AN ASYNCHRONOUS BUNDLE METHOD FOR DISTRIBUTED LEARNING PROBLEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a novel asynchronous bundle method to solve distributed learning problems. Compared to existing asynchronous methods, our algorithm computes the next iterate based on a more accurate approximation of the objective function and does not require any prior information about the maximal information delay in the system. This makes the proposed method fast and easy to tune. We prove that the algorithm converges in both deterministic and stochastic (mini-batch) settings, and quantify how the convergence times depend on the level of asynchrony. The practical advantages of our method are illustrated through numerical experiments on classification problems of varying complexities and scales.

1 INTRODUCTION

We consider a setting where data is distributed among n workers, each with its own smooth convex loss function $f_i : \mathbf{R}^d \to \mathbf{R}$. Our goal is to compute a solution $x^\star$ of

$$\text{minimize } F(x) \triangleq f(x) + R(x), \tag{1}$$

where $f(x) \triangleq \sum_{i=1}^n f_i(x)$ and $R : \mathbf{R}^d \to \mathbf{R}$ is a (possibly non-smooth) proper closed and convex regularizer. This problem template is ubiquitous in machine learning and includes lasso (Tibshirani, 1996), logistic regression (Koh et al., 2007) and many other important problem classes.

When data is distributed among multiple workers, algorithms that require the workers to synchronize in every iteration are bottlenecked by the slowest worker. *Asynchronous* algorithms (Bertsekas & Tsitsiklis, 1989; Assran et al., 2020) mitigate this issue by imposing less restrictive synchronization requirements, potentially resulting in machine learning systems that are both faster and easier to implement than their synchronous counterparts (Hannah & Yin, 2017). However, the design of asynchronous algorithms is challenging, since information such as function values and gradients computed by workers may be obsolete when received by the coordinating mechanism like a central server. Consequently, existing convergence results for asynchronous algorithms often depend on a typically large and unknown upper bound on the information delay from the workers. Furthermore, the admissible stepsizes often decay rapidly as the upper delay bound increases. In practice this complicates the implementation of an asynchronous algorithm: if the estimated upper bound on the delays is too small, then the bound may not be valid. On the other hand, an excessively large upper bound will lead to small step sizes and slow practical convergence.

The design of most optimization algorithms is based on a *simple* approximation of the objective function (in the optimization literature, this is sometimes called a *model* of the objective). Examples include the quadratic upper bound of an L-smooth convex function that is used in the design of the gradient descent algorithm, or the piecewise linear lower bound on a convex function with a known optimal value that is the basis of the Polyak step-size (Polyak, 1964). However, a string of recent papers on *synchronous* algorithms (Davis & Drusvyatskiy, 2019; Asi & Duchi, 2019; Nesterov & Florea, 2021) suggests that it can be beneficial to compute the next iterate based on a more accurate approximation of the objective. This raises the question of whether *asynchronous* algorithms could also be improved by more accurate approximations of the objective, potentially resulting in asynchronous algorithms with faster practical convergence and simpler tuning.

Contributions. We propose a method for parallel and asynchronous optimization that uses a more accurate approximation of the objective function to compute the next iterate. The method is suitable for a parameter server architecture (Li et al., 2013) and decouples gradient evaluations at the

workers from decision vector updates at the coordinating master, and thereby becomes robust to the amount of asynchrony in the system. On the theoretical side, we prove that our algorithm converges for all bounded delays and can be implemented without knowledge of the maximum delay (we are only aware of two other asynchronous algorithms (Mishchenko et al., 2018; Wu et al., 2022) that 1) require no upper bound on the delays, 2) explicitly take the regularizer $R(x)$ into account, and 3) are designed for a parameter server). On the practical side, we present an algorithm that converges quickly with essentially no tuning. Our method supports stochastic function and gradient evaluations, and can be seen as an asynchronous bundle method, generalizing the algorithms in (Nesterov & Florea, 2021; Asi & Duchi, 2019) to an asynchronous setting.

Outline. The paper is structured as follows. In §2 we relate our contribution to existing work. In §3, we introduce a new asynchronous model-based algorithm, followed by a convergence analysis in §4. We discuss implementation details and present numerical experiments in §5 and §6, respectively. Finally, in §7 we summarize and conclude our work.

2 Related work

Model-based optimization. Model-based optimization is a general framework in which an approximation, or *model*, of the objective function is maintained and used to compute the next iterate. This framework encompasses several well-known algorithms and principles, including the expectation-maximization algorithm (Neal & Hinton, 1998), quasi-Newton methods (Dennis & Moré, 1977), bundle methods (Mäkelä, 2002), the majorization-minimization principle (Mairal, 2015; Lange, 2016), and acceleration (d'Aspremont et al., 2021). In the context of stochastic optimization, recent work has demonstrated the benefits of using more accurate approximations of the objective function to compute the next iterate, yielding algorithms that are faster and more robust to step size selection (Duchi & Ruan, 2018; Davis & Drusvyatskiy, 2019; Asi & Duchi, 2019). Furthermore, for composite non-stochastic optimization, Nesterov & Florea (2021) recently demonstrated that building up a piecewise linear model of the smooth part of the objective, instead of only using the most recent gradient to approximate the objective, can lead to improved performance.

Our work aims to extend the idea of more accurate objective function models to asynchronous optimization.

Parallel and asynchronous optimization. For parallel optimization with a parameter server architecture, asynchronous algorithms can be much faster than their synchronous counterparts (Hannah & Yin, 2017). Asynchronous methods have also demonstrated promising results for other architectures (Recht et al., 2011; Chaturapruek et al., 2015). The stepsizes in many asynchronous methods are either diminishing (Duchi et al., 2015; Assran & Rabbat, 2020) or rely on a predetermined maximum iteration number (Koloskova et al., 2022; Mishchenko et al., 2022; Recht et al., 2011). Exceptions that are well-suited for a parameter server can be broadly categorized into two groups: methods that require the knowledge of an upper bound on the information delays (Zhang & Kwok, 2014; Peng et al., 2016; Gürbüzbalaban et al., 2017; Vanli et al., 2018; Wai et al., 2020; Sun et al., 2019), and algorithms that do not rely on such a bound (Feyzmahdavian et al., 2014; Mishchenko et al., 2018; Wu et al., 2022; 2023). In practice, however, an upper bound on the information delay is often unknown a priori, and since the admissible stepsizes are reduced as the upper delay bound increases, it is often difficult to guarantee that the algorithms in the first group converge in practice.

Most of the asynchronous methods mentioned above employ cheap closed-form updates at the central server, while the bulk of the computational work, such as gradient and function evaluations, is offloaded to the workers. The reason for the cheap update at the central server is, in the terminology of model-based optimization, that the central server maintains a simple model of the objective function based only on the *most recent* information from each worker. Our method is different in the sense that we propose to use a more accurate model of the objective function that incorporates more than just the latest information from each worker. The fact that the computationally intensive tasks are handled by the workers suggests an opportunity to investigate algorithms that are slightly more complex at the server, like the one we propose, to potentially improve the overall system performance.

Bundle methods. A key challenge in asynchronous optimization is that gradients provide *local descent directions*, making it challenging to combine gradients from different workers computed

at different iterates into a meaningful search direction that ensures descent. In contrast, gradients (together with function values and convexity) provide *global lower bounds* on the objective function, making it easier to combine gradients evaluated at widely different points into a valid lower bound of the objective function. This observation motivates our method, which can be seen as an *asynchronous bundle method*.

In the *non-smooth* optimization literature, several versions of asynchronous bundle methods have been proposed (Emiel & Sagastizábal, 2010; Iutzeler et al., 2020; van Ackooij & Frangioni, 2018; de Oliveira & Eckstein, 2015). Non-smooth optimization encompasses a wide range of problems, thus leading to quite weak convergence results for the existing asynchronous bundle methods. (A typical convergence result is that cluster points of the sequence of iterates solve the problem; see, for example, (Emiel & Sagastizábal, 2010, Proposition 4) or (de Oliveira & Eckstein, 2015, Theorem 3.6).) Our setting in this paper is different. While the aforementioned works assume that the non-smoothness of the objective is present in the finite-sum structure, we assume that the finite-sum structure arises in a smooth part of the objective, and the non-smoothness of the objective is caused by a regularizer. This key distinction allows us to derive much stronger convergence guarantees under a quadratic functional growth assumption: not only do we prove convergence, but we also show that our method converges linearly (see Theorem 4.5 for a precise statement).

Bundle methods have also received direct attention from the machine learning community (Teo et al., 2007; Franc & Sönnenburg, 2009; Teo et al., 2010; Chu et al., 2017; Paren et al., 2022). The first three works design variants of bundle methods for general empirical risk minimization, but they differ from our method in several ways. For example, these methods maintain a piecewise linear model of the sum $f = \sum_{i=1}^n f_i$, whereas our algorithm maintains separate piecewise linear models for each f_i. Furthermore, unlike our method, these methods require the workers to synchronize in every iteration.

Bundle methods have also been successfully applied to non-convex problems (see, for example, Hare & Sagastizábal (2010)). In particular, the special case of the Polyak step-size, where the bundle only consists of the current cut and a lower bound on the objective, has proven to yield strong performance in deep neural network training (Loizou et al., 2021; Wang et al., 2023a).

3 Algorithm

In this section we present a model-based algorithm for solving (1) asynchronously. We use a parameter server architecture (Li et al., 2013) with one central server and n workers. The central server maintains a copy of the global decision variable and can query each worker for its function value and gradient. Based on this information, the central server builds up a piecewise linear model of each worker's loss function, and then uses this model to compute the next iterate. A notable feature of the algorithm is that its implementation does not require knowledge of an upper bound on the information delay.

3.1 Main idea

To simplify the presentation of the method we consider a fixed iteration and drop the iteration index. We assume that the iteration number is sufficiently large to ensure that the central server has received information from each worker in at least m previous iterates. The parameter m is referred to as the *bundle size*. For $i \in \{1, \dots, n\}$ we introduce an algorithmic parameter $M_i > 0$ which, roughly speaking, represents the smoothness parameter of worker i, and we let $M \triangleq \sum_{i=1}^n M_i$. (An exact definition of M_i is given in §4.) We will later show that in a practical implementation of our method, the parameters $M_i,\ i = 1, \dots, n$ are estimated adaptively and require no tuning.

Let z_j^i for $j = 1, \dots, m$ denote the m previous iterates in which the central server has received information from worker i. We label the iterates so that z_m^i is the *most recent* iterate for which the central server has received information from worker i, and let $\bar{z} \triangleq \frac{1}{M} \sum_{i=1}^n M_i z_m^i$ denote a weighted average of these points. The central server maintains the following piecewise linear model of f_i:

$$\check{f}_i(x) = \max_{1 \le j \le m} \left\{ f_i(z_j^i) + \langle \nabla f_i(z_j^i), x - z_j^i \rangle \right\}. \tag{2}$$

When the central server receives information from one or several workers, it replaces the oldest iterate in the bundle for those workers. The next iterate is then computed as an *approximate* solution of

$$\min_{x\in\mathbf{R}^d}\left\{\sum_{i=1}^n \check{f}_i(x) + R(x) + \frac{M}{2}\|x-\bar{z}\|_2^2\right\}. \tag{3}$$

(We will later specify what we mean by an *approximate* solution.) Note in particular that the bundle center in (3) is chosen as a weighted average of the most recent iterates for the workers; this is essential for the convergence analysis in §4.

In our method, the central server must store m gradients of size d for all n workers, resulting in a total memory complexity of order $\mathcal{O}(mnd)$. This is more than the $\mathcal{O}(nd)$ memory required by methods that only store the most recent gradient of each worker, but often substantially less than the $\mathcal{O}(d^2)$ requirement of methods such as (Soori et al., 2020) that store an approximation of the Hessian at the central server.

