Variable metric proximal stochastic gradient methods with additional sampling

Nataša Krklec Jerinkić[1], Federica Porta[2], Valeria Ruggiero[3] and Ilaria Trombini[3,4*]

[1]Faculty of Sciences, Department of Mathematics and Informatics, University of Novi Sad, Trg Dositeja Obradovića 4, Novi Sad, 21000, Serbia.
[2]Department of Physics, Informatics and Mathematics, University of Modena and Reggio Emilia, Via Campi 213A, Modena, 41125, Italy.
[3]Department of Mathematics and Computer Science, University of Ferrara, Via Machiavelli 30, Ferrara, 44121, Italy.
[4]Department of Mathematical, Physical and Computer Sciences, University of Parma, Parco Area delle Scienze, 7/A, Parma, 43124, Italy.

*Corresponding author(s). E-mail(s): ilaria.trombini@unife.it;
Contributing authors: natasa.krklec@dmi.uns.ac.rs;
federica.porta@unimore.it; rgv@unife.it;

Abstract

Regularized empirical risk minimization problems arise in a variety of applications, including machine learning, signal processing, and image processing. Proximal stochastic gradient algorithms are a standard approach to solve these problems due to their low computational cost per iteration and a relatively simple implementation. This paper introduces a class of proximal stochastic gradient methods built on three key elements: a variable metric underlying the iterations, a stochastic line search governing the decrease properties and an incremental mini-batch size technique based on additional sampling. Convergence results for the proposed algorithms are proved under different hypotheses on the function to minimize. No assumption is required

regarding the Lipschitz continuity of the gradient of the differentiable part of the objective function. Possible strategies to automatically select the parameters of the suggested scheme are discussed. Numerical experiments on both binary classification and nonlinear regression problems show the effectiveness of the suggested approach compared to other state-of-the-art proximal stochastic gradient methods.

Keywords: Proximal stochastic gradient methods, Variable Metrics, Additional sampling, Machine Learning

MSC Classification: 65K05 , 90C15 , 62L20

1 Introduction

In this paper we are interested in solving the following composite optimization problem

$$\min_{x\in\mathbb{R}^d}\left\{H_{\mathcal{N}}(x) := f_{\mathcal{N}}(x) + R(x)\right\} \tag{1}$$

where $f_{\mathcal{N}}(x)$ is the average of many smooth component functions $f_i(x)$, i.e.,

$$f_{\mathcal{N}}(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x),$$

and R is a proper, lower semi-continuous and convex function, which may be non-differentiable. The problem (1) is often referred to as regularized empirical risk minimization [1] and covers a broad range of applications in machine learning (see, e.g. [2–5]) as well as in signal and image processing [6, 7].

1.1 Proximal gradient methods

Proximal gradient methods are a standard approach to solve problem (1). Indeed this algorithm consists of a forward step, which leverages the differentiability of $f_{\mathcal{N}}$, and a backward step, which exploits the convexity of R. Given an initial point $x^{(0)} \in \mathbb{R}^d$, the simplest proximal gradient method (Prox-GD) is based on the following update rule

$$x_{k+1} = \operatorname{argmin}_{x\in\mathbb{R}^d}\left\{\nabla f_{\mathcal{N}}(x_k)^T x + \frac{1}{2\alpha_k}\|x - x_k\|^2 + R(x)\right\}, \tag{2}$$

where α_k is a positive learning rate. By defining the proximal mapping of a convex function $R(\cdot)$ at $y \in \mathbb{R}^d$ as

$$\operatorname{prox}_R(y) = \operatorname{argmin}_{x\in\mathbb{R}^d}\left\{\frac{1}{2}\|x - y\|^2 + R(x)\right\},$$

the proximal gradient iteration (2) can be more compactly written as

$$x_{k+1} = \text{prox}_{\alpha_k R}(x_k - \alpha_k \nabla f_{\mathcal{N}}(x_k)). \tag{3}$$

