta_k. \tag{18}$$

Since $M_{k+1} = 2^{i_k-1}M_k$, we see that as soon as $M_{k+1} \geq LG$, (10) holds. Thus, the algorithm is correctly defined. Note also that by the same reason we have

$$M_{k+1} \leq 2LG. \tag{19}$$

Our next goal is to prove that, for all $x \in Q$, we have

$$\begin{aligned} &A_{k+1}f(x_{k+1}) - A_k f(x_k) + (1 + A_{k+1}\mu + A_{k+1}m)D_\phi[u_{k+1}](x) \\ &- (1 + A_k\mu + A_k m)D_\phi[u_k](x) \leq \alpha_{k+1}f(x) + 2\delta_k A_{k+1} + \widetilde{\delta}_k. \end{aligned} \tag{20}$$

Since by Definition 2.1 with $x = y$, we get $f(x) - \delta_k \leq f_{\delta_k}(x) \leq f(x)$, and, using (10), we have

$$\begin{aligned} f(x_{k+1}) \overset{(6)}{\leq} f_{\delta_k}(x_{k+1}) + \delta_k \overset{(10)}{\leq} f_{\delta_k}(y_{k+1}) + \psi_{\delta_k}(x_{k+1}, y_{k+1}) \\ + \frac{M_{k+1}\alpha_{k+1}^2}{A_{k+1}^2} D_\phi[u_k](u_{k+1}) + 2\delta_k. \end{aligned}$$

Substituting in this expression definition (14) of the point x_{k+1}, using that $A_{k+1} = A_k + \alpha_{k+1}$ and that, by Definition 2.1, $\psi_{\delta_k}(\cdot, y)$ is convex, we have

$$\begin{aligned} f(x_{k+1}) \leq \frac{A_k}{A_{k+1}}\left(f_{\delta_k}(y_{k+1}) + \psi_{\delta_k}(x_k, y_{k+1})\right) + \frac{\alpha_{k+1}}{A_{k+1}}\left(f_{\delta_k}(y_{k+1}) + \psi_{\delta_k}(u_{k+1}, y_{k+1})\right) \\ + \frac{M_{k+1}\alpha_{k+1}^2}{A_{k+1}^2} D_\phi[u_k](u_{k+1}) + 2\delta_k. \end{aligned}$$

In view of the definition (11) of the sequence α_{k+1} and left inequality in (6), we obtain

$$\begin{aligned} f(x_{k+1}) \leq \frac{A_k}{A_{k+1}} f(x_k) + \frac{\alpha_{k+1}}{A_{k+1}} \Big(f_{\delta_k}(y_{k+1}) + \psi_{\delta_k}(u_{k+1}, y_{k+1}) \\ + \frac{1 + A_k\mu + A_k m}{\alpha_{k+1}} D_\phi[u_k](u_{k+1}) \Big) + 2\delta_k. \end{aligned} \tag{21}$$

By Lemma 2.7, for the optimization problem in (13) with $\psi(x) = \alpha_{k+1}\psi_{\delta_k}(x, y_{k+1})$, $\beta = 1 + A_k\mu + A_k m$, $z = u_k$, $\gamma = \alpha_{k+1}\mu$, and $u = y_{k+1}$, it holds that

$$\begin{aligned} &\alpha_{k+1}\psi_{\delta_k}(u_{k+1}, y_{k+1}) + (1 + A_k\mu + A_k m) D_\phi[u_k](u_{k+1}) + \alpha_{k+1}\mu D_\phi[y_{k+1}](u_{k+1}) \\ &\quad + (1 + A_{k+1}\mu + A_{k+1}m) D_\phi[u_{k+1}](x) - \widetilde{\delta}_k \\ &\leq \alpha_{k+1}\psi_{\delta_k}(x, y_{k+1}) + (1 + A_k\mu + A_k m) D_\phi[u_k](x) + \alpha_{k+1}\mu D_\phi[y_{k+1}](x). \end{aligned}$$

From the fact that $D_\phi[y_{k+1}](u_{k+1}) \geq 0$, we have

$$\begin{aligned} &\alpha_{k+1}\psi_{\delta_k}(u_{k+1}, y_{k+1}) + (1 + A_k\mu + A_k m) D_\phi[u_k](u_{k+1}) \\ &\leq \alpha_{k+1}\psi_{\delta_k}(x, y_{k+1}) + (1 + A_k\mu + A_k m) D_\phi[u_k](x) \\ &\quad - (1 + A_{k+1}\mu + A_{k+1}m) D_\phi[u_{k+1}](x) + \alpha_{k+1}\mu D_\phi[y_{k+1}](x) + \widetilde{\delta}_k. \end{aligned} \tag{22}$$

Combining (21) and (22), we obtain

$$\begin{aligned} f(x_{k+1}) \leq \frac{A_k}{A_{k+1}} f(x_k) + \frac{\alpha_{k+1}}{A_{k+1}} \Big(& f_{\delta_k}(y_{k+1}) + \psi_{\delta_k}(x, y_{k+1}) + \mu D_\phi[y_{k+1}](x) \\ & + \frac{1 + A_k\mu + A_k m}{\alpha_{k+1}} D_\phi[u_k](x) \\ & - \frac{1 + A_{k+1}\mu + A_{k+1}m}{\alpha_{k+1}} D_\phi[u_{k+1}](x) + \frac{\widetilde{\delta}_k}{\alpha_{k+1}} \Big) + 2\delta_k. \end{aligned}$$

We finish the proof of (20) applying the left inequality in (6):

$$\begin{aligned} f(x_{k+1}) \leq \frac{A_k}{A_{k+1}} f(x_k) + \frac{\alpha_{k+1}}{A_{k+1}} f(x) + \frac{1 + A_k\mu + A_k m}{A_{k+1}} D_\phi[u_k](x) \\ - \frac{1 + A_{k+1}\mu + A_{k+1}m}{A_{k+1}} D_\phi[u_{k+1}](x) + 2\delta_k + \frac{\widetilde{\delta}_k}{A_{k+1}}. \end{aligned}$$

We now telescope the inequality (20) for k from 0 to $N-1$ and take $x = x_*$:

$$\begin{aligned} A_N f(x_N) \leq & A_N f(x_*) + D_\phi[u_0](x_*) - (1 + A_N(\mu + m)) D_\phi[u_N](x_*) \\ & + 2\sum_{k=0}^{N-1} A_{k+1}\delta_k + \sum_{k=0}^{N-1} \widetilde{\delta}_k. \end{aligned} \tag{23}$$

Since $V[u_{k+1}](x_*) \geq 0$ for all $k \geq 0$, we have

$$A_N f(x_N) - A_N f(x_*) \leq D_\phi[u_0](x_*) + 2\sum_{k=0}^{N-1} A_{k+1}\delta_k + \sum_{k=0}^{N-1} \widetilde{\delta}_k.$$

The last inequality proves (15). Inequality (16) is a straightforward from (23) since $f(x) \geq f(x_*)$ for all $x \in Q$. $\square$

To finish the analysis of Algorithm 1 we estimate the growth rate of the sequence A_N. The result is proved in the same way as Lemma 3.7 in [31] with the change $L_k \to M_k$.

Lemma 2.8. *For all $N \geq 0$, we have*

$$\begin{aligned} A_N &\geq \max\left\{\frac{1}{4}\left(\sum_{k=0}^{N-1}\frac{1}{\sqrt{M_{k+1}}}\right)^2, \frac{1}{M_1}\prod_{k=1}^{N-1}\left(1+\sqrt{\frac{\mu+m}{4M_{k+1}}}\right)^2\right\} \\ &\geq \max\left\{\frac{N^2}{4\widetilde{M}_N}, \frac{1}{M_1}\exp\left(N\sqrt{\frac{\mu+m}{4\widetilde{M}_N}}\right)\right\}, \end{aligned}$$

where $\widetilde{M}_N^{-1/2} = \frac{1}{N}\sum_{k=0}^{N-1} M_{k+1}^{-1/2}$.

Note that from (19) we have that $\widetilde{M}_N^{-1/2} = \frac{1}{N}\sum_{k=0}^{N-1} M_{k+1}^{-1/2} \geq \frac{1}{\sqrt{2LG}}$, which leads to the following estimate for the convergence rate of Algorithm 1

$$\begin{aligned} f(x_N) - f(x_*) \leq & D_\phi[u_0](x_*) \min\left\{\frac{8LG}{N^2}, 2LG\exp\left(-N\sqrt{\frac{\mu+m}{8LG}}\right)\right\} \\ & + \frac{2\sum_{k=0}^{N-1} A_{k+1}\delta_k}{A_N} + \frac{\sum_{k=0}^{N-1}\widetilde{\delta}_k}{A_N}. \end{aligned}$$

3. Inexact statistically preconditioned accelerated gradient method

In this section, we return to the distributed empirical risk minimization problem (1), (4), where we deal with m machines or worker nodes, with sample size n at each. Moreover, without loss of generality we index the central node as node 1. Following the same algorithmic structure as DANE [2] and SPAG [1], we define a reference function

$$\phi(x) = \frac{1}{n}\sum_{i=1}^{n}\ell(x;\zeta_i) + \frac{\sigma}{2}\|x\|_2^2, \tag{24}$$

where the samples ζ_i are taken from the node which is chosen to be central. It is easy to see from (2) and (3) that $\phi(x)$ is L_ϕ-smooth, and μ_ϕ-strongly convex since it has a similar form as $F(x)$. The value of the parameter σ is set to be an upper bound that quantifies how similar the function F_1 is to F, i.e., we assume that with high probability, it holds that

$$\|\nabla^2 F(x) - \nabla^2 F_1(x)\|_2 \leq \sigma, \ \forall x \in \mathrm{dom} h \tag{25}$$

where the norm is the operator norm for matrices (i.e., the largest singular value). The rationale behind this statistical similarity assumption are statistical arguments that allow to show [1] that (25) holds with σ proportional to $\frac{1}{\sqrt{n}}$. Further, it follows that $F(x)$ is $L_{F/\phi}$-relatively smooth and $\mu_{F/\phi}$-relatively strongly convex with respect to $\phi(x)$ [12,1], i.e.,

$$\mu_{F/\phi} D_\phi[x](y) \leq D_F[x](y) \leq L_{F/\phi} D_\phi[x](y), \tag{26}$$

with $L_{F/\phi} = 1$, $\mu_{F/\phi} = \mu_F/(\mu_F + 2\sigma)$, and $\kappa_{F/\phi} = L_{F/\phi}/\mu_{F/\phi} = 1 + 2\sigma/\mu_F$.