3.2 Solving the master problem

In every iteration the central server must solve the *master problem* (3), which for common regularizers such as $R(x) = \lambda\|x\|_1$ can be formulated as a quadratic program with linear inequality constraints. If the dimension d is large, solving (3) can become a computational bottleneck. However, when an aggregated piecewise linear model of the objective function is used and $R(x) = 0$, it is well known that the dual of (3) is a low-dimensional quadratic program over the probability simplex (see, for example, (Hiriart-Urruty & Lemarechal, 1993, p. 296)). As the following lemma shows, a similar observation can be made when a disaggregated piecewise linear model of f is used and when $R(x) \neq 0$. To state the lemma we define matrices $\boldsymbol{G}_i \in \mathbf{R}^{d\times m}$ containing old gradient information of f_i by

$$\boldsymbol{G}_i = \begin{bmatrix}\nabla f_i(z_1^i) & \dots & \nabla f_i(z_m^i)\end{bmatrix}.$$

For $i = 1,\dots,n$, let $v_i \in \mathbf{R}^m$ be defined componentwise by $(v_i)_j = \langle \nabla f_i(z_j^i), z_j^i\rangle - f_i(z_j^i)$ and let $v = (v_1,\dots,v_n) \in \mathbf{R}^{mn}$. Recall that for $\gamma > 0$, the *Moreau envelope* of R and the *proximal operator* of R are defined by

$$H_R^\gamma(y) = \min_x\left\{R(x) + \frac{1}{2\gamma}\|x-y\|_2^2\right\}, \quad \mathbf{prox}_{\gamma R}(y) = \arg\min_x\left\{R(x) + \frac{1}{2\gamma}\|x-y\|_2^2\right\}.$$

Lemma 3.1. *Let $\lambda_i \in \mathbf{R}^m,\ i = 1,\dots,n$ and $\lambda = (\lambda_1,\dots,\lambda_n) \in \mathbf{R}^{mn}$. Define $g : \mathbf{R}^{mn} \to \mathbf{R}$ by*

$$g(\lambda) = \frac{M}{2}\|\bar{z} - \frac{1}{M}\sum_{i=1}^n \boldsymbol{G}_i\lambda_i\|_2^2 - H_R^{1/M}\left(\bar{z} - \frac{1}{M}\sum_{i=1}^n \boldsymbol{G}_i\lambda_i\right) + \langle v,\lambda\rangle.$$

The Lagrange dual of (3) *is given by*

$$\begin{array}{ll} \textit{minimize} & g(\lambda) \\ \textit{subject to} & \mathbf{1}^T\lambda_i = 1,\ \lambda_i \geq 0,\ i = 1,\dots,n. \end{array} \tag{4}$$

Furthermore, if $\lambda^\star$ is optimal in (4)*, then the unique solution of* (3)*, denoted by x_{exact}, is given by*

$$x_{exact} = \mathbf{prox}_{\frac{1}{M}R}\big(\bar{z} - \frac{1}{M}\sum_{i=1}^n \boldsymbol{G}_i\lambda_i^\star\big). \tag{5}$$

Proof. The proofs of this and all forthcoming results are given in the appendix. □

3.3 An efficient approximate master problem solver

According to Lemma 3.1, we can solve the master problem (3) by solving its low-dimensional dual (4). However, even if the dual is low-dimensional, it can be too expensive to solve it to high accuracy since the second term in the definition of the dual objective function itself involves a minimization problem in x. Therefore, in our algorithm, we only generate approximate solutions to (3) using inexact solutions to (4). The goal of this subsection is to introduce equation (6) below, which defines a termination criterion that we use to specify what we mean by an inexact solution.

Algorithm 1

Setup: x_0, parameters M_i, bundle size m, tolerance $\delta > 0$
Initialization: the central server receives $f_i(x_0)$ and $\nabla f_i(x_0),\ i = 1, \dots, n$
while not interrupted by central server: each worker i **do**
  receive x from the server, compute $f_i(x)$ and $\nabla f_i(x)$, and send them back to the server
end while
while not converged: central server **do**
  for $i = 1, \dots, n$ **do**
    if received information from worker i **then**
      update the bundle of worker i by throwing out the oldest information
    end if
  end for
  compute $\bar{\lambda}$ satisfying (6) and then update x according to (7)
  send back x to all workers that the server received information from
end while

Denote the feasible set of (4) by $\mathbf{\Delta} \subseteq \mathbf{R}^{mn}$. The dual objective function g is differentiable since the Moreau envelope is differentiable. Hence, from optimality conditions for convex optimization (see, for example, (Nesterov, 2018, p. 177)), $\lambda^\star$ solves (4) if and only if

$$\langle \nabla g(\lambda^\star), \lambda^\star - \lambda \rangle \leq 0 \text{ for all } \lambda \in \mathbf{\Delta}.$$

As in (Nesterov & Florea, 2021), we allow for inexact solution of (4) by introducing a parameter $\delta > 0$ together with the requirement that we compute a point $\bar{\lambda}$ satisfying

$$\langle \nabla g(\bar{\lambda}), \bar{\lambda} - \lambda \rangle \leq \delta \text{ for all } \lambda \in \mathbf{\Delta}. \tag{6}$$

The next iterate, denoted by x_+, is then computed as (cf. (5))

$$x_+ = \mathbf{prox}_{\frac{1}{M}R}(\bar{z} - \frac{1}{M}\sum_{i=1}^{n} \boldsymbol{G}_i \bar{\lambda}_i). \tag{7}$$

A summary of the algorithm we propose is given in Algorithm 1. Implementation details, including how to find $\bar{\lambda}$ satisfying (6), are given in §5.

3.4 Extension to stochastic function values and gradients

While the main focus of this paper is the setting where exact (full batch) function and gradient evaluations are used, we will also analyze a variant that uses stochastic (mini-batch) function values and gradients. For this setting we assume that each worker $i \in \{1, \dots, n\}$ has access to an oracle that when queried at a point x, draws a random variable ξ from some distribution and outputs both a stochastic function value $F_i(x; \xi)$ approximating $f_i(x)$, and a stochastic gradient $G_i(x; \xi)$ approximating $\nabla f_i(x)$. In this stochastic setting, the central server replaces the piecewise linear model (2) of f_i with the following stochastic piecewise linear model:

$$\check{f}_i(x, \boldsymbol{\xi}) = \max_{1 \leq j \leq m} \left\{ F_i(z_j^i; \xi_j^i) + \langle G_i(z_j^i; \xi_j^i), x - z_j^i \rangle \right\}.$$

Here ξ_j^i is the random variable from the query of the oracle of worker i in the point z_j^i, and the bold $\boldsymbol{\xi}$ represents all randomness used to construct the current bundle.

4 Convergence analysis

In this section we study the convergence of Algorithm 1 and its stochastic extension. Our main result is Theorem 4.5 which shows that under standard assumptions, the algorithm converges linearly to a neighborhood of $x^\star$ whose size depends on the accuracy used to solve the master problem. Theorem 4.5 also characterizes how the information delays affect the convergence rate. The convergence analysis uses two sequences of points: the sequence of iterates x_k for $k \in \mathbf{Z}_+ \triangleq \{0, 1, 2, \dots\}$ and the sequence of points $z_{k,j}^i$ for $k \in \mathbf{Z}_+, i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$ used to construct the

piecewise linear model of f_i in iteration k. (These points are previous iterates, *i.e.,* for $(k, i, j) \in \mathbf{Z}_+ \times \{1, \ldots, n\} \times \{1, \ldots, m\}$ there exists a non-negative integer $s^i_{k,j} \leq k$ such that $z^i_{k,j} = x_{s^i_{k,j}}$.)

We will assume that the points are labeled such that $z^i_{k,m}$ denotes the *most recent iterate* in which the central server has received information from worker i in iteration k. With this convention, the quantity $k - s^i_{k,m} \geq 0$ is called the *delay of worker i in iteration k*. Our first assumption is standard (see, for example, Feyzmahdavian et al. (2014); Gürbüzbalaban et al. (2017); Vanli et al. (2018); Mishchenko et al. (2018)) and states that the maximum delay is bounded by an integer $\tau \geq 0$.

Assumption 4.1. In every iteration k the delay of worker i is bounded by τ. In other words, for $(k, i) \in \mathbf{Z}_+ \times \{1, \ldots, n\}$ it holds that $k - s^i_{k,m} \leq \tau$.

We will also make the following standard assumption on the objective function.

Assumption 4.2. The loss function of worker $i \in \{1, \ldots, n\}$ is smooth with parameter L_i.

Occasionally we will further make the following common growth assumption that is similar to, but weaker, than strong convexity (see, for example, Necoara et al. (2019)).

Assumption 4.3. The full objective function F has a quadratic functional growth with parameter $\mu > 0$, meaning that $F(x) - F(x^\star) \geq (\mu/2)\|x - x^\star\|_2^2$ for all $x \in \mathbf{R}^d$.

4.1 Analysis for exact function values and gradients

We now present a convergence analysis when exact (non-stochastic) function values and gradients are used. First we need an additional assumption.

Assumption 4.4. The loss function f_i of worker $i \in \{1, \ldots, n\}$ is star-convex, meaning that $f_i(x^\star) \geq f_i(x) + \langle \nabla f_i(x), x^\star - x \rangle$ for all $x \in \mathbf{R}^d$.

Under the growth assumption we can show linear convergence to a neighborhood of the solution.

Theorem 4.5. *Under Assumptions 4.1, 4.2, 4.3 and 4.4 the iterates of Algorithm 1 using $M_i = L_i$, $i = 1, \ldots, n$ satisfy*

$$\|x_k - x^\star\|_2^2 \leq \rho^k \|x_0 - x^\star\|_2^2 + \epsilon_\delta, \tag{8}$$

where $\rho = (L/(L + \mu))^{1/(1+\tau)}$ and $\epsilon_\delta = 2\delta/\mu$.

Remark 4.6. A notable feature of our algorithm is that neither its implementation nor its tuning requires any information about the level of asynchrony in the system. The method converges with default parameters as long as the information delay from all workers are finite, and under Assumption 4.3, the convergence rate decreases as the level of asynchrony in the system increases.

Remark 4.7. Rather than relying on the common yet unrealistic assumption of solving the subproblem (3) exactly, our analysis explicitly accounts for and characterizes the impact of inexact subproblem solutions. As a result, an error term depending on δ naturally appears in the convergence result. However, we should point out that in practice, the algorithm we propose has no issues with finding highly accurate solutions (see §6).

For the analysis without the growth assumption we will use the following new sequence result that might be of independent interest.

Lemma 4.8. *Suppose that $(V_k)_{k=0}^\infty$ and $(W_k)_{k=0}^\infty$ are non-negative sequences satisfying*

$$V_{k+1} \leq \max_{(k-\tau)_+ \leq \ell \leq k} V_\ell - W_{k+1} + r, \qquad k = 0, 1, 2,$$

for a non-negative constant r. Then, for any $k \geq 1$,

$$\min_{t \leq k} W_t \leq \frac{(\tau + 1)V_0}{k} + r.$$

Using Lemma 4.8 we can prove sublinear convergence in terms of the function value gap.

Theorem 4.9. *Under Assumptions 4.1, 4.2, and 4.4, the iterates of Algorithm 1 using $M_i = L_i$, $i = 1, \ldots, n$, satisfy that for any $k \geq 1$,*

$$\min_{t \leq k} F(x_t) - F(x^\star) \leq \frac{(\tau + 1)L\|x_0 - x^\star\|_2^2}{2k} + \delta.$$

4.2 Analysis for stochastic function values and gradients

When stochastic function values and gradients are used we will make the following assumptions.

Assumption 4.10. For each worker $i \in \{1, \ldots, n\}$:

1. The oracle is star-convex, *i.e.,* $F_i(x^\star; \xi) \geq F_i(x; \xi) + \langle G_i(x; \xi), x^\star - x \rangle$ for all $x \in \mathbf{R}^d$. Furthermore, $\mathbf{E}[F_i(x, \xi)] = f_i(x)$ and $\mathbf{E}[G_i(x, \xi)] = \nabla f_i(x)$ for all $x \in \mathbf{R}^d$.
2. The variance of the stochastic gradients is bounded by some finite constant $\sigma_2^2 > 0$, meaning that $\mathbf{E}[\|G_i(x, \xi) - \nabla f_i(x)\|_2^2] \leq \sigma_2^2$ for all $x \in \mathbf{R}^d$.
3. The function value noise at the optimal solution $x^\star$ is bounded by some finite constant $\sigma_1^2 > 0$, meaning that $\mathbf{E}[(F_i(x^\star; \xi) - f_i(x^\star))^2] \leq \sigma_1^2$.

Relation to previous assumptions in the literature. The second assumption bounding the noise of the gradients is common in the analysis of stochastic algorithms (see, for example, Koloskova et al. (2022); Mishchenko et al. (2022)). The third assumption is less common, since most algorithms often only use stochastic gradients and *not* function values. However, recent analysis of stochastic algorithms that use stochastic function values in addition to stochastic gradients make a similar assumption (see, for example, Loizou et al. (2021); Wang et al. (2023b)).