Proximal gradient methods have been extensively studied, leading to various versions that incorporate features such as inertial steps, approximate computation of the proximal operator, variable metric strategies, and adaptive step length selection rules (see, for example, [8–10], the survey [11] and references therein). Below, we outline the class of variable metric proximal gradient algorithm, suggested by several authors [12–16], which serves as a starting point for the method proposed in this paper. Given proper positive parameters α_k and t_k, a general variable metric proximal gradient method can be defined through the following update

$$\begin{cases} d_k = \text{prox}_{\alpha_k R}^{S_k}(x_k - \alpha_k S_k^{-1} \nabla f_{\mathcal{N}}(x_k)) - x_k \\ x_{k+1} = x_k + t_k d_k \end{cases}. \tag{4}$$

Here, $\{S_k\} \subset \mathbb{R}^{d \times d}$ is a sequence of symmetric and positive definite scaling matrices, designed to capture some local features of the minimization problem without introducing significant additional computational costs. In this context, definition of the proximity operator of a convex function αR for $\alpha > 0$, with respect to a symmetric and positive definite matrix S, is generalized as

$$\text{prox}_{\alpha R}^{S}(y) = \text{argmin}_{x \in \mathbb{R}^d} \left\{ \frac{1}{2} \|x - y\|_S^2 + \alpha R(x) \right\},$$

where $\|\cdot\|_S$ denotes the norm induced by S.

1.2 Proximal stochastic gradient methods

When the number of components N is very large, computing $f_{\mathcal{N}}$ and its gradient may become unfeasible from the practical point of view. For this reason, a proper estimator of $f_{\mathcal{N}}$ is typically considered leading to the class of proximal stochastic gradient descent (Prox-SGD) methods. The update formula for Prox-SGD algorithms reads as

$$x_{k+1} = \text{prox}_{\alpha_k R}(x_k - \alpha_k \nabla f_{\mathcal{N}_k}(x_k)),$$

where $\mathcal{N}_k$ is a subset of $\mathcal{N}$ of size N_k, randomly and uniformly chosen at iteration k, and

$$f_{\mathcal{N}_k}(x) = \frac{1}{N_k} \sum_{i \in \mathcal{N}_k} f_i(x). \tag{5}$$

Hereafter $\mathcal{N}_k$ will be also referred to as mini-batch.

Prox-SGD offers an advantage over the Prox-GD scheme in (3) because it computes the gradient of only a relatively small number of randomly selected

functions f_i at each iteration. However, for Prox-SGD to converge, its step-size must decrease to zero at an appropriate rate, resulting in a convergence rate of $\mathcal{O}(1/\sqrt{k})$ for $\mathbb{E}[H_{\mathcal{N}}(x_k) - H_{\mathcal{N}}(x^*)]$ [17]. Additionally, even when $H_{\mathcal{N}}(x)$ is strongly convex, the convergence rate for Prox-SGD improves only to $\mathcal{O}(1/k)$ [18], which is significantly slower than the linear convergence rate achieved by Prox-GD.

To address the issue of diminishing step-size and slow convergence typical of standard stochastic gradient methods, either incremental techniques [19–25] or variance reduced strategies [26–32] have been proposed in the literature. Incremental stochastic gradient schemes are based on the idea of progressively increasing the mini-batch size over iterations, thereby employing increasingly accurate gradient estimates as the optimization process proceeds. The main limitations of such approaches are either the excessively rapid growth of the mini-batch size, as for the methods suggested in [20, 24], or the computationally expensive and memory demanding increasing conditions needed to be checked by the schemes proposed in [19, 21–23, 25]. On the other hand, variance reduced stochastic gradient algorithms require periodic computation of a full gradient (or an estimate based on a large mini-batch). For this reason, these approaches are generally not employed in applications involving large-scale datasets or deep neural networks.

Both incremental and variance reduced strategies allow to avoid vanishing sequences of learning rates, which are typically defined by a fixed value. Selecting this value is challenging, as it significantly impacts the numerical performance of the algorithms. Manual tuning of the learning rate through trial-and-error is common in this context, resulting in a high workload.

To conclude this section we remark that improved proximal stochastic gradient approaches can be also realized by combining incremental techniques with trust region strategies. In particular, some of the existing stochastic trust region methods [33–36] are based on adaptive rules for the selection of the mini-batch size that seem too stringent to be applied to deep learning applications, as they require the mini-batch size to be bigger than $1/\delta_k^4$ or $1/\delta_k^2$, with δ_k converging to zero.