Once the specific Bregman divergence has been defined based on statistical similarity and using the reference function (statistical preconditioner) $\phi(x)$ as in (24), distributed statistical preconditioning methods rely on Bregman proximal steps, where the algorithm needs to solve at every iteration the problem of the form (here $\alpha > 0$)

$$\arg\min_{x \in \mathbb{R}^d} \{\alpha(\langle \nabla F(z), x - z\rangle + h(x)) + D_\phi[u](x)\}. \tag{27}$$

Non-accelerated methods based on steps of the form (27) have an iteration complexity of $\widetilde{O}(\kappa_{F/\phi})$ [39,30,31]. Thus, statistical preconditioning allows for the relative condition number $\kappa_{F/\phi}$ to determine the convergence rate instead of κ_F. The authors in [1] showed that for quadratic functions $\sigma = \widetilde{O}(L_F/\sqrt{n})$, which implies $\kappa_{F/\phi} = 1 + \widetilde{O}(\kappa_F/\sqrt{n})$. Similarly, for non-quadratic functions $\sigma = \widetilde{O}(\kappa_F\sqrt{d/n})$, thus $\kappa_{F/\phi} = 1 + \widetilde{O}(\kappa_F\sqrt{d/n})$. This, in turn, leads to the total number of communication rounds $\widetilde{O}\left(\kappa_{F/\phi}\right)$, which is quantitatively better than for methods that do not use such statistical preconditioning [15,40,5]. A similar argument follows for accelerated algorithms, where the iteration complexity will be $\widetilde{O}\big(\kappa_{F/\phi}^{1/2}\big)$ [1].

Next, we study the building blocks of our approach to advance this line of works. First, we consider the inexact version of the SPAG algorithm [1] wherein each iteration subproblems of the form (27) are solved inexactly with such accuracy that the overall performance of the algorithm is affected only by a logarithmic factor. Notably, the required accuracy decreases as iterations go, meaning that the approximate solution's quality may not be high in the first iterations. Next, we introduce and analyze a Hyperfast second-order method for third-order smooth and uniformly convex functions, which we will apply to solve subproblems (27) in each iteration of our inexact SPAG (InSPAG) algorithm when $h(x) = 0$. Finally, we analyze the total complexity for the combination of InSPAG plus the Hyperfast second-order method to solve our problem of interest. This combination is advantageous because we only use first-order information on the individual losses from the whole dataset and obtain a small subproblem on the central node. Then, a fast second-order method is used to solve this subproblem on the central node.

3.1. InSPAG and its convergence rate theorem

This subsection introduces the InSPAG algorithm together with its convergence rate analysis. The main idea is to implement Algorithm 1 on the central node and use Theorem 2.6. Inexactness in statistically preconditioned problems has been studied for DANE, resulting in InexactDANE, AIDE [11], and D^{2}ANE [3]. To propose our InSPAG algorithm we rely on the results of Section 2. From (26) and Examples 2.3 and 2.4 we see that $f_\delta(y) = f(y)$ and $\psi_\delta(x, y) = \langle \nabla F(y), x - y\rangle + h(x) - h(y)$ constitute a $(0, L_{F/\phi}, \mu_{F/\phi}, 0, \phi)$-model of the function f defined in (1). Thus, the main idea of InSPAG is to implement Algorithm 1 for problem (1) using distributed computations. We further assume that the solution x_* of the problem (1) belongs to some Euclidean ball $B_2(0, R)$, and define $R_\phi^2 = 2L_\phi R^2$. Using this quantity we set the inexactness of the projection in each iteration to be $\widetilde{\delta}_k = \frac{R_\phi^2}{k}$ (cf. (13)).

The pseudocode of the proposed InSPAG algorithm is presented as Algorithm 2. Unlike [1], our algorithm is inspired by a similar-triangles type of accelerated methods [41–44,31,45], which leads to a slightly simpler algorithm. Another important difference with [1] is that our algorithm is adaptive simultaneously to the constants $L_{F/\phi}$

Algorithm 2 InSPAG $(L_{F/\phi}, \mu_{F/\phi}, x_0, R)$.

```
1: Input: $R$ s.t. $x_* \in B_2(0,R)$, $R_\phi^2 = 2L_\phi R^2$, $\mu_{F/\phi}$, $M_0$.
2: Set $y_0 = u_0 = x_0 \in B_2(0,R)$, $A_0 := \alpha_0 := 0$.
3: for $k \geq 0$ do
4:     Set $i_k = 0$
5:     repeat
6:         At the central node set $M_{k+1} = 2^{i_k - 1} M_k$ and find $\alpha_{k+1}$ from $A_{k+1}(1 + A_k \mu_{F/\phi}) = M_{k+1}\alpha_{k+1}^2$.
           Set $A_{k+1} := A_k + \alpha_{k+1}$.
7:         At the central node set $y_{k+1} := \frac{\alpha_{k+1} u_k + A_k x_k}{A_{k+1}}$ and send to each worker.
8:         At every worker node $j$ compute $\frac{1}{n}\sum_{i=1}^n \nabla \ell(y_{k+1}; \zeta_i^{(j)})$ and send it to the central node.
9:         At the central node compute $\nabla F(y_{k+1}) = \frac{1}{nm}\sum_{j=1}^m \sum_{i=1}^n \nabla \ell(y_{k+1}; \zeta_i^{(j)})$.
10:        At the central node solve $u_{k+1} = \arg\min^{R_\phi^2/k}_{x \in B_2(0,R)} \Phi_{k+1}(x)$,
               $$\text{where } \Phi_{k+1}(x) = \alpha_{k+1}(\langle \nabla F(y_{k+1}), x - y_{k+1}\rangle + h(x)) + $$
               $$+ (1 + A_k \mu_{F/\phi}) D_\phi[u_k](x) + \alpha_{k+1}\mu_{F/\phi} D_\phi[y_{k+1}](x). \quad (28)$$
11:        At the central node set $x_{k+1} := \frac{\alpha_{k+1} u_{k+1} + A_k x_k}{A_{k+1}}$.
12:        Set $i_k = i_k + 1$.
13:    until
           $$F(x_{k+1}) \leq F(y_{k+1}) + \langle \nabla F(y_{k+1}), x_{k+1} - y_{k+1}\rangle + \frac{M_{k+1}\alpha_{k+1}^2}{A_{k+1}^2} D_\phi[u_k](u_{k+1}). \quad (29)$$
14: end for
15: Output: $x_k$
```

and G (see Assumption 2.5), which may lead to further acceleration in practice since locally, the constant $L_{F/\phi}G$ can be smaller leading to larger step-sizes. Note that Line 10 of Algorithm 2 requires approximate minimization of the auxiliary function (28). First, we present the complexity analysis of Algorithm 2 in Theorem 3.1 assuming the approximate solution to (28). In Subsection 3.2, we show the complexity of obtaining said approximate solution efficiently when $h(x) = 0$ using high-order methods.

We are now in a position to state the main result on InSPAG.

Theorem 3.1. *Assume that the function F in (1) is $\mu_{F/\phi}$-strongly convex and $L_{F/\phi}$-smooth with respect to the function ϕ, where ϕ satisfies Assumption 2.5. Moreover, let x_k, $k \geq 0$ be the sequence generated by Algorithm 2. Then, after K iterations it holds that*

$$f(x_K) - f(x_*) \leq \frac{2L_\phi R^2(1 + \ln K)}{A_K}. \tag{30}$$

Moreover, the value A_K grows as follows:

$$A_K \geq \max\left\{\frac{K^2}{4\widetilde{M}_K}, \frac{1}{M_1}\exp\left(K\sqrt{\frac{\mu_{F/\phi}}{4\widetilde{M}_K}}\right)\right\}, \tag{31}$$

where $\widetilde{M}_K^{-1/2} = \frac{1}{K}\sum_{k=0}^{K-1} M_{k+1}^{-1/2}$.

Proof. Clearly, Algorithm 2 is a distributed implementation of Algorithm 1 with $\delta_k = 0$, $k \geq 0$. We only note that for this particular setting with $f_\delta(y) = f(y)$ and $\psi_\delta(x, y) = \langle \nabla F(y), x - y\rangle + h(x) - h(y)$, inequality (10) becomes

$$\begin{aligned} F(x_{k+1}) + h(x_{k+1}) \leq & F(y_{k+1}) + h(y_{k+1}) + \langle \nabla F(y_{k+1}), x_{k+1} - y_{k+1}\rangle \\ & + h(x_{k+1}) - h(y_{k+1}) + \frac{M_{k+1}\alpha_{k+1}^2}{A_{k+1}^2} D_\phi[u_k](u_{k+1}), \end{aligned}$$

which is equivalent to (29). Thus, we can apply Theorem 2.6, which gives the following estimate

$$\begin{aligned} f(x_K) - f(x_*) &\leq \frac{D_\phi[u_0](x_*)}{A_K} + \frac{\sum_{k=0}^{K-1}\widetilde{\delta}_k}{A_K} \leq \frac{L_\phi(2R)^2}{2A_K} + \frac{1}{A_K}\sum_{k=0}^{K-1}\frac{R_\phi^2}{k} \\ &\leq \frac{R_\phi^2(1 + \ln K)}{A_K} = \frac{2L_\phi R^2(1 + \ln K)}{A_K} \end{aligned}$$

The lower bound for A_K follows from Lemma 2.8. □

To apply Theorem 3.1 we need to ensure that Assumption 2.5 is satisfied.

Lemma 3.2. *Under the assumption that ϕ is μ_ϕ-strongly convex and L_ϕ-smooth Assumption 2.5 is satisfied with $G = L_\phi/\mu_\phi = \kappa_\phi$.*

Proof. Since ϕ is μ_ϕ-strongly convex and L_ϕ-smooth, we have that

$$\frac{\mu_\phi}{2}\|x-y\|^2 \le D_\phi[x](y) \le \frac{L_\phi}{2}\|x-y\|^2, \quad \forall x,y \in \operatorname{dom}\phi.$$

Thus, for all x, y, u, u_+ such that $x - y = \tau(u_+ - u)$ for some $\tau \in [0,1]$, we have

$$D_\phi[y](x) \le \frac{L_\phi}{2}\|x-y\|^2 = \frac{L_\phi\tau^2}{2}\|u_+-u\|^2 \le \frac{L_\phi\tau^2}{\mu_\phi}D_\phi[u](u_+). \quad \square$$

From Lemma 3.2, we see that if ϕ is a quadratic function, then, $G = \kappa_\phi$ and by (19) we have that $M_{k+1} \le 2L_{F/\phi}\kappa_\phi$. Then, the number of iterations K to reach accuracy ε, i.e., the number of communications between the central node and the worker nodes, is bounded as $O(\sqrt{\kappa_{F/\phi}\kappa_\phi}\ln\frac{1}{\varepsilon})$. As we see below, for quadratic functions the estimate for G can be improved to $G = 1$, which gives a better communication complexity $O(\sqrt{\kappa_{F/\phi}}\ln\frac{1}{\varepsilon})$. In the general case, where ϕ is not quadratic, similarly to [1,46], we next show that $M_{k+1} \to L_{F/\phi}$ linearly with rate $\widetilde{O}(\sqrt{\kappa_{F/\phi}})$. This means that the convergence rate of InSPAG quickly approaches the convergence rate with condition number $\sqrt{\kappa_{F/\phi}}$.