Theorem 4.11. *Consider Algorithm 1 with $M_i = \alpha L_i$, $i = 1, \ldots, n$ where $\alpha > 1$. Assume that stochastic function values and gradients are used. Under Assumptions 4.1, 4.2, and 4.10, the iterates of Algorithm 1 satisfy that for any $k \geq 1$,*

$$\min_{t \leq k} \mathbf{E}[F(x_t)] - F(x^\star) \leq \frac{\alpha(\tau + 1)L\|x - x_0\|_2^2}{2k} + \epsilon, \tag{9}$$

where $\epsilon = \epsilon_\delta + \epsilon_{\sigma_1} + \epsilon_{\sigma_2}$ with

$$\epsilon_\delta = \delta, \qquad \epsilon_{\sigma_1} = n\sigma_1\sqrt{m}, \qquad \epsilon_{\sigma_2} = \frac{\sigma_2^2}{2(\alpha - 1)} \cdot \sum_{i=1}^n \frac{1}{L_i}.$$

If, in addition, Assumption 4.3 holds, then

$$\mathbf{E}[\|x_k - x^\star\|_2^2] \leq \rho^k \|x_0 - x^\star\|_2^2 + 2\epsilon/\mu,$$

where $\rho = (\alpha L/(\alpha L + \mu))^{1/(1+\tau)}$.

Remark 4.12. Compared to algorithms with a simple explicit update rule of the form $x_{k+1} = \mathbf{prox}_{\gamma g}(x_k + \gamma d_k)$ where d_k is a direction and γ is a step size, the update mechanism of Algorithm 1 is more implicit since it involves solving the dual subproblem (4) approximately. This makes the analysis challenging. One of the main technical challenges in the proof in the stochastic setting is to carefully manage correlations between recently queried gradients $G_i(z_m^i)$, the dual variable $\bar{\lambda}$, and the next iterate x_{k+1}, all of which are correlated random variables.

Remark 4.13. The size of the neighborhood of the solution that x_k converges to in expectation depends on three terms: one term ϵ_δ which depends on the accuracy δ, and two other terms ϵ_{σ_1} and ϵ_{σ_2} which depend on the strength of the noise. The noise terms ϵ_{σ_1} and ϵ_{σ_2} depend on n, which may seem uncommon. This dependency arises because we analyze the canonical form $f(x) = \sum_{i=1}^n f_i(x)$ instead of the more common form $f(x) = (1/n)\sum_{i=1}^n f_i(x)$. Under the latter form, the noise terms would not depend on n.

5 Implementation

When implementing Algorithm 1, two issues must be addressed.

Solving the subproblem. First, we must find an approximate solution to (4) by finding $\bar{\lambda} \in \mathbf{R}^{mn}$ satisfying (6). Since $\mathbf{\Delta}$ is a Cartesian product of simplices, it is cheap to verify condition (6) by noting that

$$\sup_{\lambda \in \mathbf{\Delta}} \langle \nabla g(\bar{\lambda}), \bar{\lambda} - \lambda \rangle = \langle \nabla g(\bar{\lambda}), \bar{\lambda} \rangle - \inf_{\lambda \in \mathbf{\Delta}} \langle \nabla g(\bar{\lambda}), \lambda \rangle = \langle \nabla g(\bar{\lambda}), \bar{\lambda} \rangle - \sum_{i=1}^n \min \nabla_i g(\bar{\lambda}),$$

where $\min \nabla_i g(\bar{\lambda})$ is the smallest element of $\nabla_i g(\bar{\lambda})$. (Here $\nabla_i g(\bar{\lambda})$ is the gradient of g with respect to λ_i.) To find $\bar{\lambda}$ we have implemented an accelerated projected gradient method for solving (4) (Beck, 2017, page 291). Each iteration requires the gradient of g and projecting onto the feasible set $\mathbf{\Delta}$. Since $\mathbf{\Delta}$ is a Cartesian product of low-dimensional simplices, the projection can be done efficiently (see, for example, (Condat, 2016)). Furthermore, from properties of the Moreau-envelope (Beck, 2017, p. 166) it follows that the gradient of g with respect to λ_i is

$$\nabla_i g(\lambda) = \boldsymbol{G}_i^T(u - \mathbf{prox}_{\frac{1}{M}R}(u)) - \boldsymbol{G}_i^T u + v_i = v_i - \boldsymbol{G}_i^T \mathbf{prox}_{\frac{1}{M}R}(u), \tag{10}$$

where $u \triangleq \bar{z} - \frac{1}{M}\sum_{i=1}^n \boldsymbol{G}_i \lambda_i$.

In the appendix we compare the cost of solving the subproblem using this specialized method versus a high-performance interior-point solver. The main conclusion is that this specialized approach is more than an order of magnitude faster and that the complexity for solving the subproblem is of order $\mathcal{O}(nmd)$.

Adaptive estimation of smoothness parameters. First-order methods for solving (1) typically require knowledge of smoothness parameters. These parameters are often unknown or expensive to compute in practice. To eliminate the need for choosing a suitable value on L_i in Algorithm 1 we propose to estimate it adaptively using similar ideas to (Malitsky & Mishchenko, 2020).

Recall that $\nabla f_i(z_m^i)$ is the *most recent* gradient that the central server has received from worker i, and let $\nabla f_i(z_{m-1}^i)$ denote the *next most recent* gradient that the central server has received from worker i. Given z_m^i, $\nabla f_i(z_m^i)$, z_{m-1}^i and $\nabla f_i(z_{m-1}^i)$, a natural estimate of the local smoothness of f_i is the quantity

$$\hat{L}_i = \frac{\|\nabla f_i(z_m^i) - \nabla f_i(z_{m-1}^i)\|_2}{\|z_m^i - z_{m-1}^i\|_2}.$$

Every time the central server receives a new gradient from worker i, we propose to update the smoothness parameter L_i using this estimate.

6 Experiments

We consider binary and multiclass classification problems based on a logistic model. For the binary classification, we use the objective function

$$f(x) = \frac{1}{N}\sum_{j=1}^N \left(\log(1 + e^{-y_j(a_j^T x)}) + \frac{\lambda_2}{2}\|x\|_2^2\right)$$

and the regularizer $R(x) = \lambda_1\|x\|_1$, where $a_1, \ldots, a_N \in \mathbf{R}^p$ are the feature vectors and $y_1, \ldots, y_N \in \{-1, 1\}$ are the corresponding labels. Due to the space limitations, we defer the results for multiclass classification to the appendix.

We conduct experiments on three datasets (`mnist8m`/`infimnist`, `epsilon`, `rcv1`) from the LIBSVM library (Chang & Lin, 2011) and on the `SVHN` dataset (Netzer et al., 2011). We pick $\lambda_2 = 1/N$, and tune λ_1 for each dataset to obtain a classifier $x^\star$ with 10-20% non-zero entries. The dataset `mnist8m` corresponds to a multiclass problem with 10 different labels. To use it for binary classification, we select data corresponding to the digits 7 and 9 and discard the rest. Table 1 shows the dimensions of each problem and the value of λ_1.

All methods are evaluated on a workstation using 10 cores. One core is assigned the role as the central server and the remaining $n = 9$ cores are workers. The data is distributed evenly among the workers. The code is written in Python using MPI4PY (Dalcin & Fang, 2021) and will be made publicly available. To evaluate the objective value and the gradients we use PyTorch (Paszke et al., 2019) for the dense datasets, and sparse linear algebra for `rcv1`.

6.1 Benchmarking

To benchmark our asynchronous bundle method (`ABM`) we compare it with two asynchronous proximal gradient methods, namely `DAve-RPG` (Mishchenko et al., 2018) and `PIAG` with delay-tracking (Wu et al., 2022). We selected these methods since they operate under the same conditions as `ABM` in

Table 1: Properties of the datasets that we use. The total number of data points is denoted by N and d is the dimension of the decision variable. The column labeled DENSITY shows the percentage of non-zero entries. The label DENSE means that the data matrix is stored as a dense matrix.

DATASET		N	d	DENSITY	λ_1
MNIST8M	(BINARY)	164 8890	784	DENSE	$3e$-3
RCV1	(BINARY)	677 399	47236	0.15%	$3e$-6
EPSILON	(BINARY)	500 000	2000	DENSE	$5e$-5
SVHN	(MULTICLASS)	630 420	10240	DENSE	$1e$-3
MNIST8M	(MULTICLASS)	8100 000	7840	DENSE	$8e$-3

the sense that they use a parameter server, require no delay information in parameters, and explicitly incorporate the regularizer. Both DAve-RPG and PIAG use exact gradients so we also use exact function values and gradients for ABM.

For ABM we use bundle size $m = 10$, master problem tolerance $\delta = 10^{-7}$, and adaptive smoothness estimation. DAve-RPG has two hyperparameters: the step size γ and the number of inner prox-steps p. We use step size $\gamma = 1/L_{\text{average}}$ where L_{average} is the average smoothness parameter of the workers, and $p = 1$ inner prox-steps (as in (Mishchenko et al., 2018)). For PIAG with delay-tracking we implemented the first adaptive step size strategy described in (Wu et al., 2022).

The first row of Figure 1 shows the relative suboptimality $(f(x_k) - f^\star)/f^\star$ versus the runtime. We see that ABM clearly outperforms the other two methods. The second row shows the suboptimality versus the number of gradients received by the server. We see that ABM achieves much higher accuracy with fewer gradients received by the server. For rcv1 the difference is striking: in 300 seconds the central server in ABM receives about 4000 gradients, while the servers of DAve-RPG and PIAG receive more than 70000 gradients in the same amount of time. This is in contrast to mnist8m and epsilon where all servers receive roughly the same number of gradients in the same amount of time. The reason underlying this observation is that the gradients are cheap to compute for the sparse data set rcv1 and more expensive to compute for the dense datasets (cf. Table 1). Consequently, for rcv1, the time required to solve the subproblem at the central server is non-negligible compared to the time needed to evaluate gradients. In contrast, for the dense datasets, this computation time is almost negligible.

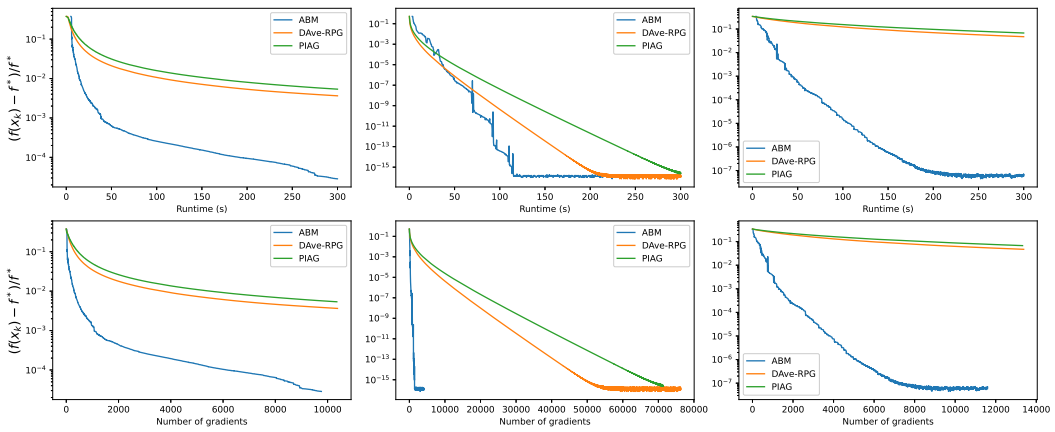


Figure 1: The progress of ABM, DAve-RPG and PIAG on the binary classification problems. The datasets are arranged in the order mnist8m, rcv1 and epsilon from the left.

6.2 SENSITIVITY TO HYPERPARAMETERS

Strictly speaking, ABM has two hyperparameters: the tolerance δ and the bundle size m. We will now investigate the sensitivity of the algorithm's performance to these parameters. The first row of Figure 2 shows the progress of ABM for fixed $\delta = 10^{-7}$ and bundle size $m \in \{2, 5, 10\}$. Increasing

the bundle size from $m = 2$ to $m \in \{5, 10\}$ results in much faster convergence for `mnist8m`. Furthermore, `ABM` makes *no* progress with bundle size $m = 2$ for `rcv1` and `epsilon`, but with bundle size $m \in \{5, 10\}$ the convergence is fast. This indicates the advantage of using a more accurate approximation of the objective function for computing the next iterate.

Next we run `ABM` with fixed bundle size $m = 10$ and tolerance $\delta \in \{10^{-5}, 10^{-7}, 10^{-9}\}$. The result is shown in the second row of Figure 2. We see that `ABM` has good performance for all three values on δ.

This experiment suggests that `ABM` essentially requires no tuning in these experiments: the values $m = 10$ and $\delta = 10^{-7}$ seem to work well. In particular, the performance of `ABM` is not as sensitive to its hyperparameters as, for example, stochastic gradient descent is to its step size.