1.3 Contributions

The main aim of this paper is to develop a stochastic version of the variable metric proximal gradient method, as defined in (4), where the gradient of the approximation (5) is used in place of the gradient of $f_{\mathcal{N}}$. The main ingredients of the suggested approach are the following.

(i) *Flexibility in selecting α_k and S_k.* The convergence results provided for the suggested scheme hold under very general assumptions on the parameters α_k and S_k. This flexibility allows users to adopt the most appropriate strategy for defining both the learning rate and the sequence of scaling matrices in order to accelerate convergence. Different definitions of α_k and S_k result in different methods. Possible automatic and adaptive techniques to select α_k

and S_k are discussed with the aim of avoiding manual parameter tuning. We stress that there is no requirement for the learning rate sequence to vanish.

(ii) *A stochastic Armijo-like line search to define* t_k. The additional parameter t_k is employed to ensure a sufficient reduction of the current stochastic approximation of the objective function (5) at each iteration. Examples of stochastic gradient schemes combined with line search procedures can also be found in [19, 22, 37–39]. However, in all of these works, the aim of the line search is to adjust the learning rate α_k and this approach can limit the flexibility of its selection. Moreover, since the starting value for t_k must be always set to 1 and then it is automatically and dynamically adjusted via the line search, t_k does not introduce any additional parameters that need tuning. On the other hand, the numerical performance of the schemes proposed in [19, 22, 37–39] depends on a proper strategy for selecting the initial guess for the line search.

(iii) *A proper incremental strategy to select the mini-batch size based on an additional sampling.* The algorithm developed in this paper falls within the class of incremental stochastic gradient methods. In particular, the mini-batch size increases (or stays the same) with each iteration based on the so-called additional sampling, employed for example in [40, 41]. With the additional sampling, alongside the mini-batch $\mathcal{N}_k$ used for the line search on t_k, a second (randomly chosen) mini-batch is introduced to evaluate whether a reduction, or at least a controlled increase, in the stochastic approximation of the objective function, related to this second mini-batch, is also achieved. If this situation does not occur, the mini-batch size is appropriately increased, as the current size probably does not ensure a reduction in the true objective function. In the approach suggested in this paper, the additional sampling is used to determine whether or not to keep the same mini-batch fixed, not only its size. In this way different successive iteration of the optimization process are employed to efficiently minimize the same stochastic approximation of the objective function, while not excluding the possibility to reduce the true objective function due to the presence of additional sampling. The idea of keeping the same mini-batch fixed for a predetermined number of iterations has also been considered in [41, 42]. Nevertheless, our method differs from those in [40–42] for two main reasons: first, the algorithms in [40–42] are thought for objective functions which do not incorporate a regularization term; second, they do not use a variable metric to enhance the convergence of the schemes. Lastly, it is worth noting that the method proposed in [43] also employs additional sampling to control step acceptance and adjust the sample size when needed. However, it differs significantly from our approach, as it is based on the trust-region framework and utilizes Hessian approximations

For general objective functions, we prove that the limit points of the sequence generated by the proposed algorithm are almost surely stationary. Furthermore, we establish the almost sure convergence of both the sequence of the

objective function values and the sequence of the iterates, given the additional assumptions of convexity and strong convexity for the objective function, respectively. All the convergence results hold without requiring the gradient of $f_{\mathcal{N}}$ to be Lipschitz continuous.

The suggested method has been applied to binary classification problems, demonstrating promising results compared to state-of-the-art algorithms, as well as robustness in parameter tuning.

1.4 Outline of the paper

The paper is organized as follows. In Section 2 we detail the structure of the general variable metric proximal stochastic gradient algorithm we are proposing. Moreover, a possibility to select both the learning rate and the scaling matrix is described. Finally the convergence results of the scheme are stated under different assumptions on the objective function. In Section 3 we report the results of the numerical experiments we carried out on two different regularized empirical risk minimization problems. Conclusions are presented in the final section, which also outlines directions for future work.

1.5 Notations

The following notations will be used throughout the paper.