Lemma 3.3. *Under the assumptions of Theorem 3.1 and Lemma 3.2 assume additionally that the Hessian of ϕ is H-Lipschitz-continuous, i.e.*

$$\|\nabla^2\phi(x) - \nabla^2\phi(y)\| \le H\|x-y\|. \tag{32}$$

Then the inequality (29) is satisfied with

$$M_{k+1} = L_{F/\phi}\min\left\{\kappa_\phi, 1 + \frac{Hd_k}{\mu_\phi}\right\}, \tag{33}$$

where $d_k = \|x_{k+1} - y_{k+1}\| + \|u_k - x_k\| + \|u_k - u_{k+1}\|$.

Proof. By the local quadratic representation of the Bregman divergence, we have for any $a, b \in \operatorname{dom}\phi$ and for some $\tau \in [0,1]$ that $D_\phi[a](b) = \|a-b\|^2_{\nabla^2\phi(\tau a+(1-\tau)b)}$. We use $H(a,b)$ to denote the corresponding Hessian $\nabla^2\phi(\tau a + (1-\tau)b)$. We have

$$D_\phi[x_{k+1}](y_{k+1}) = \|x_{k+1} - y_{k+1}\|^2_{H(x_{k+1},y_{k+1})} \overset{(14),(12)}{=} \frac{\alpha_{k+1}^2}{A_{k+1}^2}\|u_{k+1} - u_k\|^2_{H(x_{k+1},y_{k+1})}$$

$$\le \frac{\alpha_{k+1}^2}{A_{k+1}^2}\left(\|u_{k+1} - u_k\|^2_{H(u_{k+1},u_k)} + \|H(x_{k+1},y_{k+1}) - H(u_{k+1},u_k)\|\|u_{k+1} - u_k\|^2\right)$$

$$\leq \frac{\alpha_{k+1}^2}{A_{k+1}^2}\left(D_\phi[u_k](u_{k+1}) + \|H(x_{k+1}, y_{k+1}) - H(u_{k+1}, u_k)\| \frac{D_\phi[u_k](u_{k+1})}{\mu_\phi}\right)$$
$$\overset{(32)}{\leq} \frac{\alpha_{k+1}^2}{A_{k+1}^2} D_\phi[u_k](u_{k+1})\left(1 + \frac{H\|z - z'\|}{\mu_\phi}\right),$$

where $z \in [x_{k+1}, y_{k+1}]$ and $z' \in [u_{k+1}, u_k]$. Using the latter and (12), (14), we obtain

$$\|z - z'\| \leq \|z - y_{k+1}\| + \|y_{k+1} - u_k\| + \|u_k - z'\|$$
$$\leq \|x_{k+1} - y_{k+1}\| + \|x_k - u_k\| + \|u_k - u_{k+1}\| \triangleq d_k.$$

Combining the above with the relative smoothness property (26), we obtain that (29) holds when $M_{k+1} = L_{F/\phi}\left(1 + \frac{Hd_k}{\mu_\phi}\right)$. Since (29) holds also when $M_{k+1} = L_{F/\phi}\kappa_\phi$ (see Lemma 3.2 and (26)), we obtain the statement of the Lemma. $\square$

From (16) and (31) since $M_{k+1} \leq L_{F/\phi}\kappa_\phi$ we know that the sequence u_k, $k \geq 0$ converges to x_* linearly with condition number $\sqrt{\kappa_{F/\phi}\kappa_\phi}$. From (15) by the strong convexity, we see that the sequence x_k, $k \geq 0$ converges to x_* also linearly with the same condition number. Hence, by (12) we conclude the same on the sequence y_k, $k \geq 0$. Thus, d_k converges linearly to zero with the same condition number and M_{k+1} approaches $L_{F/\phi}$ with the same rate. This, in turn, means that the convergence rate in Theorem 3.1 quickly approaches $O((1 - \sqrt{\kappa_{F/\phi}})^K)$ when the Hessian of ϕ is Lipschitz-continuous.

Next, we study the properties of the auxiliary problem in step 10 of Algorithm 2 and, under the additional assumption that the loss function ℓ has bounded fourth-order derivatives, we show the explicit complexity of computing an approximate solution to this auxiliary problem using Hyperfast second-order methods.

3.2. Hyperfast second-order method for the auxiliary problem

In this subsection, we elaborate the properties of the auxiliary problem in step 10 of Algorithm 2 and propose a Hyperfast second-order algorithm to solve it when the function ϕ is strongly convex and sufficiently smooth. The main result is a complexity estimate for solving the auxiliary problem by the Hyperfast algorithm. Recall that, at each iteration of Algorithm 2 we need to find an approximate minimizer in the sense of Definition 2.2 of the function $\Phi_{k+1}(x)$ on the Euclidean ball $B_2(0, R)$. Throughout this subsection we assume that the regularizer $h(x) \equiv 0$.

We first study some properties of the function $\Phi_{k+1}(x)$ defined in (28) and the minimization problem solved in step 10 of Algorithm 2. Using our assumption that $h(x) = 0$, the fact that $A_{k+1} = A_k + \alpha_{k+1}$, the definition of the Bregman divergence, and ignoring constant terms in that problem, we see that it is equivalent to the problem $u_{k+1} = \arg\min_{x \in B_2(0,R)}^{R_\phi^2/k} \Psi_{k+1}(x)$, where

$$\Psi_{k+1}(x) \triangleq \langle \alpha_{k+1}\nabla F(y_{k+1}) - (1 + A_k\mu_{F/\phi})\nabla\phi(u_k) - \alpha_{k+1}\mu_{F/\phi}\nabla\phi(y_{k+1}), x\rangle +$$

$$+ (1 + A_{k+1}\mu_{F/\phi})\phi(x). \tag{34}$$

Lemma 3.4. *Assume that ϕ is μ_ϕ-strongly convex and L_ϕ-smooth w.r.t. the Euclidean norm. Also assume that for some $\theta > 0$ and all $x \in B_2(0,R)$, it holds that $\max\{\|\nabla F(x)\|_2/\mu_{F/\phi}, \|\nabla\phi(x)\|_2\} \le \theta$. Let us denote $x^*_{k+1} = \arg\min_{x\in B_2(0,R)} \Psi_{k+1}(x)$ and let the point $\hat{x}_{k+1}$ satisfy*

$$\Psi_{k+1}(\hat{x}_{k+1}) - \Psi_{k+1}(x^*_{k+1}) \le \Delta_k \triangleq \frac{\mu_\phi R_\phi^4}{2k^2(2L_\phi R + 3\theta)^2(1 + A_{k+1}\mu_{F/\phi})}. \tag{35}$$

Then $\hat{x}_{k+1} = \arg\min^{R_\phi^2/k}_{x\in B_2(0,R)} \Psi_{k+1}(x)$.

Proof. Since ϕ is μ_ϕ-strongly convex and L_ϕ-smooth, Ψ_{k+1} in (34) is μ_Ψ-strongly convex with $\mu_\Psi = (1 + A_{k+1}\mu_{F/\phi})\mu_\phi$ and L_Ψ-smooth with $L_\Psi = (1 + A_{k+1}\mu_{F/\phi})L_\phi$. Further, by the assumption of the lemma, we have, for all $x \in B_2(0,R)$,

$$\begin{aligned}&\|\nabla\Psi_{k+1}(x)\|_2 = \|\alpha_{k+1}\nabla F(y_{k+1}) - (1 + A_k\mu_{F/\phi})\nabla\phi(u_k) - \alpha_{k+1}\mu_{F/\phi}\nabla\phi(y_{k+1}) \\ &+ (1 + A_{k+1}\mu_{F/\phi})\nabla\phi(x)\|_2 \le 3(1 + A_{k+1}\mu_{F/\phi})\theta,\end{aligned} \tag{36}$$

where we used also that $\alpha_{k+1} \le A_{k+1}$ and that $A_{k+1} = A_k + \alpha_{k+1}$. By the strong convexity of Ψ, we have

$$\|\hat{x}_{k+1} - x^*_{k+1}\|_2 \le \sqrt{\frac{2}{\mu_\Psi}(\Psi_{k+1}(\hat{x}_{k+1}) - \Psi_{k+1}(x^*_{k+1}))} \le \sqrt{2\Delta_k/\mu_\Psi}. \tag{37}$$

Hence, for any $x \in B_2(0,R)$,

$$\begin{aligned}&\langle\nabla\Psi_{k+1}(\hat{x}_{k+1}), x - \hat{x}_{k+1}\rangle = \langle\nabla\Psi_{k+1}(\hat{x}_{k+1}) - \nabla\Psi_{k+1}(x^*_{k+1}), x - \hat{x}_{k+1}\rangle \\ &+ \langle\nabla\Psi_{k+1}(x^*_{k+1}), x - x^*_{k+1}\rangle + \langle\nabla\Psi_{k+1}(x^*_{k+1}), x^*_{k+1} - \hat{x}_{k+1}\rangle \\ &\ge -L_\Psi\|x^*_{k+1} - \hat{x}_{k+1}\|_2\|x - \hat{x}_{k+1}\|_2 + 0 - \|\nabla\Psi_{k+1}(x^*_{k+1})\|_2\|x^*_{k+1} - \hat{x}_{k+1}\|_2 \\ &\overset{(36),(37)}{\ge} -(2L_\Psi R + 3(1 + A_{k+1}\mu_{F/\phi})\theta)\sqrt{2\Delta_k/\mu_\Psi} \\ &= -(1 + A_{k+1}\mu_{F/\phi})(2L_\phi R + 3\theta)\sqrt{\frac{2\Delta_k}{(1 + A_{k+1}\mu_{F/\phi})\mu_\phi}} \ge -R_\phi^2/k\end{aligned}$$

where we used the definitions of L_Ψ and μ_Ψ and the expression for Δ_k. Thus, $\hat{x}_{k+1}$ satisfies Definition 2.2 with $\widetilde{\delta} = R_\phi^2/k$. □

Next, we propose an efficient Hyperfast second-order method to obtain a point $\hat{x}_{k+1}$ for which (35) holds. To do this, we make an additional assumption on the function ϕ.

Assumption 3.5. The function ϕ has bounded fourth-order derivatives, which is equivalent to Lipschitz third-order derivative, i.e. there exists $0 \leq L_{\phi,3} < \infty$ s.t.

$$\|\nabla^3 \phi(x) - \nabla^3 \phi(y)\|_2 \leq L_{\phi,3}\|x - y\|_2, \;\; \forall x, y \in B_2(0, R),$$

where the norm of a tensor is induced by the Euclidean norm in a standard way [22].

The idea is to use a second-order implementation of a third-order method, in the sense of [25, Sect. 5.2] or [26, Algorithm 2], to minimize $\Psi_{k+1}(x)$ in each iteration of InSPAG. Such methods are called Hyperfast second-order methods since, due to the additional assumption of third-order smoothness, they have faster convergence rates than the optimal second-order method [47]. In our case, the objective $\Psi_{k+1}(x)$ is additionally strongly convex. Thus, we can achieve faster rates than the basic schemes in [25,26] that do not use strong convexity. We propose an extension of Hyperfast second-order methods for minimizing strongly convex functions and show that they have faster convergence rate.[3] Our algorithm is described below as Algorithm 3.