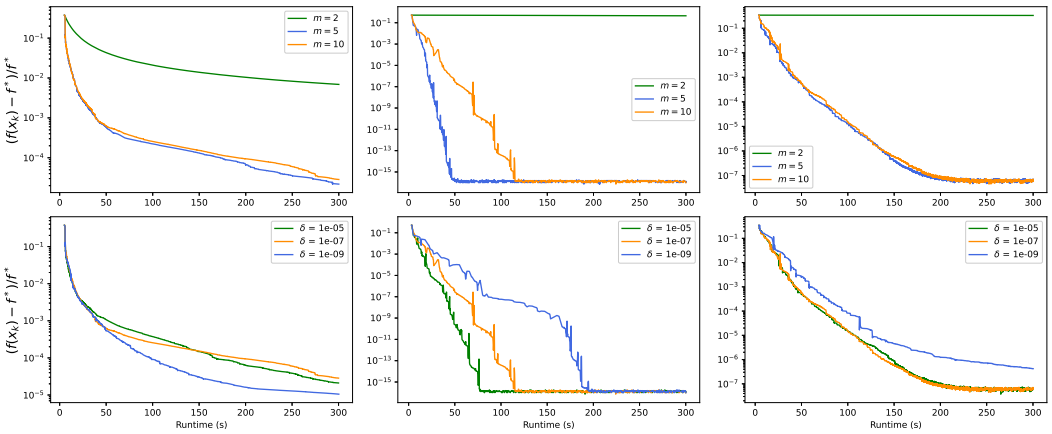


Figure 2: Sensitivity of `ABM`'s performance with respect to hyperparameters. The first row shows the performance for different bundle sizes m and fixed tolerance δ. The second row shows the performance for different tolerances δ and fixed bundle size. Each column corresponds to one dataset, and the columns are arranged in the order `mnist8m`, `rcv1`, `epsilon`.

6.3 Further experiments

In the appendix we conduct further experiments when $\lambda_2 = 0$, *i.e.,* when the objective function is convex but not strongly convex, and we also conduct experiments on multiclass classification. The main conclusion is that `ABM` outperforms `DAve-RPG` and `PIAG` with delay tracking (see Figure 3 and 4 in the appendix). We also test the stochastic version of `ABM` using mini-batches for multiclassification on the biggest data set `mnist8m`. The experiment indicates that mini-batching can speed up `ABM` if only a modest accuracy is required (see Figure 5 in the appendix).

7 Discussion

We have presented an asynchronous bundle method that is suitable for distributed learning problems. The algorithm constructs a piecewise linear model to approximate the local loss of each worker and uses this model to compute the next iterate. Compared to other first-order asynchronous algorithms, our proposed method employs a more refined model of the objective function. This allows it to converge quickly in practice with minimal tuning or specification of unknown constants.

Our method uses a fixed bundle size. An interesting extension would be to design a scheme that dynamically adjusts the bundle size based on the actual delays, potentially discarding outdated function value information that hasn't been used recently in the server subproblem. (The jth linear approximation of f_i is not used in the server subproblem if $\lambda^\star_{ij} = 0$, where $\lambda^\star$ is the solution of (4).) Another extension could be to let the central server maintain a low-rank approximation of the Hessian for each worker, enabling a better approximation of the curvature of the loss function, and possibly faster convergence. We leave these extensions for future work.

REFERENCES

Martin Skovgaard Andersen, Joachim Dahl, Zhang Liu, and Lieven Vandenberghe. *Interior-point methods for large-scale cone programming*. MIT Press, 2011. ISBN 9780262016469.

Hilal Asi and John C. Duchi. The importance of better models in stochastic optimization. *Proceedings of the National Academy of Sciences*, 116(46):22924–22930, 2019. doi: 10.1073/pnas.1908018116. URL https://www.pnas.org/doi/abs/10.1073/pnas.1908018116.

By Mahmoud Assran, Arda Aytekin, Hamid Reza Feyzmahdavian, Mikael Johansson, and Michael G. Rabbat. Advances in Asynchronous Parallel and Distributed Optimization. *Proceedings of the IEEE*, 108(11):2013–2031, 2020. doi: 10.1109/JPROC.2020.3026619.

Mahmoud S Assran and Michael G Rabbat. Asynchronous gradient push. *IEEE Transactions on Automatic Control*, 66(1):168–183, 2020.

Amir Beck. *First-Order Methods in Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017. doi: 10.1137/1.9781611974997. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611974997.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Inc., USA, 1989. ISBN 0136487009.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004. ISBN 0521833787.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.*, 2(3), may 2011. ISSN 2157-6904. doi: 10.1145/1961189.1961199. URL https://doi.org/10.1145/1961189.1961199.

Sorathan Chaturapruek, John C Duchi, and Christopher Ré. Asynchronous stochastic convex optimization: the noise is in the noise and SGD don't care. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/c8c41c4a18675a74e01c8a20e8a0f662-Paper.pdf.

Dejun Chu, Changshui Zhang, and Qing Tao. A Faster Cutting Plane Algorithm with Accelerated Line Search for Linear SVM. *Pattern Recogn.*, 67(C):127–138, jul 2017. ISSN 0031-3203. doi: 10.1016/j.patcog.2017.02.006. URL https://doi.org/10.1016/j.patcog.2017.02.006.

Laurent Condat. Fast projection onto the simplex and the l_1 ball. *Mathematical Programming*, 158(1):575–585, 07 2016. ISSN 1436-4646. doi: 10.1007/s10107-015-0946-6. URL https://doi.org/10.1007/s10107-015-0946-6.

Lisandro Dalcin and Yao-Lung L. Fang. mpi4py: Status Update After 12 Years of Development. *Computing in Science & Engineering*, 23(4):47–54, 2021. doi: 10.1109/MCSE.2021.3083216.

Damek Davis and Dmitriy Drusvyatskiy. Stochastic Model-Based Minimization of Weakly Convex Functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019. doi: 10.1137/18M1178244. URL https://doi.org/10.1137/18M1178244.

Welington de Oliveira and Jonathan Eckstein. Bundle method for exploiting additive structure in difficult optimization problems. *Optimization Online,*, 2015.

J. E. Dennis, Jr. and Jorge J. Moré. Quasi-Newton Methods, Motivation and Theory. *SIAM Review*, 19(1):46–89, 1977. doi: 10.1137/1019005. URL https://doi.org/10.1137/1019005.

John C. Duchi and Feng Ruan. Stochastic Methods for Composite and Weakly Convex Optimization Problems. *SIAM Journal on Optimization*, 28(4):3229–3259, 2018. doi: 10.1137/17M1135086. URL https://doi.org/10.1137/17M1135086.

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

Alexandre d'Aspremont, Damien Scieur, and Adrien Taylor. Acceleration methods. *Foundations and Trends® in Optimization*, 5(1-2):1–245, 2021. ISSN 2167-3888. doi: 10.1561/2400000036. URL http://dx.doi.org/10.1561/2400000036.

Grégory Emiel and Claudia A. Sagastizábal. Incremental-like bundle methods with application to energy planning. *Comput. Optim. Appl.*, 46(2):305–332, 2010.

Hamid Reza Feyzmahdavian, Arda Aytekin, and Mikael Johansson. A delayed proximal gradient method with linear convergence rate. In *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, 2014. doi: 10.1109/MLSP.2014.6958872.

V. Franc and S. Sönnenburg. Optimized Cutting Plane Algorithm for Large-Scale Risk Minimization. *Journal of Machine Learning Research*, 10(76):2157–2192, 2009. URL http://jmlr.org/papers/v10/franc09a.html.

Paul Goulart and Yuwen Chen. Clarabel: A library for optimization and control, 2021. URL https://oxfordcontrol.github.io/ClarabelDocs/stable/.

M. Gürbüzbalaban, A. Ozdaglar, and P. A. Parrilo. On the Convergence Rate of Incremental Aggregated Gradient Algorithms. *SIAM Journal on Optimization*, 27(2):1035–1048, 2017. doi: 10.1137/15M1049695. URL https://doi.org/10.1137/15M1049695.

Robert Hannah and Wotao Yin. More Iterations per Second, Same Quality – Why Asynchronous Algorithms may Drastically Outperform Traditional Ones, 2017. URL https://arxiv.org/abs/1708.05136.

Warren Hare and Claudia Sagastizábal. A redistributed proximal bundle method for nonconvex optimization. *SIAM Journal on Optimization*, 20(5):2442–2473, 2010. doi: 10.1137/090754595. URL https://doi.org/10.1137/090754595.

J.B. Hiriart-Urruty and C. Lemarechal. *Convex Analysis and Minimization Algorithms II: Advanced Theory and Bundle Methods*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 1993. ISBN 9783662064092. URL https://books.google.se/books?id=QinsCAAAQBAJ.

Franck Iutzeler, Jérôme Malick, and Welington de Oliveira. Asynchronous level bundle methods. *Mathematical Programming*, 184:319–348, November 2020. doi: 10.1007/s10107-019-01414-y. URL https://hal.archives-ouvertes.fr/hal-02182996.

Kwangmoo Koh, Seung-Jean Kim, and Stephen Boyd. An Interior-Point Method for Large-Scale l1-Regularized Logistic Regression. *Journal of Machine Learning Research*, 8(54):1519–1555, 2007. URL http://jmlr.org/papers/v8/koh07a.html.

Anastasiia Koloskova, Sebastian U Stich, and Martin Jaggi. Sharper convergence guarantees for asynchronous sgd for distributed and federated learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 17202–17215. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/6db3ea527f53682657b3d6b02a841340-Paper-Conference.pdf.

Kenneth Lange. *MM Optimization Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016. doi: 10.1137/1.9781611974409. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611974409.

Mu Li, Li Zhou, Zichao Yang, Aaron Li, Fei Xia, David G Andersen, and Alexander Smola. Parameter server for distributed machine learning. In *Big Learning NIPS Workshop*, volume 6, pp. 2, 2013.

Nicolas Loizou, Sharan Vaswani, Issam Hadj Laradji, and Simon Lacoste-Julien. Stochastic Polyak Step-size for SGD: An Adaptive Learning Rate for Fast Convergence. In Arindam Banerjee and Kenji Fukumizu (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 1306–1314. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/loizou21a.html.

Julien Mairal. Incremental Majorization-Minimization Optimization with Application to Large-Scale Machine Learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

Yura Malitsky and Konstantin Mishchenko. Adaptive Gradient Descent without Descent. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6702–6712. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/malitsky20a.html.

Konstantin Mishchenko, Franck Iutzeler, Jérôme Malick, and Massih-Reza Amini. A Delay-tolerant Proximal-Gradient Algorithm for Distributed Learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3587–3595. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/mishchenko18a.html.

Konstantin Mishchenko, Francis Bach, Mathieu Even, and Blake E Woodworth. Asynchronous sgd beats minibatch sgd under arbitrary delays. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 420–433. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/029df12a9363313c3e41047844ecad94-Paper-Conference.pdf.

Marko Mäkelä. Survey of Bundle Methods for Nonsmooth Optimization. *Optimization Methods and Software*, 17(1):1–29, 2002. doi: 10.1080/10556780290027828. URL https://doi.org/10.1080/10556780290027828.

Radford M. Neal and Geoffrey E. Hinton. *A View of the Em Algorithm that Justifies Incremental, Sparse, and other Variants*, pp. 355–368. Springer Netherlands, Dordrecht, 1998.

Ion Necoara, Yurii Nesterov, and François Glineur. Linear convergence of first order methods for non-strongly convex optimization. *Mathematical Programming*, 175(1):69–107, May 2019. ISSN 1436-4646. doi: 10.1007/s10107-018-1232-1. URL https://doi.org/10.1007/s10107-018-1232-1.

Y. Nesterov. *Lectures on Convex Optimization*. Springer Publishing Company, Incorporated, 2nd edition, 2018. ISBN 3319915770.

Yurii Nesterov and Mihai I. Florea. Gradient methods with memory. *Optimization Methods and Software*, 0(0):1–18, 2021. doi: 10.1080/10556788.2020.1858831. URL https://doi.org/10.1080/10556788.2020.1858831.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Alasdair Paren, Leonard Berrada, Rudra PK Poudel, and M Pawan Kumar. A Stochastic Bundle Method for Interpolating Networks. *Journal of Machine Learning Research*, 23:1–57, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Zhimin Peng, Yangyang Xu, Ming Yan, and Wotao Yin. ARock: An Algorithmic Framework for Asynchronous Parallel Coordinate Updates. *SIAM Journal on Scientific Computing*, 38(5): A2851–A2879, 2016. doi: 10.1137/15M1024950. URL https://doi.org/10.1137/15M1024950.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR computational mathematics and mathematical physics*, 4(5):1–17, 1964.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. *Advances in Neural Information Processing Systems*, 24:693–701, 2011.

Saeed Soori, Konstantin Mishchenko, Aryan Mokhtari, Maryam Mehri Dehnavi, and Mert Gurbuzbalaban. DAve-QN: A Distributed Averaged Quasi-Newton Method with Local Superlinear Convergence Rate. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 1965–1976. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/soori20a.html.