- $\mathbb{R}_+$ is the set of non negative real numbers; $\mathbb{R}_{++}$ is the set of positive real numbers.
- $\|\cdot\|$ denotes the standard ℓ_2 norm. Given a symmetric and positive definite matrix S of order k, the S-norm of a vector $x \in \mathbb{R}^d$ is defined as $\|x\|_S \equiv \sqrt{x^T S x}$.
- Given $\mu \geq 1$, we denote by $\mathcal{M}_\mu$ the set of all symmetric positive definite matrices with all eigenvalues contained in the interval $[\frac{1}{\mu}, \mu]$.
- Let $D_1, D_2 \in \mathbb{R}^{d\times d}$ be symmetric and positive definite matrices. The notation $D_1 \succeq D_2$ indicates that $D_1 - D_2$ is a symmetric and positive semidefinite matrix or, equivalently, $x^T D_1 x \geq x^T D_2 x$ for any $x \in \mathbb{R}^d$.
- $\mathbb{E}[\cdot]$ and $\mathbb{E}[\cdot|\ \mathcal{F}]$ denote mathematical expectation and conditional expectation with respect to σ-algebra $\mathcal{F}$, respectively.
- We use "a.s." to abbreviate "almost sure/surely" and "i.i.d." to abbreviate "independent and identically distributed", while "SAA" stands for "sample average approximation".
- We denote by $|\mathcal{N}|$ the cardinality of set $\mathcal{N}$.
- Given a matrix $A \in \mathbb{R}^{d\times d}$, we denote by $\mathrm{diag}(A)$ the diagonal matrix whose diagonal elements are the diagonal metrics of A.

2 The algorithm and its convergence analysis

In this section we present a variable metric proximal stochastic gradient method based on both the line search and the additional sampling. Moreover the convergence analysis of the scheme will be provided.

2.1 The algorithm

The method we suggest is based on the following iteration

$$\begin{cases} d_k^{(\mathcal{N}_k)} = v_k^{(\mathcal{N}_k)} - x_k = \text{prox}_{\alpha_k R}^{S_k}(x_k - \alpha_k S_k^{-1} \nabla f_{\mathcal{N}_k}(x_k)) - x_k \\ x_{k+1} = x_k + t_k d_k^{(\mathcal{N}_k)}, \end{cases}$$

where

$\mathcal{N}_k$ is a randomly chosen subset of $\mathcal{N}$ of size N_k;
$\alpha_k \in \mathbb{R}$ is a positive learning rate such that $0 < \alpha_{min} \leq \alpha_k \leq \alpha_{max}$;
S_k is a symmetric and positive definite scaling matrix of size d;
$t_k \in (0, 1]$ is a line search parameter employed to ensure a sufficient decrease of the current approximation $H_{\mathcal{N}_k}(x) = f_{\mathcal{N}_k}(x) + R(x)$ of the objective function. Indeed, starting from $t_k = 1$, it is reduced by a factor $\beta \in (0, 1)$ until the following condition is met

$$H_{\mathcal{N}_k}(x_k + t_k d_k) \leq H_{\mathcal{N}_k}(x_k) + \eta t_k q_{\mathcal{N}_k}^{\alpha_k, S_k}(v_k^{(\mathcal{N}_k)}), \tag{6}$$

where

$$q_{\mathcal{N}_k}^{\alpha_k, S_k}(y) = (y - x_k)^T \nabla f_{\mathcal{N}_k}(x_k) + \frac{1}{2\alpha_k} \|y - x_k\|_{S_k}^2 + R(y) - R(x_k) \tag{7}$$

and $\eta \in (0, 1)$. It is immediate to prove that (6) ensures a reduction of $H_{\mathcal{N}_k}(\cdot)$ moving from x_k to $x_k + t_k d_k$ since $q_{\mathcal{N}_k}^{\alpha_k, S_k}(v_k^{(\mathcal{N}_k)})$ is non-positive. Indeed seeing that

$$\begin{aligned} v_k^{(\mathcal{N}_k)} &= \text{prox}_{\alpha_k R}^{S_k}(x_k - \alpha_k S_k^{-1} \nabla f_{\mathcal{N}_k}(x_k)) \\ &= \text{argmin}_{y \in \mathbb{R}^d} R(y) + \frac{1}{2\alpha_k} \|y - (x_k - \alpha_k S_k^{-1} \nabla f_{\mathcal{N}_k}(x_k))\|_{S_k}^2 \\ &= \text{argmin}_{y \in \mathbb{R}^d} \; q_{\mathcal{N}_k}^{\alpha_k, S_k}(y) + R(x_k) + \frac{\alpha_k}{2} \|\nabla f_{\mathcal{N}_k}(x_k)\|_{S_k}^2 \\ &= \text{argmin}_{y \in \mathbb{R}^d} \; q_{\mathcal{N}_k}^{\alpha_k, S_k}(y) \end{aligned}$$

the following inequality holds:

$$q_{\mathcal{N}_k}^{\alpha_k, S_k}(v_k^{(\mathcal{N}_k)}) \leq q_{\mathcal{N}_k}^{\alpha_k, S_k}(x_k) = 0.$$