Algorithm 3 Restarted hyperfast second-order method.

```
Require: $z_0 \in B_2(0, R)$, constant $c$ which defines convergence rate of the basic Hyperfast method, strong
    convexity parameter $\mu_\phi$.
 1: Set $R_0 = 2R$
 2: for $t = 0, 1, \ldots$ do
 3:     Set $R_t = R_0 \cdot 2^{-k}$, and $N_t = \max\{\lceil (8cL_{\phi,3}R_t^2/\mu_\phi)^{\frac{1}{5}} \rceil, 1\}$,
 4:     Set $z_{t+1} = y_{N_t}$ as the output of the basic Hyperfast Second-Order Method (either [25, Eq.3.6]
        for $p = 3$ and $\beta = 1/2$ and with auxiliary steps described in [25, Sect. 5.2] or [26, Algorithm 2])
        started from $z_t$ and run for $N_t$ steps applied to $\Psi_{k+1}(x)$.
 5:     Set $t = t + 1$.
 6: end for
Ensure: $z_t$.
```

As a building block, this method uses basic Hyperfast second-order method which has convergence rate of the form $cL_3\|x_* - z_0\|_2^4/k^5$, where k is the iteration counter, $c = 48$ for [25, Theorem 2] and $c = 35$ for [26, Theorem 2].

Theorem 3.6. *Under assumptions of Lemma 3.4 let additionally Assumption 3.5 to hold. Let also sequence z_t, $t \geq 0$ be generated by Algorithm 3. Then*

$$\frac{\mu_\Psi}{2}\|z_t - x_{k+1}^*\|_2^2 \leq \Psi_{k+1}(z_t) - \Psi_{k+1}(x_{k+1}^*) \leq 2\mu_\Psi R^2 \cdot 2^{-2t}, t \geq 0. \tag{38}$$

Moreover, the total number of steps of the basic Hyperfast second-order method to reach $\Psi_{k+1}(z_t) - \Psi_{k+1}(x_{k+1}^*) \leq \Delta_k$ is bounded by

[3] Section 5 extends Hyperfast second-order methods for a more general setting of minimizing uniformly convex functions. Here we use a particular case that corresponds to uniform convexity of the order $q = 2$, equivalent to strong convexity.

$$5\left(\frac{32cL_{\phi,3}R^2}{\mu_\phi}\right)^{\frac{1}{5}} + \log_2 \frac{(1+A_{k+1}\mu_{F/\phi})^2 k^2 (2L_\phi R + 3\theta)^2}{L_\phi^2 R^2}.$$

Proof. Let us denote for shortness $x^* = x^*_{k+1}$ and $\Psi(x) = \Psi_{k+1}(x)$. For $t = 0$ we have $\|x^* - z_0\|_2 \le R_0$. Let us assume that $\|x^* - z_t\|_2 \le R_t$ and show that $\|x^* - z_{t+1}\|_2 \le R_{t+1}$. By Assumption 3.5 and (34) it is clear that $\Psi(x)$ has $L_{\Psi,3}$-Lipschitz third-order derivative with $L_{\Psi,3} = (1+A_{k+1}\mu_{F/\phi})L_{\phi,3}$. Recall that $\mu_\Psi = (1+A_{k+1}\mu_{F/\phi})\mu_\phi$. From [25][Theorem 2] since Ψ is μ_Ψ-strongly convex and has $L_{\Psi,3}$-Lipschitz third-order derivative, it holds that

$$\frac{\mu_\Psi}{2}\|z_{t+1} - x^*\|_2^2 \le \Psi(z_{t+1}) - \Psi(x^*) \le \frac{cL_{\Psi,3}\|z_t - x^*\|_2^4}{N_t^5} \le \frac{\mu_\Psi (R_t/2)^2}{2} = \frac{\mu_\Psi R_{t+1}^2}{2}$$

by the choice of N_t and since $L_{\Psi,3}/\mu_\Psi = L_{\phi,3}/\mu_\phi$. Thus, by induction, we have (38).

It remains to estimate the number of iterations of the basic Hyperfast method. From (38) we see that to reach the accuracy Δ_k it is sufficient to make $T = \frac{1}{2}\log_2 \frac{2\mu_\Psi R^2}{\Delta_k}$ restarts. Summing up the number of operations $N_t,\ t = 0, ..., T$, we obtain

$$\sum_{t=0}^{T} N_t \le \sum_{t=0}^{T}\left[\left(\frac{8cL_{\phi,3}R_t^2}{\mu_\phi}\right)^{\frac{1}{5}} + 1\right] = \left(\frac{8cL_{\phi,3}R_0^2}{\mu_\phi}\right)^{\frac{1}{5}} \sum_{t=0}^{T} 2^{-\frac{2t}{5}} + T$$
$$\le 5\left(\frac{32cL_{\phi,3}R^2}{\mu_\phi}\right)^{\frac{1}{5}} + \log_2 \frac{2\mu_\Psi R^2}{\Delta_k}.$$

Let us estimate the last term using (35) and that $\mu_\Psi = (1+A_{k+1}\mu_{F/\phi})\mu_\phi$, $R_\phi^2 = 2L_\phi R^2$:

$$\log_2 \frac{2\mu_\Psi R^2}{\Delta_k} = \log_2 \frac{2(1+A_{k+1}\mu_{F/\phi})\mu_\phi R^2}{\frac{\mu_\phi (2L_\phi R^2)^2}{2k^2(2L_\phi R + 3\theta)^2(1+A_{k+1}\mu_{F/\phi})}}$$
$$= \log_2 \frac{(1+A_{k+1}\mu_{F/\phi})^2 k^2 (2L_\phi R + 3\theta)^2}{L_\phi^2 R^2}.$$

Combining this with the previous chain of inequalities, we obtain the second statement of the lemma. □

3.3. *InSPAG plus hyperfast method with application to logistic regression*

This subsection combines the building blocks introduced in the previous two subsections and considers a particular application to a regularized logistic regression problem, for which we obtain a total complexity bound in terms of the number of iterations of the Hyperfast second-order method. We further discuss the arithmetic iteration complexity of our method and compare it to that of stochastic variance-reduced first-order algorithms and indicate a regime in which our algorithm is preferable.

Combining Theorems 3.1 and 3.6, we obtain the following result.

Theorem 3.7. *Assume that in problem (1), $h(x) = 0$, and that its solution x_* belongs to the ball $B_2(0, R)$. Assume that the function F in this problem is $\mu_{F/\phi}$-strongly convex and $L_{F/\phi}$-smooth with respect to the function ϕ, where ϕ satisfies Assumption 2.5, is μ_ϕ-strongly convex, L_ϕ-smooth and has $L_{\phi,3}$-Lipschitz third-order derivative. Also assume that for some $\theta > 0$ and all $x \in B_2(0, R)$, it holds that $\max\{\|\nabla F(x)\|_2/\mu_{F/\phi}, \|\nabla\phi(x)\|_2\} \leq \theta$. Let $\varepsilon > 0$ be the target accuracy. Finally, let InSPAG (Algorithm 2) be applied to problem (1), and in step 10 of this algorithm let Restarted Hyperfast method (Algorithm 3) be applied to solve the auxiliary problem. Then a sufficient number of iterations of the basic Hyperfast method to find an ε-solution to (1) is bounded as*

$$O\left(K\left(\frac{L_{\phi,3}R^2}{\mu_\phi}\right)^{\frac{1}{5}} + K\log_2\frac{\mu_{F/\phi}L_\phi R^2(L_\phi R+\theta)K\ln K}{L_\phi R\varepsilon}\right), \tag{39}$$

where K is such that $\frac{2L_\phi R^2(1+\ln(K+1))}{A_{K+1}} \leq \varepsilon < \frac{2L_\phi R^2(1+\ln K)}{A_K}$.

Proof. From (30) we see that InSPAG can be stopped at iteration K when we have $\frac{2L_\phi R^2(1+\ln(K+1))}{A_{K+1}} \leq \varepsilon < \frac{2L_\phi R^2(1+\ln K)}{A_K}$. Then, $f(x_{K+1}) - f(x_*) \leq \varepsilon$. Also, applying Theorem 3.6 we obtain that the total number of iterations of the basic Hyperfast method, up to numerical constant multipliers, is bounded by

$$\begin{aligned}
&\sum_{k=0}^{K}\left(\left(\frac{L_{\phi,3}R^2}{\mu_\phi}\right)^{\frac{1}{5}} + \log_2\frac{(1+A_k\mu_{F/\phi})k(L_\phi R+\theta)}{L_\phi R}\right)\\
&\leq_c K\left(\left(\frac{L_{\phi,3}R^2}{\mu_\phi}\right)^{\frac{1}{5}} + \log_2\frac{(1+A_K\mu_{F/\phi})K(L_\phi R+\theta)}{L_\phi R}\right) = (39),
\end{aligned}$$

where in equality $\leq_c$ means a usual inequality up to a numerical constant factor. □

From (31) and Lemma 3.3 we know that when ϕ has also Lipschitz Hessian, it is sufficient to take $K = O\left(\sqrt{\kappa_{F/\phi}\kappa_\phi}\ln\frac{1}{\varepsilon}\right)$. Lemma 3.3 also implies that for quadratic function ϕ it is sufficient to take $K = O\left(\sqrt{\kappa_{F/\phi}}\ln\frac{1}{\varepsilon}\right)$ and that for non-quadratic function ϕ the result is the same up to a fast asymptotic. In the language of the individual loss ℓ and the number of samples n used for preconditioning, our result is the same $\widetilde{O}(\sqrt{\kappa_\ell}/n^{1/4})$ as for the exact algorithm [1]. Thus, the total number of iterations of the basic Hyperfast method to find an ε-solution to (1) can be bounded as

$$\widetilde{O}\left(\sqrt{\kappa_{F/\phi}}\left(\frac{L_{\phi,3}R^2}{\mu_\phi}\right)^{\frac{1}{5}}\right). \tag{40}$$

So far, we have not explicitly used the finite-sum structure of problem (1), (2) and the statistical similarity (25). In order to do this, we consider the sparse empirical risk minimization problem with regularized logistic loss, where in (2), for $i \in \{1, \dots, N\}$,

$$\ell(x;\zeta_i) = \log\left(1 + \exp(-\eta_i \langle x, \xi_i \rangle)\right) + \lambda_1 \sum_{j \in I_S} x_j^2 + \lambda_2 \sum_{j \in I_D} x_j^2, \tag{41}$$

where $\zeta_i = (\xi_i, \eta_i)$, $\eta_i = 1$ indicates a positive (clicked) example, and $\eta_i = -1$ otherwise. We assume there are two types of features, namely, sparse and dense features. Let $\xi_{i,j}$ be the j-th element of the vector ξ_i. Then, $\xi_{i,j}$ is a sparse feature if $\xi_{i,j} = 0$ for almost all $i \in \{1, \dots, N\}$, and a dense feature if $\xi_{i,j} \neq 0$ for many $i \in \{1, \dots, N\}$. We denote by I_S (and I_D) the set of sparse (and dense) features with $I_S \cup I_D = \{1, \dots, d\}$ and $I_S \cap I_D = \emptyset$. Moreover, it follows from [48, Section 4.4] that in this case the function F is L_F-smooth with $L_F = \max\{\lambda_1, \lambda_2\} + \frac{1}{N}\sum_{i=1}^N \|\eta_i \xi_i\|_2^2 = O(s)$, where s is the average number of nonzero elements in ξ_i, and μ_F-strongly convex with $\mu_F = \min\{\lambda_1, \lambda_2\}$. For the same reasons, function ϕ defined in (24) is L_ϕ-smooth with $L_\phi = \max\{\lambda_1, \lambda_2\} + \frac{1}{n}\sum_{i=1}^n \|\eta_i \xi_i\|_2^2 + \sigma$ and μ_ϕ-strongly convex with $\mu_\phi = \min\{\lambda_1, \lambda_2\} + \sigma$. It also has bounded first-, second, and third-order derivatives [17]. More importantly, the logistic loss in (41) has bounded fourth-order derivatives [17], which means that Assumption 3.5 holds. Indeed, let us define matrix $A = [\eta_1 \xi_1, \dots, \eta_n \xi_n]^\top$. Then, by Theorem 5.4 in [17] with $\mu = 1$ the function $\frac{1}{n}\sum_{i=1}^n \ell(x;\zeta_i)$ has Lipschitz third-order derivative with constant $L_{\ell,3} = 15\|A^\top A\|_2^2$ w.r.t. 2-norm or with constant $L_{\ell,3} = 15$ w.r.t. $\|\cdot\|_{A^\top A}$-norm. Since adding a quadratic function does not change the Lipschitz constant for the third-order derivative, ϕ has Lipschitz third-order derivative with constant $L_{\phi,3} = L_{\ell,3}$.