Tao Sun, Yuejiao Sun, Dongsheng Li, and Qing Liao. *General proximal incremental aggregated gradient algorithms: better and novel results under general scheme*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Choon Hui Teo, Alex Smola, S. V.N. Vishwanathan, and Quoc Viet Le. A Scalable Modular Convex Solver for Regularized Risk Minimization. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pp. 727–736, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936097. doi: 10.1145/1281192.1281270. URL https://doi.org/10.1145/1281192.1281270.

Choon Hui Teo, S.V.N. Vishwanthan, Alex J. Smola, and Quoc V. Le. Bundle methods for regularized risk minimization. *J. Mach. Learn. Res.*, 11:311–365, mar 2010. ISSN 1532-4435.

Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. ISSN 00359246. URL http://www.jstor.org/stable/2346178.

Wim van Ackooij and Antonio Frangioni. Incremental Bundle Methods using Upper Models. *SIAM Journal on Optimization*, 28(1):379–410, 2018. doi: 10.1137/16M1089897. URL https://doi.org/10.1137/16M1089897.

N. D. Vanli, M. Gürbüzbalaban, and A. Ozdaglar. Global Convergence Rate of Proximal Incremental Aggregated Gradient Methods. *SIAM Journal on Optimization*, 28(2):1282–1300, 2018. doi: 10.1137/16M1094415. URL https://doi.org/10.1137/16M1094415.

Hoi-To Wai, Wei Shi, César A. Uribe, Angelia Nedić, and Anna Scaglione. Accelerating incremental gradient optimization with curvature information. *Computational Optimization and Applications*, 76(2):347–380, June 2020. doi: 10.1007/s10589-020-00183-. URL https://ideas.repec.org/a/spr/coopap/v76y2020i2d10.1007_s10589-020-00183-1.html.

Xiaoyu Wang, Mikael Johansson, and Tong Zhang. Generalized polyak step size for first order optimization with momentum. In *International Conference on Machine Learning*, pp. 35836–35863. PMLR, 2023a.

Xiaoyu Wang, Mikael Johansson, and Tong Zhang. Generalized Polyak Step Size for First Order Optimization with Momentum. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023b.

Xuyang Wu, Sindri Magnusson, Hamid Reza Feyzmahdavian, and Mikael Johansson. Delay-Adaptive Step-sizes for Asynchronous Learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 24093–24113. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/wu22g.html.

Xuyang Wu, Changxin Liu, Sindri Magnússon, and Mikael Johansson. Delay-agnostic asynchronous coordinate update algorithm. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 37582–37606. PMLR, 23–29 Jul 2023.

Ruiliang Zhang and James Kwok. Asynchronous Distributed ADMM for Consensus Optimization. In Eric P. Xing and Tony Jebara (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1701–1709, Bejing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/zhange14.html.

A Appendix

In this appendix we present proofs and additional numerical experiments.

A.1 Dual subproblem

Lemma A.1. *Let $\lambda_i \in \mathbf{R}^m,\ i = 1, \ldots, n$ and $\lambda = (\lambda_1, \ldots, \lambda_n) \in \mathbf{R}^{mn}$. Define $g : \mathbf{R}^{mn} \to \mathbf{R}$ by*

$$g(\lambda) = \frac{M}{2}\|\bar{z} - \frac{1}{M}\sum_{i=1}^{n} \boldsymbol{G}_i\lambda_i\|_2^2 - H_R^{1/M}\left(\bar{z} - \frac{1}{M}\sum_{i=1}^{n}\boldsymbol{G}_i\lambda_i\right) + \langle v, \lambda\rangle.$$

The Lagrange dual of (3) *is given by*

$$\begin{array}{ll} \textit{minimize} & g(\lambda) \\ \textit{subject to} & \mathbf{1}^T\lambda_i = 1,\ \lambda_i \geq 0,\ i = 1, \ldots, n. \end{array} \tag{11}$$

Furthermore, if $\lambda^\star$ is optimal in (4)*, then the unique solution of* (3)*, denoted by x_{exact}, is given by*

$$x_{exact} = \mathbf{prox}_{\frac{1}{M}R}\big(\bar{z} - \frac{1}{M}\sum_{i=1}^{n}\boldsymbol{G}_i\lambda_i^\star\big). \tag{12}$$

Proof. In the proof we use the notation of stochastic function values and gradients. The setting with exact function evaluations can be recovered by doing the substitutions $F_i(x;\xi) = f_i(x)$ and $G_i(x;\xi) = \nabla f_i(x)$.

Problem (3) can be formulated as

$$\begin{array}{l} \text{minimize } \displaystyle\sum_{i=1}^{n} r_i + \frac{M}{2}\|x - \bar{z}\|_2^2 + R(x) \\ \text{subject to } r_i \geq F_i(z_j^i;\xi_j^i) + \langle G_i(z_j^i;\xi_j^i), x - z_j^i\rangle,\ j = 1, \ldots, m,\ i = 1, \ldots, n, \end{array} \tag{13}$$

with variables $r \in \mathbf{R}^n$ and $x \in \mathbf{R}^d$. For $i = 1, \ldots, n$, introduce a Lagrange multiplier vector $\lambda_i \in \mathbf{R}^m$. The Lagrangian is given by

$$\begin{aligned} \mathcal{L}(x, r, \lambda_1, \ldots, \lambda_n) &= \sum_{i=1}^{n} r_i + \frac{M}{2}\|x - \bar{z}\|_2^2 + \sum_{i=1}^{n}\sum_{j=1}^{m}\lambda_{ij}(F_i(z_j^i;\xi_j^i) + \langle G_i(z_j^i;\xi_j^i), x - z_j^i\rangle - r_i) + R(x) \\ &= \sum_{i=1}^{n}\left(1 - \mathbf{1}^T\lambda_i\right)r_i + \frac{M}{2}\|x - \bar{z}\|_2^2 + \sum_{i=1}^{n}\langle\lambda_i, \boldsymbol{G}_i^T x - v_i\rangle + R(x), \end{aligned}$$

where

$$\boldsymbol{G}_i = \begin{bmatrix} G_i(z_1^i;\xi_1^i) & \ldots & G_i(z_m^i;\xi_m^i) \end{bmatrix} \in \mathbf{R}^{d\times m}$$

and $v_i \in \mathbf{R}^m$ is defined componentwise by

$$(v_i)_j = \langle G_i(z_j^i;\xi_j^i), z_j^i\rangle - F_i(z_j^i;\xi_j^i).$$

The Lagrangian is unbounded in r unless $\mathbf{1}^T\lambda_i = 1,\ i = 1, \ldots, n$. Furthermore, for such λ minimizing the Lagrangian over x yields

$$\begin{aligned} &\inf_x \mathcal{L}(x, r, \lambda_1, \ldots, \lambda_n) = \inf_x\left\{R(x) + \frac{M}{2}\|x - \bar{z}\|_2^2 + \sum_{i=1}^{n}\langle\boldsymbol{G}_i\lambda_i, x\rangle\right\} - \sum_{i=1}^{n}\langle\lambda_i, v_i\rangle \\ &= \inf_x\left\{R(x) + \frac{M}{2}\|x - (\bar{z} - \frac{1}{M}\sum_{i=1}^{n}\boldsymbol{G}_i\lambda_i)\|_2^2\right\} - \frac{M}{2}\|\bar{z} - \frac{1}{M}\sum_{i=1}^{n}\boldsymbol{G}_i\lambda_i\|_2^2 + \frac{M}{2}\|\bar{z}\|_2^2 - \sum_{i=1}^{n}\langle\lambda_i, v_i\rangle \\ &= H_R^{1/M}\left(\bar{z} - \frac{1}{M}\sum_{i=1}^{n}\boldsymbol{G}_i\lambda_i\right) - \frac{M}{2}\|\bar{z} - \frac{1}{M}\sum_{i=1}^{n}\boldsymbol{G}_i\lambda_i\|_2^2 + \frac{M}{2}\|\bar{z}\|_2^2 - \sum_{i=1}^{n}\langle\lambda_i, v_i\rangle. \end{aligned}$$

By dropping the term $\frac{M}{2}\|\bar{z}\|_2^2$ it follows that a dual problem is given by

$$\begin{aligned}&\text{minimize} \;\; g(\lambda) := -H_R^{1/M}\left(\bar{z} - \frac{1}{M}\sum_{i=1}^n \boldsymbol{G}_i\lambda_i\right) + \frac{M}{2}\|\bar{z} - \frac{1}{M}\sum_{i=1}^n \boldsymbol{G}_i\lambda_i\|_2^2 + \sum_{i=1}^n \langle \lambda_i, v_i\rangle \\ &\text{subject to} \;\; \lambda \in \boldsymbol{\Delta}.\end{aligned}$$

Since Slater's constraint qualification (Boyd & Vandenberghe, 2004, page 226) is satisfied, strong duality holds and the point x_{exact} defined by (12) minimizes $\mathcal{L}(x, r, \lambda_1^\star, \ldots, \lambda_n^\star)$ over x (here the value of r is arbitrary since the Lagrangian is independent of r for any value of λ that is dual feasible). The function $x \mapsto \mathcal{L}(x, r, \lambda_1^\star, \ldots, \lambda_n^\star)$ has a unique minimizer, so it follows that x_{exact} given by (12) indeed solves (3). □

A.2 Convergence analysis

In the analysis below we analyze the progress Algorithm 1 makes in iteration k. To simplify the notation we drop the iteration index. In other words, the notation $z_j^i,\ j = 1, \ldots, m$ below refers to the points used to construct the piecewise linear model of f_i *in iteration* k.

In the first few results we will stick with the convention of using stochastic function values and gradients. The setting with exact function evaluations can be recovered by doing the substitutions $F_i(x;\xi) = f_i(x)$ and $G_i(x;\xi) = \nabla f_i(x)$.

We will analyze Algorithm 1 by using (7). This will make the analysis depend on the dual variable $\bar{\lambda}$. The following result (inspired by (Nesterov & Florea, 2021)) will be useful to partly remove the dependence on $\bar{\lambda}$ from the analysis.

Lemma A.2. *Assume $\bar{\lambda}$ satisfies* (6) *and let x_{k+1} be given by* (7)*. Then*

$$\sum_{i=1}^n \sum_{j=1}^m \bar{\lambda}_{ij}[F_i(z_j^i;\xi_j^i) + \langle G_i(z_j^i;\xi_j^i), x_{k+1} - z_j^i\rangle] \geq \sum_{i=1}^n \check{f}_i(x_{k+1};\boldsymbol{\xi}) - \delta.$$

Proof. Since (6) is satisfied we have $\langle \bar{\lambda}, -\nabla g(\bar{\lambda})\rangle \geq \sup_{\lambda\in\boldsymbol{\Delta}}\langle \lambda, -\nabla g(\bar{\lambda})\rangle - \delta$. Note that

$$\langle \bar{\lambda}, -\nabla g(\bar{\lambda})\rangle = -\sum_{i=1}^n \langle \bar{\lambda}_i, \nabla g_i(\bar{\lambda})\rangle = \sum_{i=1}^n \sum_{j=1}^m \bar{\lambda}_{ij}[F_i(z_j^i;\xi_j^i) + \langle G_i(z_j^i;\xi_j^i), x_{k+1} - z_j^i\rangle]$$

$$\sup_{\lambda\in\boldsymbol{\Delta}} \langle \lambda, -\nabla g(\bar{\lambda})\rangle = \sup_{\lambda\in\boldsymbol{\Delta}} \sum_{i=1}^n \sum_{j=1}^m \lambda_{ij}[F_i(z_j^i;\xi_j^i) + \langle G_i(z_j^i;\xi_j^i), x_{k+1} - z_j^i\rangle] = \sum_{i=1}^n \check{f}_i(x_{k+1},\boldsymbol{\xi}).$$

□

Lemma A.3. *The next iterate x_{k+1} satisfies*

$$\begin{aligned}\frac{1}{2}\|x_{k+1} - x^\star\|_2^2 - \frac{1}{2M}\sum_{i=1}^n M_i\|z_m^i - x^\star\|_2^2 \leq \frac{1}{M}(R(x^\star) - R(x_{k+1})) - \frac{1}{2M}\sum_{i=1}^n M_i\|x_{k+1} - z_m^i\|_2^2 \\ + \frac{1}{M}\sum_{i=1}^n\sum_{j=1}^m \bar{\lambda}_{ij}\langle G_i(z_j^i;\xi_j^i), x^\star - x_{k+1}\rangle.\end{aligned} \tag{14}$$

Proof. In the proof we will apply the identity

$$\sum_{i=1}^n \frac{M_i}{2}\|y - z_m^i\|_2^2 = \frac{M}{2}\|y - \frac{1}{M}\sum_{i=1}^n M_i z_m^i\|_2^2 - \frac{1}{2M}\|\sum_{i=1}^n M_i z_m^i\|_2^2 + \sum_{i=1}^n \frac{M_i}{2}\|z_m^i\|_2^2$$

twice; once with $y = x^\star$ and once with $y = x_{k+1}$.