It is well known that the function $q_{\mathcal{N}_k}^{\alpha_k, S_k}(\cdot)$ enjoys several properties that are recalled in Appendix A.

However, the update x_{k+1} is accepted only if it is able to ensure a sufficient decrease (or, at most, a controlled increase) of another approximation of the objective function computed on a different subsample. In more detail, given

a different subsample $\mathcal{D}_k$ randomly chosen from $\mathcal{N}$, the following checking Stochastic Descent (SD) condition is controlled:

$$H_{\mathcal{D}_k}(x_k + t_k d_k^{(\mathcal{N}_k)}) \leq H_{\mathcal{D}_k}(x_k) + c_{min} q_{\mathcal{D}_k}^{\bar{\alpha}}(v_k^{(\mathcal{D}_k)}) + C_{max}\zeta_k, \tag{8}$$

where, similarly to (7),

$$q_{\mathcal{D}_k}^{\bar{\alpha}}(y) = (y - x_k)^T \nabla f_{\mathcal{D}_k}(x_k) + \frac{1}{2\bar{\alpha}}\|y - x_k\|^2 + R(y) - R(x_k) \tag{9}$$

and $v_k^{(\mathcal{D}_k)} = \text{prox}_{\bar{\alpha}R}(x_k - \bar{\alpha}\nabla f_{\mathcal{D}_k}(x_k))$ with $\bar{\alpha} \in [\alpha_{min}, \alpha_{max}]$. The parameters c_{min} and C_{max} are positive real scalars and $\{\zeta_k\}$ is a summable sequence of non-negative real numbers. We remark that for the checking condition (8) we use a non-scaled gradient direction.

If condition (8) is not met the vector $x_k + t_k d_k^{(\mathcal{N}_k)}$ is rejected, $x_{k+1} = x_k$ and a new mini-batch $\mathcal{N}_{k+1}$ of larger size $N_{k+1} \in (N_k, N]$ is considered. Conversely, when the additional condition (8) is satisfied, then $x_{k+1} = x_k + t_k d_k^{(\mathcal{N}_k)}$ is accepted, and the optimization algorithm continues using the same approximation of the objective function, meaning that the mini-batch remains unchanged. Actually if the mini-batch remains unchanged for a predetermined number of iterations, a new mini-batch is randomly selected from $\mathcal{N}$, keeping the same cardinality as for the previous mini-batch. The main steps of the proposed approach are detailed in Algorithm 1 and described below.

Step 1 is devoted to the selection of a positive step length, belonging to a bounded and closed interval, and a symmetric and positive definite scaling matrix with bounded eigenvalues. Possible strategies to define these parameters are discussed in Section 2.3.

Step 2 aims at computing the proximal stochastic gradient direction given the mini-batch $\mathcal{N}_k$, the scaling matrix S_k and the learning rate α_k. If $q_{\mathcal{N}_k}^{\alpha_k, S_k}(v_k^{(\mathcal{N}_k)})$ is equal to zero, namely x_k is a stationary point for $H_{\mathcal{N}_k}$ (see item d. of Lemma A.1), then the mini-batch is changed.

Step 3 consists in the line search procedure on the parameter t_k. Until the sufficient decrease of the current approximation $H_{\mathcal{N}_k}$ of the objective function is not ensured in terms of (6), t_k is reduced by a factor $\beta < 1$. Lemma 2.1 guarantees that the line search is well defined.

Step 4 checks if the algorithm reaches the deterministic setting (namely $N_k = N$) or not. Particularly, if $N_k = N$ then Step 5, Step 6 and Step 7 are avoided since they only refer to the stochastic scenario. We highlight that the first four steps of Algorithm 1 with $N_k = N$ reduce to a deterministic variable metric proximal gradient method with line search (see for example [12]).