Applying [1, Theorem 3], we obtain that in our setting the statistical similarity parameter in (25) is $\sigma = 1 + \widetilde{O}\left(\frac{\max_{i=1,\dots,n} \|\eta_i \xi_i\|_2^{3/2} R}{\min\{\lambda_1, \lambda_2\}\sqrt{n}}\right)$ and a sufficient number of InSPAG iterations is $\widetilde{O}(\sqrt{\kappa_\ell}/n^{1/4})$, which is similar to SPAG [1]. Further, the number of the basic Hyperfast iterations is the same up to a factor

$$\left(\frac{L_{\phi,3} R^2}{\mu_\phi}\right)^{\frac{1}{5}} \leq_c \left(\frac{\|A^\top A\|_2^2 R^2}{\min\{\lambda_1, \lambda_2\} + \sigma}\right)^{\frac{1}{5}} \leq \left(\frac{\|A^\top A\|_2^2 R^2}{\min\{\lambda_1, \lambda_2\}}\right)^{\frac{1}{5}}.$$

Informally speaking, applying statistical preconditioning allows reducing the minimization of a large sum F of N functions in (2) to the minimization of a moderate sum ϕ of n functions when making the step 10 of Algorithm 2. To conclude this subsection we would like to discuss the complexity of minimizing function Ψ in (34) which is equivalent to step 10 of Algorithm 2. To that end, we consider the setting of sparse logistic regression with loss (41). Since ϕ and Ψ have finite-sum form, a straightforward approach is to apply accelerated variance reduced methods. This leads to arithmetic operations complexity

$$\widetilde{O}\left(s \cdot \left(n + \sqrt{n\kappa}\right)\right), \tag{42}$$

where s comes from the cost of evaluating a sparse stochastic gradient $\nabla \ell(x;\zeta_i)$ for some random i, and the rest is the optimal bound on the number of stochastic gradient evaluations for such methods [49]. Note that we have $\kappa = L_\Psi/\mu_\Psi = L_\phi/\mu_\phi$.

We propose an alternative approach by applying Hyperfast second-order methods to minimize the function Ψ. Since basic Hyperfast second-order methods are a special implementation of third-order method [22,23,50,24–26], each their iteration requires to minimize the regularized third-order Taylor polynomial:

$$\min_{y\in\mathbb{R}^d}\left\{\langle \nabla\Psi(x), y-x\rangle + \frac{1}{2}\nabla^2\Psi(x)[y-x]^2 + \frac{1}{6}\nabla^3\Psi(x)\,[y-x]^3 + \frac{L_{\Psi,3}}{8}\|y-x\|_2^4\right\}. \quad (43)$$

It is shown in [22] that the objective in (43) is relatively smooth and strongly convex with respect to the function $a(y) = \frac{1}{2}\nabla^2\Psi(x)[y-x]^2 + \frac{L_{\Psi,3}}{8}\|y-x\|_2^4$ with $\mu_{\Psi/a} = 1-1/\sqrt{2}$, $L_{\Psi/a} = 1+1/\sqrt{2}$. Since $\kappa_{\Psi/a}$ is a constant, the complexity of solving (43) is, up to logarithmic factors, the same as for minimizing $a(y)$. In turn, the complexity of solving this problem, up to logarithmic factors, is the same as the complexity of a quadratic programming problem and can be estimated by the complexity of matrix inversion [51]. To sum up, the arithmetic operations complexity of minimizing the function Ψ by the Restarted Hyperfast second-order method has the form

$$\widetilde{O}\left(\left(s^2 n + d^{\log_2 7}\right)\cdot\left(\frac{L_{\phi,3}R^2}{\mu_\phi}\right)^{1/5}\right), \quad (44)$$

see [23,25,26] for more details on arithmetic complexity of each iteration of the basic Hyperfast method. The first term in (44), i.e., $s^2 n$, is due to the complexity of Hessian calculation. The second term, i.e. $d^{\log_2 7}$, corresponds to the complexity of Hessian inversion, e.g., by the matrix inversion lemma using Strassen's algorithm [28]. The term $\left(\frac{L_{\phi,3}R^2}{\mu_\phi}\right)^{1/5}$ comes from the estimate for the number of iterations of the basic Hyperfast second-order method, see Theorem 3.6. Additionally, we may expect $R^2 = O(d)$, since $\dim x_* = d$ and $L_{\phi,3} = O\big(\frac{1}{n}\sum_{i=1}^n \|\eta_i\xi_i\|_2^4\big) = O(s^2)$ since we consider sparse logistic regression.

Without loss of generality, we can assume that the parameter n can be set such that $d^{\log_2 7} = O\left(s^2 n\right)$. In this case, the Hyperfast second-order method with complexity (44) outperforms accelerated variance reduced algorithms with complexity (42) if $\mu_\phi \lesssim s^{-3}n^{-2}$. Where $\lesssim$, and $\simeq$ mean the same as $\leq$ and $=$, but up to dimension-dependent factors of the order $O(1)$. For the particular case of sparse logistic regression problems, our focused application, we can assume that $s = \widetilde{O}(1)$. Therefore, we have that if $d \lesssim n^{0.356}$ and $\mu_\phi \lesssim n^{-2}$, or, equivalently, if $d^{\log_2 7} \lesssim n \lesssim \mu_\phi^{-1/2}$, then, the Hyperfast second-order method has smaller arithmetic operations complexity than variance reduced algorithms. The last inequality is reasonable when the requirement for the accuracy is high. Indeed, in practice, via regularization [52], it is reasonable to set $\mu_\phi \simeq \mu_F \simeq \varepsilon/R \simeq \varepsilon/d$, where $\varepsilon > 0$ is a desired accuracy. Thus, in this case we can rewrite the

Table 1
Statistics of the datasets. N is the number of samples, d is the number of features, Feat. is the average number of dense features, and Size is the data size in MB.

Dataset	N	d	Feat.	Size
RCV1	20k	47k	74.05	13.7
In-house	710M	3,246k	109.86	650.8k

last inequality as $\varepsilon \lesssim n^{-1.644}$ $(d^{2.81} \lesssim n \lesssim \varepsilon^{-0.61})$. We can conclude that Hyperfast second-order methods are better when our goal is to solve sparse logistic regression with loss (41) with high accuracy. This result can be strengthened by using parallelization. In the complexity bound (42) for variance reduced algorithms, only the first term can be improved by applying parallelization on n nodes. On the contrary, in the bound (44) for Restarted Hyperfast method, the first term can be improved by parallelization on n nodes, and the second can be improved by parallelization on d nodes.

To conclude, high-order methods are competitive from the theoretical point of view for large-scale convex problems that require high accuracy of the solution, especially when the problem is sparse. Further improvements can potentially be achieved by using inexact tensor methods [51,53–55] to save some computational work.

4. Numerical analysis and implementation details

In this section, we present numerical experiments and implementation details of Algorithm 2. Namely, on the example of regularized logistic regression, we demonstrate the practical performance of InSPAG method with Hyperfast subsolver (InSPAG+Hyperfast) and compare it with the state-of-the-art methods such as DANE, DANE-HB and SPAG with SDCA subsolver. For the logistic regression, we show that InSPAG+Hyperfast outperforms other methods even for huge-dimensional problems with 710M samples and 3.2M features.

We work with binary classification problems with regularized logistic regression cost function (41) on a public dataset from LibSVM1,[4] namely RCV1 [56], and a proprietary large-scale in-house dataset that was generated from the click logs of a large-scale commercial system for mobile app install ads. The main statistics of the datasets are shown in Table 1.

We obtained an MPI-based distributed implementation of SPAG from the authors of [1] and modified it to run on an Apache Spark [57] cluster. As shown in Algorithm 2, InSPAG switches between two phases: a parallel gradient computation phase and a central-node optimization phase in which we run the Hyperfast second-order method in Algorithm 3. In our implementation, the driver carries the central-node optimization phase while executors compute the gradient. The code for the implementations was de-

[4] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html.

veloped in PyTorch [58]. Algorithm 3, in each iteration of the basic Hyperfast method, requires a line-search where to calculate a test point the full step (43) is made. The number of such line-search steps is theoretically bounded above by $O(\log(\varepsilon^{-1}))$. However, we observe that the line-search ends in approximately 5 trials in practice. Therefore, we bound the number of iterations executed in the line-search procedure. Additionally, our experiments show that the number of steps required in the line-search procedure decreases as more iterations of Algorithm 2 is executed. In the execution of the third-order step (43) it is sufficient to approximate the product of the third derivative with two vectors. To do this, we use off-the-shelf automatic differentiation codes and observe that the resulting computational complexity is equivalent approximately to $4-6$ gradient computations.

As explained in [25, Sect. 5.2], or [59, Algorithm 2], the problem (43) is solved by Bregman proximal gradient method under relative smoothness and strong convexity assumption [30]. Each step of this algorithm applied to (43) requires to solve the problem

$$\min_{s\in\mathbb{R}^d}\left\{\langle c,s\rangle+\frac{1}{2}\langle\nabla^2\Psi(x)s,s\rangle+\frac{L}{4}\|s\|_2^4\right\}, \tag{45}$$

where the vector c involves $\nabla\Psi(x)$ and $\nabla^3\Psi(x)[s]^2$, L is some regularization parameter. We solve problem (45) using ADAM [60] since then the gradient $c+\nabla^2\Psi(x)s+L\|s\|_2^2 s$ of the objective uses the Hessian only through Hessian-vector products which can be calculated using automatic differentiation. We observed that in practice this takes approximately $2-3$ times the time required for gradient computation. Thus, on the lowest level, our method is a first-order method with a Hessian-vector product and a third-order derivative product with two vectors computed by automatic differentiation techniques. The full Hessians or full third-order derivatives are not computed but are used for the method to exploit the additional curvature of the objective and improve the practical convergence speed. Moreover, the central node uses GPU to accelerate the various Hessian-related matrix-vector operations in the algorithm. We believe our implementation[5] to be the first practical implementation of an algorithm from the family of Hyperfast or even a wider family of higher-order optimizers that can operate on data at the above dimensionality.