Using the three-points lemma $\frac{1}{2}\|b - c\|_2^2 - \frac{1}{2}\|a - c\|_2^2 = \langle a - b, c - b\rangle - \frac{1}{2}\|a - b\|_2^2$ we get

$$\frac{1}{2}\|x_{k+1} - x^\star\|_2^2 - \frac{1}{2}\|\bar{z} - x^\star\|_2^2 = \langle \bar{z} - x_{k+1}, x^\star - x_{k+1}\rangle - \frac{1}{2}\|x_{k+1} - \bar{z}\|_2^2. \tag{15}$$

Since $x_{k+1} = \mathbf{prox}_{\frac{1}{M}R}(\bar{z} - \frac{1}{M}\sum_{i=1}^n \sum_{j=1}^m \bar{\lambda}_{ij} G_i(z_j^i;\xi_j^i))$ it follows from optimality conditions for convex optimization that (Nesterov, 2018, Thm 3.1.23)

$$R(x_{k+1}) \leq R(y) + M\langle x_{k+1} - (\bar{z} - \frac{1}{M}\sum_{i=1}^n\sum_{j=1}^m \bar{\lambda}_{ij} G_i(z_j^i;\xi_j^i)), y - x_{k+1}\rangle \text{ for all } y \in \mathbb{R}^d.$$

If we let $y = x^\star$ and rearrange we get

$$\langle \bar{z} - x_{k+1}, x^\star - x_{k+1}\rangle \leq \frac{1}{M}(R(x^\star) - R(x_{k+1})) + \frac{1}{M}\sum_{i=1}^n\sum_{j=1}^m \bar{\lambda}_{ij}\langle G_i(z_j^i;\xi_j^i), x^\star - x_{k+1}\rangle.$$

Using this bound in (15) shows that

$$\begin{aligned}
\frac{1}{2}\|x_{k+1} - x^\star\|_2^2 - \frac{1}{2}\|\bar{z} - x^\star\|_2^2 &\leq \frac{1}{M}(R(x^\star) - R(x_{k+1})) + \frac{1}{M}\sum_{i=1}^n\sum_{j=1}^m \bar{\lambda}_{ij}\langle G_i(z_j^i;\xi_j^i), x^\star - x_{k+1}\rangle \\
&\quad - \frac{1}{2}\|x_{k+1} - \bar{z}\|_2^2 \\
&= \frac{1}{M}(R(x^\star) - R(x_{k+1})) + \frac{1}{M}\sum_{i=1}^n\sum_{j=1}^m \bar{\lambda}_{ij}\langle G_i(z_j^i;\xi_j^i), x^\star - x_{k+1}\rangle \\
&\quad - \frac{1}{M}\Big(\sum_{i=1}^n \frac{M_i}{2}\|x_{k+1} - z_m^i\|_2^2 + \frac{1}{2M}\|\sum_{i=1}^n M_i z_m^i\|_2^2 - \sum_{i=1}^n \frac{M_i}{2}\|z_m^i\|_2^2\Big).
\end{aligned}$$

We can rearrange to obtain

$$\begin{aligned}
&\frac{1}{2}\|x_{k+1} - x^\star\|_2^2 - \frac{1}{2}\Big(\|\frac{1}{M}\sum_{i=1}^n M_i z_m^i - x^\star\|_2^2 - \frac{1}{M^2}\|\sum_{i=1}^n M_i z_m^i\|_2^2 + \frac{1}{M}\sum_{i=1}^n M_i\|z_m^i\|_2^2\Big) \\
&\leq \frac{1}{M}(R(x^\star) - R(x_{k+1})) + \frac{1}{M}\sum_{i=1}^n\sum_{j=1}^m \bar{\lambda}_{ij}\langle G_i(z_j^i;\xi_j^i), x^\star - x_{k+1}\rangle - \frac{1}{2M}\sum_{i=1}^n M_i\|x_{k+1} - z_m^i\|_2^2.
\end{aligned}$$

Note that

$$\begin{aligned}
&\|\frac{1}{M}\sum_{i=1}^n M_i z_m^i - x^\star\|_2^2 - \frac{1}{M^2}\|\sum_{i=1}^n M_i z_m^i\|_2^2 + \frac{1}{M}\sum_{i=1}^n M_i\|z_m^i\|_2^2 \\
&= \frac{2}{M}\Big(\frac{M}{2}\|x^\star - \frac{1}{M}\sum_{i=1}^n M_i z_m^i\|_2^2 - \frac{1}{2M}\|\sum_{i=1}^n M_i z_m^i\|_2^2 + \sum_{i=1}^n \frac{M_i}{2}\|z_m^i\|_2^2\Big) \\
&= \frac{2}{M}\sum_{i=1}^n \frac{M_i}{2}\|x^\star - z_m^i\|_2^2 = \frac{1}{M}\sum_{i=1}^n M_i\|x^\star - z_m^i\|_2^2.
\end{aligned}$$

Hence, we conclude that

$$\begin{aligned}
\frac{1}{2}\|x_{k+1} - x^\star\|_2^2 - \frac{1}{2M}\sum_{i=1}^n M_i\|z_m^i - x^\star\|_2^2 &\leq \frac{1}{M}(R(x^\star) - R(x_{k+1})) - \frac{1}{2M}\sum_{i=1}^n M_i\|x_{k+1} - z_m^i\|_2^2 \\
&\quad + \frac{1}{M}\sum_{i=1}^n\sum_{j=1}^m \bar{\lambda}_{ij}\langle G_i(z_j^i;\xi_j^i), x^\star - x_{k+1}\rangle.
\end{aligned}$$

□

Lemma A.4. *The next iterate x_{k+1} satisfies*

$$\begin{aligned}
&\frac{1}{2}\|x_{k+1} - x^\star\|_2^2 - \frac{1}{2M}\sum_{i=1}^n M_i\|z_m^i - x^\star\|_2^2 \leq \frac{1}{M}(R(x^\star) - R(x_{k+1})) + \frac{1}{M}f(x^\star) + \frac{\delta}{M} \\
&- \frac{1}{M}\sum_{i=1}^n (F_i(z_m^i;\xi_m^i) + \langle G_i(z_m^i;\xi_m^i), x_{k+1} - z_m^i\rangle + \frac{M_i}{2}\|x_{k+1} - z_m^i\|_2^2) \\
&+ \frac{1}{M}\sum_{i=1}^n\sum_{j=1}^m \bar{\lambda}_{ij}(F_i(x^\star;\xi_j^i) - f_i(x^\star)).
\end{aligned}$$

Proof. Note that

$$\begin{aligned}\sum_{i=1}^{m}\sum_{j=1}^{m}\bar{\lambda}_{ij}\langle G_i(z_j^i;\xi_j^i), x^\star - x_{k+1}\rangle &= \sum_{i=1}^{m}\sum_{j=1}^{m}\bar{\lambda}_{ij}(\langle G_i(z_j^i;\xi_j^i), x^\star - z_j^i\rangle + \langle G_i(z_j^i;\xi_j^i), z_j^i - x_{k+1}\rangle)\\ &\le \sum_{i=1}^{n}\sum_{j=1}^{m}\bar{\lambda}_{ij}\big(F_i(x^\star;\xi_j^i) - F_i(z_j^i;\xi_j^i) + \langle G_i(z_j^i;\xi_j^i), z_j^i - x_{k+1}\rangle\big)\\ &= \sum_{i=1}^{n}\sum_{j=1}^{m}\bar{\lambda}_{ij}F_i(x^\star;\xi_j^i) - \sum_{j=1}^{m}\bar{\lambda}_{ij}(F_i(z_j^i;\xi_j^i) + \langle G_i(z_j^i;\xi_j^i), x_{k+1} - z_j^i\rangle)\\ &\le \sum_{i=1}^{n}\sum_{j=1}^{m}\bar{\lambda}_{ij}F_i(x^\star;\xi_j^i) - \sum_{i=1}^{n}\check{f}_i(x_{k+1};\boldsymbol{\xi}) + \delta.\end{aligned}$$

In the first inequality above we used the star-convexity of the oracle (see Assumption 4.10 for the stochastic case and Assumption 4.4 for the deterministic case). In the second inequality we used Lemma A.2.

Inserting this into Lemma A.3 yields

$$\begin{aligned}\frac{1}{2}\|x_{k+1} - x^\star\|_2^2 - \frac{1}{2M}\sum_{i=1}^{n}M_i\|z_m^i - x^\star\|_2^2 \le \frac{1}{M}(R(x^\star) - R(x_{k+1})) + \frac{1}{M}\sum_{i=1}^{n}\sum_{j=1}^{m}\bar{\lambda}_{ij}F_i(x^\star;\xi_j^i)\\ - \frac{1}{M}\sum_{i=1}^{n}\big(\check{f}_i(x_{k+1};\boldsymbol{\xi}) + \frac{M_i}{2}\|x_{k+1} - z_m^i\|_2^2\big) + \frac{\delta}{M}.\end{aligned}$$

The result now follows from adding and subtracting $(1/L)f(x^\star)$ from the right side of the inequality and dropping all but the most recent cut for every piecewise linear model. □

We now distinguish between the deterministic and stochastic case. The following result from (Feyzmahdavian et al., 2014) will be useful.

Lemma A.5. *Let $(V_k)_{k=0}^\infty$ be a non-negative sequence satisfying*

$$V_{k+1} \le qV_k + p\max_{(k-\tau)_+\le\ell\le k}V_\ell + r, \qquad k = 0, 1, 2, \ldots$$

for some non-negative constants p, q and r. If $q + p < 1$, then

$$V_k \le \rho^k V_0 + \epsilon, \qquad k = 0, 1, 2, \ldots,$$

where $\rho = (p + q)^{1/(1+\tau)}$, $\epsilon = r/(1 - p - q)$ and $(k - \tau)_+ = \max\{k - \tau, 0\}$.

Lemma A.5 will be used to analyze the algorithm under the quadratic functional growth assumption. For the analysis of the convex case, we present a new sequence result that may be of independent interest.

Lemma A.6. *Suppose that $(V_k)_{k=0}^\infty$ and $(W_k)_{k=0}^\infty$ are non-negative sequences satisfying*

$$V_{k+1} \le \max_{(k-\tau)_+\le\ell\le k}V_\ell - W_{k+1} + r, \qquad k = 0, 1, 2, \ldots \tag{16}$$

for a non-negative constant r. Then, for any $k \ge 1$,

$$\min_{t\le k}W_t \le \frac{(\tau+1)V_0}{k} + r. \tag{17}$$

Proof. We prove (17) by contradiction. Suppose that for some $K \ge 1$, (17) fails to hold. Then, for all $k \le K$,

$$W_k > \frac{(\tau+1)V_0}{K} + r. \tag{18}$$

Define $\mathcal{I}_0 = \{0\}$ and for any $t \ge 1$,

$$\mathcal{I}_t = [(t-1)(\tau+1)+1, t(\tau+1)].$$

Substituting (18) into (16) gives that for all $k \leq K-1$,

$$V_{k+1} < \max_{(k-\tau)_+ \leq \ell \leq k} V_\ell - \frac{(\tau+1)V_0}{K}. \tag{19}$$

Let $\tilde{t} = \lfloor K/(\tau+1) \rfloor$. For any $t \leq \tilde{t}$, $\mathcal{I}_t \subseteq [0, K]$. Then, using (19), we can derive that for all $t \leq \tilde{t}-1$,

$$\max_{k \in \mathcal{I}_{t+1}} V_k < \max_{k \in \mathcal{I}_t} V_k - \frac{(\tau+1)V_0}{K}.$$

Summing the above equation over $t \in [0, \tilde{t}-1]$ and noting that $\max_{k \in \mathcal{I}_0} V_k = V_0$ yields

$$\max_{k \in \mathcal{I}_{\tilde{t}}} V_k < V_0 - \frac{\tilde{t}(\tau+1)V_0}{K}. \tag{20}$$

Note that $\tilde{t} = \lfloor K/(\tau+1) \rfloor$. If $\tilde{t} = K/(\tau+1)$, then

$$\frac{\tilde{t}(\tau+1)V_0}{K} = V_0,$$

substituting which into (20) yields $\max_{k \in \mathcal{I}_{\tilde{t}}} V_k < 0$, which cannot be true because $V_k \geq 0$ for all $k \geq 0$.

If $\tilde{t} < K/(\tau+1)$, then we have $K > \tilde{t}(\tau+1)$. Then, by (19),

$$V_K < \max_{k \in \mathcal{I}_{\tilde{t}}} V_k - \frac{(\tau+1)V_0}{K}.$$

Moreover, since $\tilde{t} \geq K/(\tau+1) - 1$, from (20) we have

$$\max_{k \in \mathcal{I}_{\tilde{t}}} V_k < \frac{(\tau+1)V_0}{K}.$$

Combining the above two equations, we obtain

$$V_K < 0,$$

which cannot hold.