Step 5 implements the additional sampling. A different sub-sample $\mathcal{D}_k$ is considered in order to verify if the attempt vector $x_k + t_k d_k^{(\mathcal{N}_k)}$ also guarantees a sufficient decrease of the different approximation $H_{\mathcal{D}_k}$ of the objective function. If condition (8) is verified, the algorithm trusts $x_k + t_k d_k^{(\mathcal{N}_k)}$ and the mini-batch is kept fixed for the next iteration provided that the prefixed number

Algorithm 1 Proximal Stochastic gradient method with Additional sampling and variable Metric (**Prox-SAM**)

Fix $x_0 \in \mathbb{R}^d$, $\eta, \beta \in (0,1)$, $\{\zeta_k\} \subset \mathbb{R}_+$ subject to $\sum_{k=0}^{\infty} \zeta_k \leq \overline{\zeta} < \infty$, c_{min}, $C_{max} \in \mathbb{R}_{++}$, $0 < \alpha_{min} < \alpha_{max}$, $\bar{\alpha} \in [\alpha_{min}, \alpha_{max}]$, $\mu \geq 1$, $N_0 > 0$, $\mathcal{N}_0 \subseteq \mathcal{N}$ of size N_0, $m(N_0) > 0$, $flag = 0$.

for $k = 0, 1, \ldots$ **do**

STEP 1. *Parameters selection*
Set $\alpha_k \in [\alpha_{min}, \alpha_{max}]$.
Set $S_k \in \mathcal{M}_\mu$.

STEP 2. *Computation of a scaled stochastic direction*
$v_k^{(\mathcal{N}_k)} = \mathrm{prox}_{\alpha_k R}^{S_k}(x_k - \alpha_k S_k^{-1} \nabla f_{\mathcal{N}_k}(x_k))$
$d_k^{(\mathcal{N}_k)} = v_k^{(\mathcal{N}_k)} - x_k$
IF $q_{\mathcal{N}_k}^{\alpha_k, S_k}(v_k^{(\mathcal{N}_k)}) == 0$
THEN $N_{k+1} = N_k$, $flag = 0$ and go to STEP 7.
ELSE Set $t_k = 1$ and go to STEP 3.

STEP 3. *Line search procedure*
IF $H_{\mathcal{N}_k}(x_k + t_k d_k^{(\mathcal{N}_k)}) \leq H_{\mathcal{N}_k}(x_k) + \eta t_k q_{\mathcal{N}_k}^{\alpha_k, S_k}(v_k^{(\mathcal{N}_k)})$
THEN $\overline{x}_k = x_k + t_k d_k^{(\mathcal{N}_k)}$ and go to STEP 4.
ELSE $t_k = t_k \cdot \beta$ and repeat STEP 3.

STEP 4. *Check for deterministic or stochastic setting*
IF $N_k < N$
THEN go to STEP 5.
ELSE $x_{k+1} = \overline{x}_k$ and go to STEP 1.

STEP 5. *Additional sampling*
Choose $\mathcal{D}_k$ randomly and uniformly from $\mathcal{N}$ with replacement.
$v_k^{(\mathcal{D}_k)} = \mathrm{prox}_{\bar{\alpha} R}(x_k - \bar{\alpha} \nabla f_{\mathcal{D}_k}(x_k))$
$d_k^{(\mathcal{D}_k)} = v_k^{(\mathcal{D}_k)} - x_k$
IF $H_{\mathcal{D}_k}(\overline{x}_k) \leq H_{\mathcal{D}_k}(x_k) + c_{min} q_{\mathcal{D}_k}^{\bar{\alpha}}(v_k^{(\mathcal{D}_k)}) + C_{max}\zeta_k$
THEN $x_{k+1} = \overline{x}_k$, $N_{k+1} = N_k$, $flag = flag + 1$ and go to STEP 6.
ELSE $x_{k+1} = x_k$, $N_{k+1} \in (N_k, N]$, $flag = 0$ and go to STEP 7.

STEP 6. *Check for keeping the same mini-batch*
IF $flag < m(N_k)$
THEN go to STEP 1.
ELSE $flag = 0$ and go to STEP 7.