We compare Algorithm 2 with the inner solver being Algorithm 3 and Algorithm 2 with the inner solver being Stochastic Dual Coordinate Ascent (SDCA) [61] used in [1]. For the RCV1 dataset, we also compare the performance of Algorithm 2 versus DANE [62] with both SDCA and Hyperfast as the central-node solver. We used $n=10^4$ samples for preconditioning, $\lambda=10^{-5}$, $\sigma=2\times10^{-5}$, constant $L_{F/\phi}=0.01$, and a practical approximate 10^{-2} for R_ϕ^2. We set the precision of the auxiliary subproblem to 10^{-4}.

[5] https://github.com/OPTAMI/OPTAMI/.

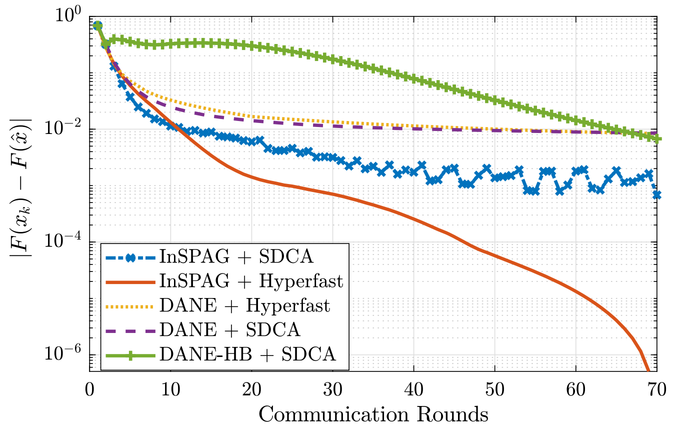

Fig. 1. Comparison of the communication rounds number for the dataset RCV1.

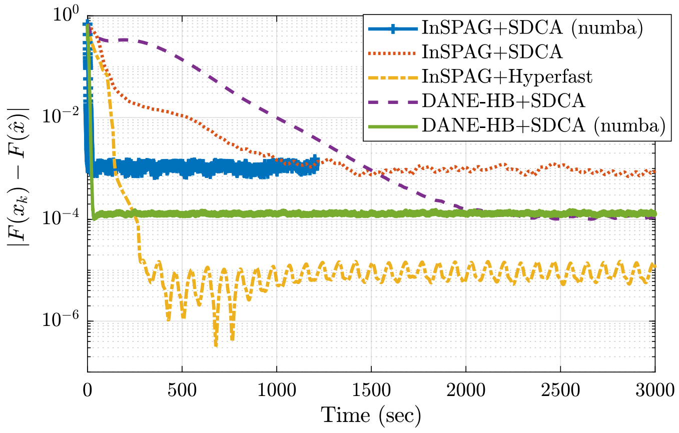

Fig. 2. Wall clock time performance of the InSPAG method for the dataset RCV1. "numba" indicates implementation using *Numba: A High Performance Python Compiler.*

Other parameters: $L_3 = 0.005$, the learning rate of ADAM is set to 1, and the number of iterations of ADAM is 2. Figs. 1 and 2 show results for the RCV1 dataset. The point $\hat{x}$ is set as the point where the minimal cost was achieved overall the iterations and runs of the algorithm and serves as a proxy point used instead of the minimizer, which is in general unknown. We see that Algorithm 2 outperforms DANE regardless of the subsolver used. Moreover, InSPAG-SDCA has better performance during initial iterations. However, InSPAG-Hyperfast outperforms all other methods by accuracy. Also, we find that Hyperfast iterations are faster than SDCA near the minimum point. For example, the first five iterations take about 20 seconds each, and the last five take about 1.5 seconds each. Hence, suggesting that some combination of methods would be used in practice. However, the Hyperfast approach finds better solutions overall.

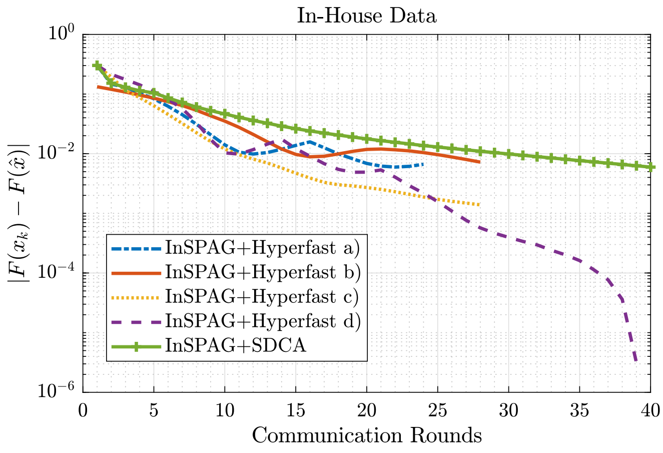

Fig. 3. Comparison of the communication rounds number for the in house dataset. a) $L_3 = 10$, ADAM learning rate 0.01, $n = 10000$; b) $L_3 = 100$, ADAM learning rate 0.1, $n = 10000$; c) $L_3 = 10$, ADAM learning rate 0.1, $n = 10000$; d) $L_3 = 15$, ADAM learning rate 0.01, $n = 1000$.

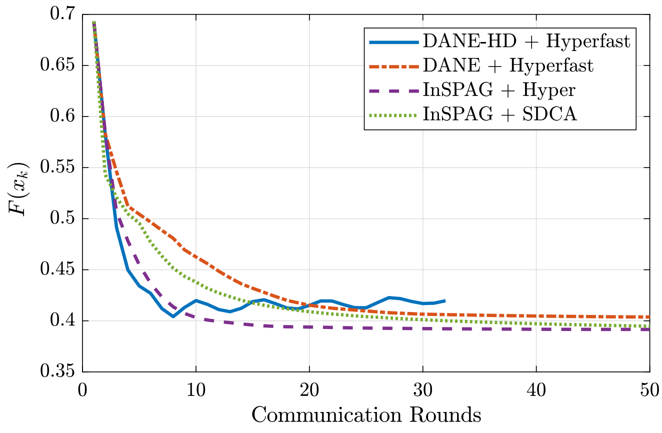

Fig. 4. Comparison of the communication rounds number for the in house dataset for different methods.

Table 2
Parameter selection for experiments on in-house data.

Run	L_3	ADAM	n	μ
a)	10	0.01	1×10^4	2×10^{-5}
b)	100	0.1	1×10^4	2×10^{-5}
c)	10	0.1	1×10^4	2×10^{-5}
d)	15	0.01	1×10^3	2×10^{-5}

Figs. 3, 4 show the results of the comparison on the in-house dataset (split over 200 nodes, i.e., $m = 200$) with $\lambda = 1 \times 10^{-7}$, $\sigma = 2 \times 10^{-5}$. Other parameters are described in Table 2. We see that InSPAG-Hyperfast outperforms InSPAG-SDCA for this large-scale dataset.

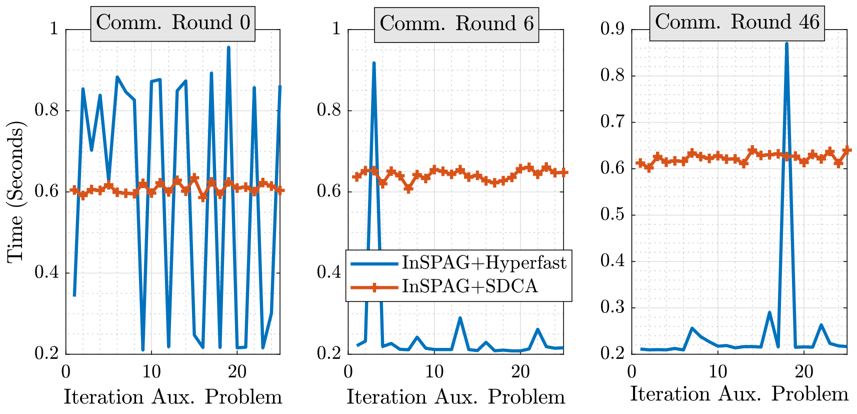

Fig. 5. The time complexity per iteration for the Hyperfast method in Algorithm 3 and the SDCA Method from [61] at communication rounds 0, 6, and 46. The x-axis is the iteration number, and the y-axis is the time required by the corresponding algorithm to complete its inner iteration.

Fig. 5 shows the times required by the Hyperfast method in Algorithm 3 and the SDCA Method from [61] to complete their inner iterations at communication rounds 0, 6, and 46. The x-axis is the iteration number, and the y-axis is the time required by the corresponding algorithm to complete an inner iteration. We can observe that in the communication round 0, the cost time required by both methods is approximately the same on average. However, for communication rounds 6 and 46, the Hyperfast method outperforms SDCA, requiring less time to complete an iteration.

Fig. 6 on the left shows the loss function $F(x_k)$ evaluated at the point x_k generated by iteration k as a function of the wall clock time recorded by the InSPAG method in Algorithm 2. Markers identify when an iteration has been completed. In this case we used the Hyperfast method in Algorithm 3 as the inner solver. Moreover, we show the dependency on the number n of points used for preconditioning. We observe that for different values of n, the final loss is about the same. However, as n increases, the wall clock time required increases as well. On the other hand, the right figure shows the loss function $F(x_k)$ evaluated at the point x_k generated by iteration k as a function of the number of communication rounds. As expected, when the number of data points used for preconditioning increases, the number of required communication rounds decreases. However, this implies that the central node needs to solve a bigger problem at every iteration and it takes longer to solve it.

Fig. 7 shows the wall clock time required by the central node to solve the auxiliary problem for every communication round. The x-axis shows the number of communication rounds, and the y-axis shows the clock time in seconds. Additionally, we show the results for different values of the preconditioning parameter n. As n increases, the time required for the solution of the auxiliary problem increases as well. However, the time complexity of the auxiliary subproblem decreases as the number of communication rounds increases.

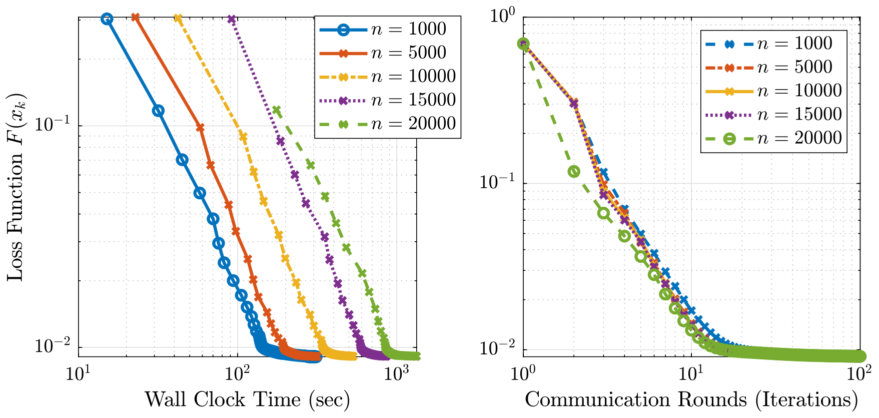

Fig. 6. A comparison of the wall clock times and communication rounds for the InSPAG method in Algorithm 2 for different number of data points used for preconditioning. On the left, the x-axis indicates time in seconds, and on the right the x-axis indicates number of communication rounds. In both cases the y-axis is the loss function at the current iteration.