Concluding the above, (17) holds for all $k \geq 1$. □

Exact function values and gradients

First we prove convergence for exact (full-batch) function values and gradients.

Theorem A.7. *Under Assumptions 4.1, 4.2, 4.3, and 4.4 the iterates of Algorithm 1 using $M_i = L_i,\ i = 1, \ldots, n$ satisfy*

$$\|x_k - x^\star\|_2^2 \leq \rho^k \|x_0 - x^\star\|_2^2 + \epsilon_\delta, \tag{21}$$

where $\rho = (L/(L+\mu))^{1/(1+\tau)}$ and $\epsilon_\delta = 2\delta/\mu$.

Proof. We insert $M_i = L_i$, $M = L$ and $F_i(x^\star; \xi_j^i) = f_i(x^\star)$ into Lemma A.4 and use that f_i is smooth with parameter L_i (see Assumption 4.2) to conclude that

$$\begin{aligned}
&\frac{1}{2}\|x_{k+1} - x^\star\|_2^2 - \frac{1}{2L}\sum_{i=1}^n L_i \|z_m^i - x^\star\|_2^2 \leq \frac{1}{L}(R(x^\star) - R(x_{k+1})) + \frac{1}{L} f(x^\star) + \frac{\delta}{L} \\
&- \frac{1}{L}\sum_{i=1}^n (f_i(z_m^i) + \langle \nabla f_i(z_m^i), x_{k+1} - z_m^i \rangle + \frac{L_i}{2}\|x_{k+1} - z_m^i\|_2^2) \\
&\leq \frac{1}{L}(R(x^\star) - R(x_{k+1})) + \frac{1}{L} f(x^\star) + \frac{\delta}{L} - \frac{1}{L} f(x_{k+1}) \\
&= \frac{1}{L}(F(x^\star) - F(x_{k+1})) + \frac{\delta}{L}.
\end{aligned}$$

Now suppose that the quadratic functional growth condition of F (see Assumption 4.3) holds. We conclude that

$$\frac{1}{2}\|x_{k+1}-x^\star\|_2^2 - \frac{1}{2L}\sum_{i=1}^n L_i\|z_m^i - x^\star\|_2^2 \le -\frac{\mu}{2L}\|x_{k+1}-x^\star\|_2^2 + \frac{\delta}{L}.$$

After rearranging terms we get

$$\begin{aligned}\|x_{k+1}-x^\star\|_2^2 &\le \frac{1}{L+\mu}\sum_{i=1}^n L_i\|z_m^i - x^\star\|_2^2 + \frac{2\delta}{L+\mu}\\ &\le \frac{L}{L+\mu}\cdot \max_{(k-\tau)_+\le \ell\le k}\|x_\ell - x^\star\|_2^2 + \frac{2\delta}{L+\mu}.\end{aligned}$$

Applying Lemma A.5 yields the desired result. □

We now consider the case without the growth assumption.

Theorem A.8. *Under Assumptions 4.1, 4.2, and 4.4, the iterates of Algorithm 1 using $M_i = L_i$, $i = 1,\dots,n$, satisfy that for any $k \ge 1$,*

$$\min_{t\le k} F(x_t) - F(x^\star) \le \frac{(\tau+1)L\|x_0 - x^\star\|_2^2}{2k} + \delta. \tag{22}$$

Proof. From the proof of Theorem A.7 we know that

$$\frac{1}{2}\|x_{k+1}-x^\star\|_2^2 - \frac{1}{2L}\sum_{i=1}^n L_i\|z_m^i - x^\star\|_2^2 \le \frac{1}{L}(F(x^\star) - F(x_{k+1})) + \frac{\delta}{L}.$$

Note that $\|z_m^i - x^\star\|^2 \le \max_{(k-\tau)_+\le\ell\le k}\|x_\ell - x^\star\|^2$ for any $i = 1,\dots,n$. Equation (22) follows by applying Lemma A.6 with $V_\ell = \frac{1}{2}\|x_\ell - x^\star\|^2$, $W_{k+1} = \frac{1}{L}(F(x_{k+1}) - F(x^\star))$ and $r = \frac{\delta}{L}$. □

STOCHASTIC FUNCTION VALUES AND GRADIENTS

Next we prove the convergence for stochastic function values and gradients.

Theorem A.9. *Consider Algorithm 1 with $M_i = \alpha L_i$, $i = 1,\dots,n$ where $\alpha > 1$. Assume that stochastic function values and gradients are used. Under Assumptions 4.1, 4.2, and 4.10, the iterates of Algorithm 1 satisfy*

$$\min_{t\le k}\mathbf{E}[F(x_t)] - F(x^\star) \le \frac{\alpha(\tau+1)L\|x - x_0\|_2^2}{2k} + \epsilon, \tag{23}$$

where $\epsilon = \epsilon_\delta + \epsilon_{\sigma_1} + \epsilon_{\sigma_2}$ with

$$\epsilon_\delta = \delta, \qquad \epsilon_{\sigma_1} = n\sigma_1\sqrt{m}, \qquad \epsilon_{\sigma_2} = \frac{\sigma_2^2}{2(\alpha-1)}\cdot\sum_{i=1}^n\frac{1}{L_i}.$$

If, in addition, Assumption 4.3 holds, then

$$\mathbf{E}[\|x_k - x^\star\|_2^2] \le \rho^k\|x_0 - x^\star\|_2^2 + 2\epsilon/\mu,$$

where $\rho = (\alpha L/(\alpha L + \mu))^{1/(1+\tau)}$.

Proof. According to Lemma A.4 we have

$$\begin{aligned}&\frac{1}{2}\|x_{k+1}-x^\star\|_2^2 - \frac{1}{2M}\sum_{i=1}^n M_i\|z_m^i - x^\star\|_2^2 \le \frac{1}{M}(R(x^\star) - R(x_{k+1})) + \frac{1}{M}f(x^\star) + \frac{\delta}{M}\\ &- \frac{1}{M}\sum_{i=1}^n \underbrace{F_i(z_m^i;\xi_m^i) + \langle G_i(z_m^i;\xi_m^i), x_{k+1} - z_m^i\rangle + \frac{M_i}{2}\|x_{k+1} - z_m^i\|_2^2}_{\triangleq T_1}\\ &+ \frac{1}{M}\sum_{i=1}^n\sum_{j=1}^m \bar{\lambda}_{ij}(F_i(x^\star;\xi_j^i) - f_i(x^\star)).\end{aligned} \tag{24}$$

Using the Cauchy-Schwarz inequality together with $M_i = \alpha L_i$ and the assumption that worker i is smooth with parameter L_i, we bound T_1 according to (this holds for all $i = 1, \ldots, n$)

$$
\begin{aligned}
T_1 &= f_i(z_m^i) + \langle \nabla f_i(z_m^i), x_{k+1} - z_m^i \rangle + \frac{L_i}{2}\|x_{k+1} - z_m^i\|_2^2 \\
&\quad + F_i(z_m^i; \xi_m^i) - f_i(z_m^i) + \langle G_i(z_m^i; \xi_m^i) - \nabla f_i(z_m^i), x_{k+1} - z_m^i \rangle + \frac{(\alpha-1)L_i}{2}\|x_{k+1} - z_m^i\|_2^2 \\
&\geq f_i(x_{k+1}) + F_i(z_m^i; \xi_m^i) - f_i(z_m^i) + \frac{(\alpha-1)L_i}{2}\|x_{k+1} - z_m^i\|_2^2 \\
&\quad - \|G_i(z_m^i; \xi_m^i) - \nabla f_i(z_m^i)\|_2 \|x_{k+1} - z_m^i\|_2 \\
&\geq f_i(x_{k+1}) + F_i(z_m^i; \xi_m^i) - f_i(z_m^i) - \frac{1}{2(\alpha-1)L_i}\|G_i(z_m^i; \xi_m^i) - \nabla f_i(z_m^i)\|_2^2,
\end{aligned}
$$

where we in the last inequality used that $(b/2)t^2 - at \geq -a^2/(2b)$ for all $t \in \mathbf{R}$, $a \in \mathbf{R}$ and $b > 0$. Inserting this bound into (24) allows us to conclude that

$$
\begin{aligned}
&\frac{1}{2}\|x_{k+1} - x^\star\|_2^2 - \frac{1}{2M}\sum_{i=1}^n M_i \|z_m^i - x^\star\|_2^2 \leq \frac{1}{M}(R(x^\star) - R(x_{k+1})) + \frac{1}{M} f(x^\star) + \frac{\delta}{M} \\
&- \frac{1}{M}\sum_{i=1}^n \big(f_i(x_{k+1}) + F_i(z_m^i; \xi_m^i) - f_i(z_m^i) - \frac{1}{2(\alpha-1)L_i}\|G_i(z_m^i; \xi_m^i) - \nabla f_i(z_m^i)\|_2^2 \big) \\
&+ \frac{1}{M}\sum_{i=1}^n \sum_{j=1}^m \bar{\lambda}_{ij}(F_i(x^\star; \xi_j^i) - f_i(x^\star)).
\end{aligned}
$$

By taking expectations conditioned on all randomness up to the current iteration and using that $\mathbf{E}[F_i(z_m^i; \xi_i^m)] = f_i(z_m^i)$ where the expectation is conditional, we get

$$
\begin{aligned}
&\frac{1}{2}\,\mathbf{E}[\|x_{k+1} - x^\star\|_2^2\,] - \frac{1}{2M}\sum_{i=1}^n M_i \|z_m^i - x^\star\|_2^2 \leq \frac{1}{M}\,\mathbf{E}[F(x^\star) - F(x_{k+1})] + \frac{\delta}{M} \\
&+ \frac{1}{M}\sum_{i=1}^n \frac{1}{2(\alpha-1)L_i} \underbrace{\mathbf{E}[\|G_i(z_m^i; \xi_m^i) - \nabla f_i(z_m^i)\|_2^2]}_{\triangleq T_2} + \frac{1}{M}\sum_{i=1}^n \sum_{j=1}^m \bar{\lambda}_{ij}(F_i(x^\star; \xi_j^i) - f_i(x^\star)).
\end{aligned}
$$

From the assumption of bounded variance (see Assumption 4.10) we have $T_2 \leq \sigma_2^2$. Hence,

$$
\begin{aligned}
&\frac{1}{2}\,\mathbf{E}[\|x_{k+1} - x^\star\|_2^2\,] - \frac{1}{2M}\sum_{i=1}^n M_i \|z_m^i - x^\star\|_2^2 \leq \frac{1}{M}\,\mathbf{E}[F(x^\star) - F(x_{k+1})] + \frac{\delta}{M} \\
&+ \frac{\sigma_2^2}{2(\alpha-1)M}\sum_{i=1}^n \frac{1}{L_i} + \frac{1}{M}\sum_{i=1}^n \sum_{j=1}^m \bar{\lambda}_{ij}(F_i(x^\star; \xi_j^i) - f_i(x^\star)).
\end{aligned}
$$

After taking expectations again and using the tower property of conditional expectations we get

$$
\begin{aligned}
&\frac{1}{2}\,\mathbf{E}[\|x_{k+1} - x^\star\|_2^2\,] - \frac{1}{2M}\sum_{i=1}^n M_i\,\mathbf{E}[\|z_m^i - x^\star\|_2^2] \leq \frac{1}{M}\,\mathbf{E}[F(x^\star) - F(x_{k+1})] + \frac{\delta}{M} \\
&+ \frac{\sigma_2^2}{2(\alpha-1)M}\sum_{i=1}^n \frac{1}{L_i} + \frac{1}{M}\sum_{i=1}^n \underbrace{\mathbf{E}\left[\sum_{j=1}^m \bar{\lambda}_{ij}(F_i(x^\star; \xi_j^i) - f_i(x^\star))\right]}_{\triangleq T_3}.
\end{aligned}
$$

To bound T_3 we note that for $i = 1, \ldots, n$, by the Cauchy-Schwarz inequality, it holds that

$$\begin{aligned}
T_3 &\leq \mathbf{E}\left[\sum_{j=1}^{m} \bar{\lambda}_{ij} \cdot |F_i(x^\star, \xi_j^i) - f_i(x^\star)|\right] \\
&\leq \left(\mathbf{E}\left[\sum_{j=1}^{m} \bar{\lambda}_{ij}^2\right]\right)^{1/2} \cdot \left(\mathbf{E}\left[\sum_{j=1}^{m} (F_i(x^\star; \xi_j^i) - f_i(x^\star))^2\right]\right)^{1/2} \\
&\leq \left(\sum_{j=1}^{m} \mathbf{E}[(F_i(x^\star; \xi_j^i) - f_i(x^\star))^2]\right)^{1/2} \\
&\leq \sigma_1 \sqrt{m}.
\end{aligned}$$