STEP 7. *Sample selection*
Randomly choose $\mathcal{N}_{k+1} \subseteq \mathcal{N}$ of size N_{k+1}.
Compute the possible maximum number $m(N_{k+1})$ of iterations with the same mini-batch.

end for

$m(N_k)$ of iterations with the same mini-batch has not been reached (see also STEP 6). Indeed, the support variable $flag$ counts the number of iterations performed with the same mini-batch. If $flag > m(N_k)$ then a new mini-batch of the same size is considered. Otherwise, if the additional sampling condition (8) does not hold, the attempt vector computed in STEP 1, STEP 2 and STEP 3

is rejected and the cardinality for the successive mini-batch is increased. When $d_k^{(\mathcal{N}_k)}$ satisfies (8), the reduction of $H_{\mathcal{D}_k}$, although relaxed by the presence of $C_{max}\zeta_k \geq 0$, can be considered as an indication that the decrease of $H_{\mathcal{N}_k}$ is acceptable in order to minimize the original objective function. In view of $\sum_{k=0}^{\infty} \zeta_k < \infty$, we have $\zeta_k \to 0$, so that the condition (8) becomes stricter as k increases. Furthermore, we highlight that there are no conditions on the size of $\mathcal{D}_k$, i.e., $\mathcal{D}_k$ can consist of only one element. Finally, we stress that if condition (8) fails to be satisfied for many iterations, as the size of the mini-batch is increased, there exists an iteration $\overline{k}$ such that, for $k \geq \overline{k}$, $N_k = N$ and the method is switched to a deterministic proximal gradient method combined with a line search.

Step 6 verifies if the same mini-batch has been already employed for $m(N_k)$ iterations.

Step 7 performs a new mini-batch selection. This step is reached only if either the additional sampling failed (namely $\bar{x}_k$ ensures a decrease only for $H_{\mathcal{N}_k}$) or the same mini-batch has been considered for $m(N_k)$ successive iterations. We remark that $m(N_k)$ can change along the iterative process.

We remark that it is possible to add a stopping criterion. Specifically, a stopping criterion should be checked before the procedure returns to Step 1.

2.2 The convergence analysis

Assumption 1. The non-negative real sequence $\{\zeta_k\}$ in (8) is such that $\sum_{k=0}^{\infty} \zeta_k \leq \overline{\zeta}$.

Assumption 2. There exists $\bar{H}_{\mathcal{N}} \in \mathbb{R}$ such that $H_{\mathcal{N}}(x) \geq \bar{H}_{\mathcal{N}}$, $\forall x \in \mathbb{R}^d$.

The first lemma regards some properties of the line search in Step 4 of Algorithm 1.

Lemma 2.1. *If the function $f_{\mathcal{N}}(\cdot)$ in (1) is continuously differentiable and the function $R(\cdot)$ in (1) is convex, then the line search procedure in* Step 3 *of Algorithm 1 is well defined.*

Proof The proof of this lemma is identical to the one of [12, Proposition 3.1]. However we report in Appendix A the main arguments for the sake of completeness. □

Let us denote by $\mathcal{D}_k^+$ the subset of all possible outcomes of $\mathcal{D}_k$ at iteration k for which the condition (8) is satisfied, i.e.,

$$\mathcal{D}_k^+ = \{\mathcal{D}_k \subset \mathcal{N} \mid H_{\mathcal{D}_k}(\overline{x}_k) \leq H_{\mathcal{D}_k}(x_k) + c_{min} q_{\mathcal{D}_k}^{\bar{\alpha}}(v_k^{(\mathcal{D}_k)}) + C_{max}\zeta_k\}. \quad (10)$$

We denote the complementary subset of outcomes at iteration k by

$$\mathcal{D}_k^- = \{\mathcal{D}_k \subset \mathcal{N} \mid H_{\mathcal{D}_k}(\overline{x}_k) > H_{\mathcal{D}_k}(x_k) + c_{min} q_{\mathcal{D}_k}^{\bar{\alpha}}(v_k^{(\mathcal{D}_k)}) + C_{max}\zeta_k\}. \quad (11)$$

The first lemma guarantees that if the mini-batches are always proper subsets of $\mathcal{N}$, then from a certain iteration forward the SD condition is always satisfied. The proof can be found in Appendix A.3 since its arguments are the same of the proofs of [41, Lemma 1] and [43, Lemma 1].