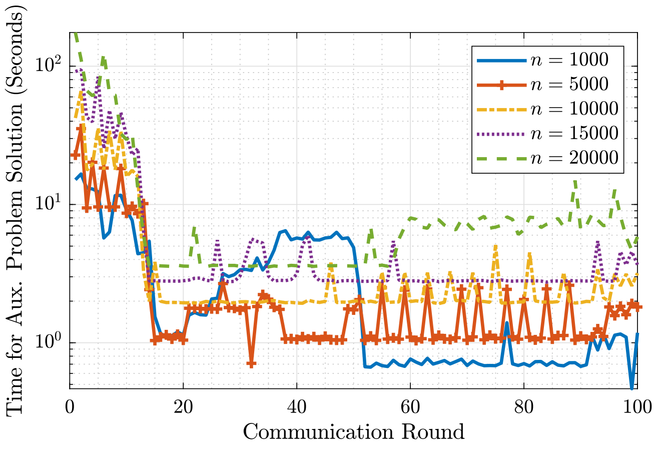

Fig. 7. Time complexity for the solution of the auxiliary subproblem for different number of preconditioning data points. The x-axis shows the number of communication rounds, and the y-axis shows the clock time in seconds.

5. Hyperfast second-order method for uniformly convex functions

For the sake of completeness, in this section we consider general problem $x_* = \arg\min_{x\in Q} f(x)$, where Q is closed convex bounded set, f has L_3-Lipschitz third-order derivative. We also assume that the objective $f(x)$ is uniformly convex of degree $4 \geq q \geq 2$ on the convex bounded set Q, i.e., there exists $\sigma_q > 0$ s.t.

$$f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle + \frac{\sigma_q}{q}\|y - x\|_2^q, \quad \forall x, y \in Q. \tag{46}$$

As a corollary,

$$f(y) \geq f(x_*) + \frac{\sigma_q}{q}\|y - x_*\|_2^q, \forall y \in Q. \tag{47}$$

Theorem 5.1 *([25][Theorem 2]). Let sequence x^k, $k \geq 0$ be generated by Hyperfast Second-Order Method [25][Eq.3.6] for $p = 3$ and $\beta = 1/2$ and with auxiliary steps described in [25][Sect. 5.2]. Then*

$$f(x_k) - f^* \leq \frac{3 \cdot 4^3 L_3 R_0^4}{1-\beta}\left[1 + \frac{2(k-1)}{4}\right]^{-5} \leq \frac{3 \cdot 4^4 L_3 R_0^4}{16k^5} = \frac{\hat{c} L_3 R_0^4}{k^5},$$

where R_0 is such that $\|x_0 - x^*\|_2 \leq R_0$, $\hat{c} = 48$.

We show how the restart technique can be used to accelerate Hyperfast second-order method under additional assumption of uniform convexity.

Algorithm 4 Restarted hyperfast second-order method.

```
Require: $q$, $\sigma_q$, $z_0, \Delta_0$ s.t. $f(z^0) - f^* \leq \Delta_0$.
1: for $k = 0, 1, \ldots$ do
2:     Set $\Delta_k = \Delta_0 \cdot 2^{-k}$ and $N_k = \max\left\{\left\lceil\left(\frac{2\hat{c}L_3 q^{\frac{4}{q}}}{\sigma_q^{\frac{4}{q}}}\Delta_k^{\frac{4-q}{q}}\right)^{\frac{1}{5}}\right\rceil, 1\right\}$.
3:     Set $z_{k+1} = y_{N_k}$ as the output of the basic Hyperfast method started from $z_k$ and run for $N_k$ steps.
4:     Set $k = k + 1$.
5: end for
Ensure: $z_k$.
```

Theorem 5.2. *Let sequence z^k, $k \geq 0$ be generated by Algorithm 4. Then*

$$\frac{\sigma_q}{q}\|z_k - x_*\|_2^q \leq f(z_k) - f^* \leq \Delta_0 \cdot 2^{-k},$$

and the total number of steps of the basic Hyperfast method is bounded by (c is the constant in Theorem 1.)

$$\left(2\hat{c}q^{\frac{4}{q}}\right)^{\frac{1}{5}} \frac{L_3^{\frac{1}{5}}}{\sigma_q^{\frac{4}{5q}}}(\Delta_0)^{\frac{4-q}{5q}} \cdot \sum_{i=0}^{k} 2^{-i\frac{4-q}{5q}} + k.$$

Proof. Let us prove the first statement of the Theorem by induction. For $k = 0$ it holds. If it holds for some $k \geq 0$, by choice of N_k, we have that

$$\frac{\hat{c}L_3}{N_k^5}\left(\frac{q\Delta_k}{\sigma_q}\right)^{\frac{4}{q}} \leq \frac{\Delta_k}{2}.$$

By (47),

$$\|z_k - x_*\|_2^4 \leq \left(\frac{q(f(z_k) - f^*)}{\sigma_q}\right)^{\frac{4}{q}} \leq \left(\frac{q\Delta_k}{\sigma_q}\right)^{\frac{4}{q}}$$

since, by our assumption, $q \le 4$. Combining the above two inequalities and Theorem 5.1, we obtain

$$f(z_{k+1}) - f^* \le \frac{\hat{c} L_3 \|z_k - x_*\|_2^4}{N_k^5} \le \frac{\Delta_k}{2} = \Delta_{k+1}.$$

It remains to bound the total number of steps of the basic Hyperfast method. Denote $\tilde{c} = \left(2\hat{c} q^{\frac{4}{q}}\right)^{\frac{1}{5}}$.

$$\sum_{i=0}^{k} N_i \le \tilde{c} \frac{L_3^{\frac{1}{5}}}{\sigma_q^{\frac{4}{5q}}} \sum_{i=0}^{k} (\Delta_0 \cdot 2^{-i})^{\frac{4-q}{5q}} + k \le \tilde{c} \frac{L_3^{\frac{1}{5}}}{\sigma_q^{\frac{4}{5q}}} (\Delta_0)^{\frac{4-q}{5q}} \cdot \sum_{i=0}^{k} 2^{-i\frac{4-q}{5q}} + k. \quad \square$$

Let us make a remark on the complexity of the restarted scheme in different settings. It is easy to see from Theorem 5.2 that, to achieve an accuracy ε, i.e., to find a point $\hat{x}$ s.t. $f(\hat{x}) - f^* \le \varepsilon$, the number of tensor steps in Algorithm 4 is

$$O\left(\frac{L_3^{\frac{1}{5}}}{\sigma_q^{\frac{4}{5q}}} (\Delta_0)^{\frac{4-q}{5q}} + \log_2 \frac{\Delta_0}{\varepsilon}\right), q < 4, \text{and } O\left(\left(\left(\frac{L_3}{\sigma_4}\right)^{\frac{1}{5}} + 1\right) \log_2 \frac{\Delta_0}{\varepsilon}\right), q = 4.$$

6. Conclusions

We study the distributed optimization problem of minimizing empirical risk with smooth and (strongly) convex losses and i.i.d. data stored at nodes. Building upon the recent result on statistical preconditioning, we propose an algorithm that iteratively minimizes the objective function taking advantage of the statistical similarity of the cost functions across the nodes. Such statistical preconditioning requires solving an auxiliary optimization problem at a designated central node. Contrary to existing approaches, we analyze the case where this auxiliary problem is solved inexactly. Moreover, we provide the conditions on the accuracy of the solution that guarantees convergence at the same rate as the algorithm with access to exact minimizers of the auxiliary problem. Additionally, we extend recently proposed Hyperfast second-order methods to the class of uniformly convex functions with bounded fourth-order derivatives. We show that the auxiliary problem in the statistically preconditioned distributed algorithm can be solved efficiently at a linear rate via this Hyperfast second-order method. We analyze the complexity of the proposed combination of the inexact statistically preconditioned algorithm with the Hyperfast second-order sub-solver and show that it converges linearly with the improved condition number. Finally, we show the first empirical results on implementing high-order methods on large-scale problems, where the dimension is of the order of 3 million, and the number of samples is 700 million. As a future research direction we indicate the application of the proposed algorithm to the regularized Wasserstein barycenter problem, which can be expressed as the minimization of large sum of higher-order smooth softmax functions [44].

Funding

The work by D. Kamzolov was supported by a grant for research centers in the field of artificial intelligence, provided by the Analytical Center for the Government of the Russian Federation in accordance with the subsidy agreement (agreement identifier 000000D730321P5Q0002) and the agreement with the Moscow Institute of Physics and Technology dated November 1, 2021 No. 70-2021-00138. The work by C. Uribe was supported by the Yahoo! Faculty Engagement Program and by the National Science Foundation under Grants No. 2211815 and No. 2213568. The work by P. Dvurechensky was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – The Berlin Mathematics Research Center MATH+ (EXC-2046/1, project ID: 390685689).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors are grateful to Hadrien Hendrikx for sharing the code of the SPAG algorithm http://proceedings.mlr.press/v119/hendrikx20a.html.

References

[1] H. Hendrikx, L. Xiao, S. Bubeck, F. Bach, L. Massoulie, Statistically preconditioned accelerated gradient method for distributed optimization, in: Proceedings of the 37th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, PMLR, vol. 119, 2020, pp. 4203–4227.

[2] O. Shamir, N. Srebro, T. Zhang, Communication-efficient distributed optimization using an approximate Newton-type method, in: Proceedings of the 31st International Conference on Machine Learning, in: Proceedings of Machine Learning Research, PMLR, Beijing, China, vol. 32, 2014, pp. 1000–1008.

[3] X.-T. Yuan, P. Li, On convergence of distributed approximate Newton methods: globalization, sharper bounds and beyond, J. Mach. Learn. Res. 21 (206) (2020) 1–51.

[4] S. Wang, F. Roosta, P. Xu, M.W. Mahoney, Giant: globally improved approximate Newton method for distributed optimization, in: Advances in Neural Information Processing Systems, 2018, pp. 2332–2342.

[5] H. Hendrikx, F. Bach, L. Massoulié, An optimal algorithm for decentralized finite-sum optimization, SIAM J. Control Optim. 31 (4) (2021) 2753–2783, https://doi.org/10.1137/20M134842X.

[6] T. Yang, Trading computation for communication: distributed stochastic dual coordinate ascent, in: Advances in Neural Information Processing Systems, 2013, pp. 629–637.