Hence,

$$\frac{1}{2}\,\mathbf{E}[\|x_{k+1} - x^\star\|_2^2\,] - \frac{1}{2M}\sum_{i=1}^{n} M_i\,\mathbf{E}[\|z_m^i - x^\star\|_2^2] \leq \frac{1}{M}\,\mathbf{E}[F(x^\star) - F(x_{k+1})] + \frac{\delta}{M}$$
$$+ \frac{\sigma_2^2}{2(\alpha-1)M}\sum_{i=1}^{n}\frac{1}{L_i} + \frac{n\sigma_1\sqrt{m}}{M}.$$

Note that $\mathbf{E}[\|z_m^i - x^\star\|^2] \leq \max_{(k-\tau)_+ \leq \ell \leq k} \mathbf{E}[\|x_\ell - x^\star\|^2]$ for any $i = 1, \ldots, n$. Then, by Lemma A.6 with $V_\ell = \frac{1}{2}\,\mathbf{E}[\|x_\ell - x^\star\|^2]$, $W_{k+1} = \frac{1}{M}(\mathbf{E}[F(x_{k+1}) - F(x^\star)])$ and $r = \frac{\delta}{M} + \frac{\sigma_2^2}{2(\alpha-1)M}\sum_{i=1}^{n}\frac{1}{L_i} + \frac{n\sigma_1\sqrt{m}}{M}$, we have (23)

Now further assume that the quadratic functional growth of F (see Assumption 4.3) holds. We have

$$\frac{1}{2}\,\mathbf{E}[\|x_{k+1} - x^\star\|_2^2\,] - \frac{1}{2M}\sum_{i=1}^{n} M_i\,\mathbf{E}[\|z_m^i - x^\star\|_2^2] \leq -\frac{\mu}{2M}\,\mathbf{E}[\|x_{k+1} - x^\star\|_2^2] + \frac{\delta}{M}$$
$$+ \frac{\sigma_2^2}{2(\alpha-1)M}\sum_{i=1}^{n}\frac{1}{L_i} + \frac{n\sigma_1\sqrt{m}}{M}.$$

Rearranging the terms shows that

$$\begin{aligned}
\mathbf{E}[\|x_{k+1} - x^\star\|_2^2] &\leq \frac{1}{M+\mu}\sum_{i=1}^{n} M_i\,\mathbf{E}[\|z_m^i - x^\star\|_2^2] + \frac{2}{M+\mu}\left(\delta + n\sigma_1\sqrt{m} + \frac{\sigma_2^2}{2(\alpha-1)}\sum_{i=1}^{n}\frac{1}{L_i}\right) \\
&\leq \frac{M}{M+\mu} \cdot \max_{(k-\tau)_+ \leq \ell \leq k} \mathbf{E}[\|x_\ell - x^\star\|_2^2] + \frac{2}{M+\mu}\left(\delta + n\sigma_1\sqrt{m} + \frac{\sigma_2^2}{2(\alpha-1)}\sum_{i=1}^{n}\frac{1}{L_i}\right).
\end{aligned}$$

Applying Lemma A.5 yields the desired result. □

A.3 Solving the master problem

Here we discuss the complexity of solving the master problem (3) approximately. We use an accelerated projected gradient method to solve the dual problem

$$\begin{array}{ll} \text{minimize} & g(\lambda) \\ \text{subject to} & \lambda \in \mathbf{\Delta}. \end{array} \tag{25}$$

The objective function $g(\lambda)$ is defined in Lemma 3.1 and $\mathbf{\Delta} \subseteq \mathbf{R}^{mn}$ is the Cartesian product of n probability simplices of dimension m. In each iteration we must project onto $\mathbf{\Delta}$, which can be done at a cost of order $\mathcal{O}(nm \log m)$. The cost for evaluating the gradient (10) of the objective function $g(\lambda)$ is dominated by a term of order $\mathcal{O}(nmd)$, in addition to the cost of evaluating the proximal operator of R. (We recall that n is the number of workers, m is the bundle size, and d is the dimension of x.) If, for example, $R(x) = \lambda\|x\|_1$, then the cost of evaluating the proximal operator is $\mathcal{O}(d)$, so in this case the total cost per iteration of the projected gradient method is dominated by a term of order $\mathcal{O}(mnd)$. In practice we found that only a dozen of iterations was often sufficient to satisfy the termination criteria (6).

Modern interior-point solvers are fast, easy-to-use, robust, and good at exploiting sparsity. It is therefore natural to use an interior-point solver for solving the master problem (3). If $R(x) = \lambda\|x\|_1$, the master problem can be formulated as a quadratic program with a separable objective function and a coefficient matrix that is quite sparse (see, for example, (Andersen et al., 2011)).

To investigate the impact of our specialized approach for solving the subproblem, we ran ABM twice and solved the master problem with either the projected gradient method applied to the dual (25) using accuracy $\delta = 10^{-7}$, or the state-of-the-art interior-point solver Clarabel (Goulart & Chen, 2021) applied to the quadratic programming formulation of (3). For mnist8m, rcv1 and epsilon, the average time to solve the master problem was 0.017, 0.29, and 0.027 seconds for the gradient method, versus 0.20, 13.4, and 0.66 seconds for Clarabel. In other words, the gradient method (implemented in Python) is more than an order of magnitude faster. This comparison is not completely fair, since Clarabel in general finds a solution with higher accuracy. However, as shown in Theorem 4.5, it is not necessary to solve the master problem exactly to maintain convergence guarantees.

A.4 Additional Numerical Experiments

Convex Experiments

Figure 3 shows the performance of ABM, DAve-RPG and PIAG for binary logistic regression with $\lambda_2 = 0$, *i.e.,* when the objective function is convex but not strongly convex. The optimization trajectories are very similar to the progress in the strongly convex case for mnist8m and epsilon, but removing the strong convexity degrades the performance on rcv1 for all three methods (see Figure 1 in the main text). Nevertheless, ABM outperforms the two competitors.

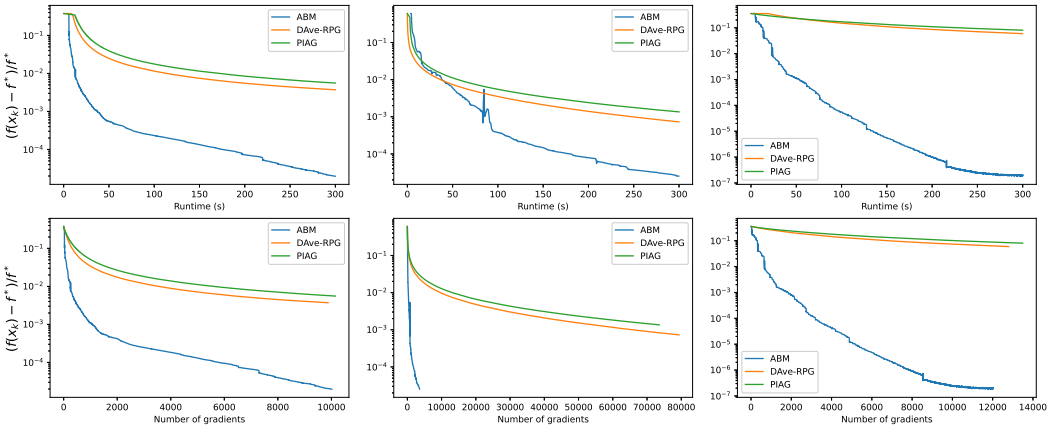


Figure 3: The progress of ABM, DAve-RPG and PIAG on the binary classification problems for $\lambda_2 = 0$. The datasets are arranged in the order mnist8m, rcv1 and epsilon from the left.

Benchmarking on Multinomial Logistic Regression

For the multiclass classification problems the objective function is

$$f(x) = -\frac{1}{N}\sum_{j=1}^{N}\sum_{k=1}^{K}\mathbf{1}\{y_j = k\}\log\left(\frac{e^{x_k^T a_j}}{\sum_{\ell=1}^{K}\exp(x_\ell^T a_j)}\right) + \frac{\lambda_2}{2}\sum_{k=1}^{K}\|x_k\|_2^2$$

$$R(x) = \lambda_1\sum_{k=1}^{K}\|x_k\|_1,$$

where $y_1, \ldots, y_N \in \{1, 2, \ldots, K\}$ are the labels. Here the decision variable is $x = (x_1, \ldots, x_K)$ where each x_j, $1 \le j \le K$ is a vector with dimension equal to the number of features. Figure 4 shows the relative suboptimality of ABM, DAve-RPG and PIAG on the two multiclass datasets SVHN and mnist8m. (The dimension of the problems can be found in Table 1 in the main text.)

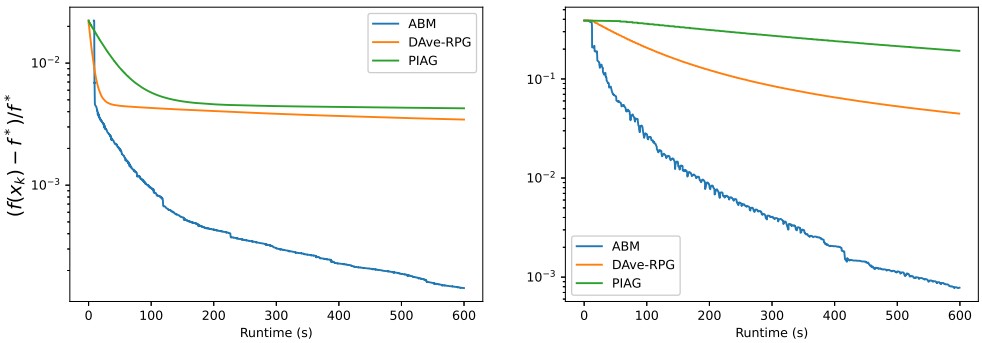


Figure 4: The progress of ABM, DAve-RPG and PIAG on the multiclass classification problems. The datasets are arranged in the order `SVHN` and `mnist8m` from the left.

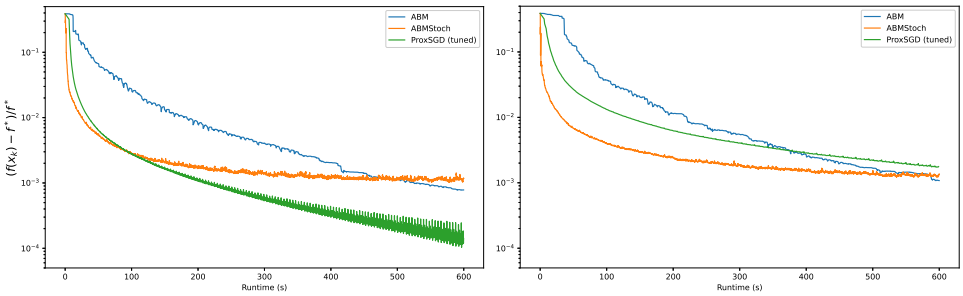
Stochastic function values and subgradients

We test the stochastic extension of `ABM` using mini-batches. We split each worker's data set into 100 mini-batches. For `ABM` based on exact function and gradient evaluations we estimate the smoothness parameters as described in §5. For the stochastic variant using mini-batches we estimate the smoothness parameters as follows. Each worker stores the last point, say z^i_{m-1}, it was queried in, and when queried again in a point z^i_m, the worker draws ξ representing a mini-batch and then evaluates both $G_i(z^i_m;\xi)$ and $G_i(z^i_{m-1};\xi)$. The worker can then estimate the smoothness parameter L_i with

$$\hat{L}_i = \frac{\|G_i(z^i_m;\xi) - G_i(z^i_{m-1};\xi)\|_2}{\|z^i_m - z^i_{m-1}\|_2}.$$

The worker then sends back both the gradient $G_i(z^i_m;\xi)$ and the smoothness estimate $\hat{L}_i$ to the central server.

For comparison we also implemented a synchronous proximal stochastic gradient method. The step size parameter was carefully tuned. The left part of Figure 5 shows the progress of `ABM`, the stochastic extension (`ABMStoch`), and the proximal stochastic gradient method (`ProxSGD`). The right part of Figure 5 shows the progress in the presence of some struggling workers. (For the right part we let one third of the workers have random delays uniformly distributed in the interval $[2t_{\text{grad}}, 4t_{\text{grad}}]$ every time they compute a gradient, where t_{grad} is the time required to compute the gradient.) It is interesting to note that while `ProxSGD` performs much worse for simulated delays, both `ABM` and `ABMStoch` are barely affected by the delays.



(a) (b)

Figure 5: **Left**: The progress of `ABM`, the stochastic variant `ABMStoch` and a tuned synchronous proximal stochastic gradient method `ProxSGD`. **Right**: The progress under simulated delays.