[7] M. Li, D.G. Andersen, J.W. Park, A.J. Smola, A. Ahmed, V. Josifovski, J. Long, E.J. Shekita, B.-Y. Su, Scaling distributed machine learning with the parameter server, in: 11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14), 2014, pp. 583–598.

[8] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.

[9] G. Lan, S. Lee, Y. Zhou, Communication-efficient algorithms for decentralized and stochastic optimization, Math. Program. (2018) 1–48.

[10] Y. Nesterov, et al., Lectures on Convex Optimization, vol. 137, Springer, 2018.
[11] S.J. Reddi, J. Konečnỳ, P. Richtárik, B. Póczós, A. Smola, Aide: fast and communication efficient distributed optimization, arXiv preprint, arXiv:1608.06879.
[12] Y. Zhang, X. Lin, Disco: distributed optimization for self-concordant empirical loss, in: Proceedings of the 32nd International Conference on Machine Learning, in: Proceedings of Machine Learning Research, PMLR, Lille, France, vol. 37, 2015, pp. 362–370.
[13] H. Lin, J. Mairal, Z. Harchaoui, A universal catalyst for first-order optimization, in: Proceedings of the 28th International Conference on Neural Information Processing Systems, vol. 2, NIPS'15, MIT Press, Cambridge, MA, USA, 2015, pp. 3384–3392.
[14] R.-A. Dragomir, A. Taylor, A. d'Aspremont, J. Bolte, Optimal complexity and certification of Bregman first-order methods, Math. Program. 194 (1) (2022) 41–83, https://doi.org/10.1007/s10107-021-01618-1.
[15] Y. Arjevani, O. Shamir, Communication complexity of distributed convex learning and optimization, in: Advances in Neural Information Processing Systems, vol. 28, Curran Associates, Inc., 2015, pp. 1756–1764.
[16] Y. Sun, G. Scutari, A. Daneshmand, Distributed optimization based on gradient tracking revisited: enhancing convergence rate via surrogation, SIAM J. Control Optim. 32 (2) (2022) 354–385, https://doi.org/10.1137/19M1259973.
[17] B. Bullins, Highly smooth minimization of non-smooth problems, in: Proceedings of Thirty Third Conference on Learning Theory, in: Proceedings of Machine Learning Research, PMLR, vol. 125, 2020, pp. 988–1030.
[18] E.G. Birgin, J.L. Gardenghi, J.M. Martínez, S.A. Santos, P.L. Toint, Worst-case evaluation complexity for unconstrained nonlinear optimization using high-order regularized models, Math. Program. 163 (1) (2017) 359–368, https://doi.org/10.1007/s10107-016-1065-8.
[19] Y. Carmon, J.C. Duchi, O. Hinder, A. Sidford, Lower bounds for finding stationary points I, Math. Program. 184 (1) (2020) 71–120.
[20] C. Cartis, N.I. Gould, P.L. Toint, Universal regularization methods: varying the power, the smoothness and the accuracy, SIAM J. Control Optim. 29 (1) (2019) 595–615.
[21] M. Baes, Estimate Sequence Methods: Extensions and Approximations, Institute for Operations Research, ETH, Zürich, Switzerland, 2009.
[22] Y. Nesterov, Implementable tensor methods in unconstrained convex optimization, Math. Program. (2019) 1–27.
[23] A. Gasnikov, P. Dvurechensky, E. Gorbunov, E. Vorontsova, D. Selikhanovych, C.A. Uribe, B. Jiang, H. Wang, S. Zhang, S. Bubeck, Q. Jiang, Y.T. Lee, Y. Li, A. Sidford, Near optimal methods for minimizing convex functions with Lipschitz p-th derivatives, in: Proceedings of the Thirty-Second Conference on Learning Theory, in: Proceedings of Machine Learning Research, PMLR, Phoenix, USA, vol. 99, 2019, pp. 1392–1393.
[24] Y. Nesterov, Superfast second-order methods for unconstrained convex optimization, J. Optim. Theory Appl. 1 (2021) 1–30, https://doi.org/10.1007/s10957-021-01930-y.
[25] Y. Nesterov, Inexact high-order proximal-point methods with auxiliary search procedure, SIAM J. Control Optim. 31 (4) (2021) 2807–2828, https://doi.org/10.1137/20M134705X.
[26] D. Kamzolov, A. Gasnikov, Near-optimal hyperfast second-order method for convex optimization and its sliding, arXiv preprint, arXiv:2002.09050.
[27] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, MIT Press, 2009.
[28] J. Huang, T.M. Smith, G.M. Henry, R.A. van de Geijn, Strassen's algorithm reloaded, in: SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2016, pp. 690–701.
[29] H.H. Bauschke, J. Bolte, M. Teboulle, A descent lemma beyond Lipschitz gradient continuity: first-order methods revisited and applications, Math. Oper. Res. 42 (2) (2016) 330–348.
[30] H. Lu, R.M. Freund, Y. Nesterov, Relatively smooth convex optimization by first-order methods, and applications, SIAM J. Control Optim. 28 (1) (2018) 333–354.
[31] F. Stonyakin, A. Tyurin, A. Gasnikov, P. Dvurechensky, A. Agafonov, D. Dvinskikh, M. Alkousa, D. Pasechnyuk, S. Artamonov, V. Piskunova, Inexact model: a framework for optimization and variational inequalities, Optim. Methods Softw. 36 (6) (2021) 1155–1201.
[32] A. Ben-Tal, A. Nemirovski, Lectures on Modern Convex Optimization (Lecture Notes), Personal web-page of A. Nemirovski, https://www2.isye.gatech.edu/~nemirovs/LMCOLN2021WithSol.pdf, 2020.
[33] O. Devolder, F. Glineur, Y. Nesterov, First-order methods of smooth convex optimization with inexact oracle, Math. Program. 146 (1) (2014) 37–75, https://doi.org/10.1007/s10107-013-0677-5.

[34] P. Dvurechensky, A. Gasnikov, Stochastic intermediate gradient method for convex problems with stochastic inexact oracle, J. Optim. Theory Appl. 171 (1) (2016) 121–145.
[35] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM J. Imaging Sci. 2 (1) (2009) 183–202, https://doi.org/10.1137/080716542.
[36] Y. Nesterov, Gradient methods for minimizing composite functions, Math. Program. 140 (1) (2013) 125–161, first appeared in 2007 as CORE discussion paper 2007/76.
[37] F. Hanzely, P. Richtárik, L. Xiao, Accelerated Bregman proximal gradient methods for relatively smooth convex optimization, Comput. Optim. Appl. 79 (2) (2021) 405–440, https://doi.org/10.1007/s10589-021-00273-8.
[38] M.I. Florea, Exact gradient methods with memory, Optim. Methods Softw. (2022) 1–28, https://doi.org/10.1080/10556788.2022.2091559.
[39] H.H. Bauschke, J. Bolte, M. Teboulle, A descent lemma beyond Lipschitz gradient continuity: first-order methods revisited and applications, Math. Oper. Res. 42 (2) (2017) 330–348.
[40] K. Scaman, F. Bach, S. Bubeck, Y.T. Lee, L. Massoulié, Optimal algorithms for smooth and strongly convex distributed optimization in networks, in: Proceedings of the 34th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, PMLR, vol. 70, 2017, pp. 3027–3036.
[41] A.V. Gasnikov, Y.E. Nesterov, Universal method for stochastic composite optimization problems, Comput. Math. Math. Phys. 58 (1) (2018) 48–64.
[42] Y. Nesterov, Lectures on Convex Optimization, 2nd edition, Springer Optimization and Its Applications, vol. 137, Springer International Publishing, 2018.
[43] P. Dvurechensky, A. Gasnikov, A. Kroshnin, Computational optimal transport: complexity by accelerated gradient descent is better than by Sinkhorn's algorithm, in: Proceedings of the 35th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 80, 2018, pp. 1367–1376.
[44] P. Dvurechensky, D. Dvinskikh, A. Gasnikov, C.A. Uribe, A. Nedić, Decentralize and randomize: faster algorithm for Wasserstein barycenters, in: Advances in Neural Information Processing Systems, vol. 31, NIPS'18, Curran Associates, Inc., 2018, pp. 10783–10793.
[45] P. Dvurechensky, S. Shtern, M. Staudigl, First-order methods for convex optimization, EURO J. Comput. Optim. 9 (2021) 100015, https://doi.org/10.1016/j.ejco.2021.100015.
[46] Q. Lin, L. Xiao, An adaptive accelerated proximal gradient method and its homotopy continuation for sparse optimization, in: Proceedings of the 31st International Conference on Machine Learning, in: Proceedings of Machine Learning Research, PMLR, Bejing, China, vol. 32, 2014, pp. 73–81.
[47] R.D. Monteiro, B.F. Svaiter, An accelerated hybrid proximal extragradient method for convex optimization and its implications to second-order methods, SIAM J. Control Optim. 23 (2) (2013) 1092–1125.
[48] Y. Nesterov, Smooth minimization of non-smooth functions, Math. Program. 103 (1) (2005) 127–152.
[49] G. Lan, First-Order and Stochastic Optimization Methods for Machine Learning, Springer, 2020.
[50] N. Doikov, Y. Nesterov, Contracting proximal methods for smooth convex optimization, SIAM J. Control Optim. 30 (4) (2020) 3146–3169, https://doi.org/10.1137/19M130769X.
[51] Y. Nesterov, Inexact basic tensor methods for some classes of convex optimization problems, Optim. Methods Softw. (2020) 1–29.
[52] A. Gasnikov, Universal gradient descent, arXiv preprint, arXiv:1711.00394.
[53] N. Doikov, Y. Nesterov, Inexact tensor methods with dynamic accuracies, in: Proceedings of the 37th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, PMLR, vol. 119, 2020, pp. 2577–2586.
[54] A. Agafonov, D. Kamzolov, P. Dvurechensky, A. Gasnikov, Inexact tensor methods and their application to stochastic convex optimization, arXiv:2012.15636.
[55] D. Kamzolov, A. Gasnikov, P. Dvurechensky, Optimal combination of tensor optimization methods, in: Optimization and Applications, Springer International Publishing, Cham, 2020, pp. 166–183.
[56] D.D. Lewis, Y. Yang, T.G. Rose, F. Li, Rcv1: a new benchmark collection for text categorization research, J. Mach. Learn. Res. 5 (Apr 2004) 361–397.
[57] Apache, Spark 2.4.5, https://spark.apache.org/, 2020.
[58] Pytorch, 1.5.0, https://pytorch.org/, 2020.
[59] D. Kamzolov, Near-optimal hyperfast second-order method for convex optimization, in: Mathematical Optimization Theory and Operations Research, Springer International Publishing, Cham, 2020, pp. 167–178.
[60] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint, arXiv:1412.6980.

[61] S. Shalev-Shwartz, Sdca without duality, regularization, and individual convexity, in: Proceedings of the 33rd International Conference on Machine Learning, in: Proceedings of Machine Learning Research, PMLR, New York, New York, USA, vol. 48, 2016, pp. 747–754.

[62] O. Shamir, N. Srebro, T. Zhang, Communication-efficient distributed optimization using an approximate Newton-type method, in: International Conference on Machine Learning, 2014, pp. 1000–1008.