Lemma 2.2. *Suppose that Assumption 1 holds. If $N_k < N$ for all $k \in \mathbb{N}$, then a.s. there exists $k_1 \in \mathbb{N}$ such that $\mathcal{D}_k^- = \emptyset$ for all $k \geq k_1$.*

Next, we show that Lemma 2.2 implies that the Armijo-like inequality (8) holds for the overall objective function for all k sufficiently large in the mini-batch scenario. The proof is similar to the one of Lemma 2 in [43].

Lemma 2.3. *Suppose that Assumption 1 holds. If $N_k < N$ for all $k \in \mathbb{N}$, then, given $\bar{\alpha} > 0$ and $v_k^{(i)} = \mathrm{prox}_{\bar{\alpha}R}(x_k - \bar{\alpha}\nabla f_i(x_k))$,*

$$H_{\mathcal{N}}(\bar{x}_k) \leq H_{\mathcal{N}}(x_k) - \frac{c_{min}}{2\bar{\alpha}} \frac{1}{N} \sum_{i=1}^{N} \|x_k - v_k^{(i)}\|^2 + C_{max}\zeta_k,$$

holds a.s. for all $k \geq k_1$ where k_1 is as in Lemma 2.2.

Proof We first prove that the following inequality

$$q_{\mathcal{D}_k}^{\bar{\alpha}}(v_k^{(\mathcal{D}_k)}) \leq -\frac{1}{2\bar{\alpha}} \|v_k^{(\mathcal{D}_k)} - x_k\|^2 \tag{12}$$

holds true. In view of the definition of $v_k^{(\mathcal{D}_k)} = \mathrm{prox}_{\bar{\alpha}R}(x_k - \bar{\alpha}\nabla f_{\mathcal{D}_k}(x_k))$, it follows that

$$\frac{1}{\bar{\alpha}}(x_k - \bar{\alpha}\nabla f_{\mathcal{D}_k}(x_k) - v_k^{(\mathcal{D}_k)}) \in \partial R(v_k^{(\mathcal{D}_k)}). \tag{13}$$

Now we state an inequality for the elements of $\partial R(v_k^{(\mathcal{D}_k)})$. Indeed, for any $w \in \partial R(v_k^{(\mathcal{D}_k)})$ we have

$$\begin{aligned} q_{\mathcal{D}_k}^{\bar{\alpha}}(v_k^{(\mathcal{D}_k)}) &= \nabla f_{\mathcal{D}_k}(x_k)^T(v_k^{(\mathcal{D}_k)} - x_k) + \frac{1}{2\bar{\alpha}}\|v_k^{(\mathcal{D}_k)} - x_k\|^2 + \\ &\quad + R(v_k^{(\mathcal{D}_k)}) - R(x_k) \\ &\leq \nabla f_{\mathcal{D}_k}(x_k)^T(v_k^{(\mathcal{D}_k)} - x_k) + \frac{1}{2\bar{\alpha}}\|v_k^{(\mathcal{D}_k)} - x_k\|^2 + w^T(v_k^{(\mathcal{D}_k)} - x_k) \end{aligned}$$

Hence the previous inequality holds true for $\frac{1}{\bar{\alpha}}(x_k - \bar{\alpha}\nabla f_{\mathcal{D}_k}(x_k) - v_k^{(\mathcal{D}_k)})$ (see (13)). This results in

$$\begin{aligned} q_{\mathcal{D}_k}^{\bar{\alpha}}(v_k^{(\mathcal{D}_k)}) &\leq \nabla f_{\mathcal{D}_k}(x_k)^T(v_k^{(\mathcal{D}_k)} - x_k) + \frac{1}{2\bar{\alpha}}\|v_k^{(\mathcal{D}_k)} - x_k\|^2 + \\ &\quad + \frac{1}{\bar{\alpha}}(x_k - \bar{\alpha}\nabla f_{\mathcal{D}_k}(x_k) - v_k^{(\mathcal{D}_k)})^T(v_k^{(\mathcal{D}_k)} - x_k) \\ &= -\frac{1}{2\bar{\alpha}}\|v_k^{(\mathcal{D}_k)} - x_k\|^2 \end{aligned}$$

Lemma 2.2, together with (12), implies that a.s.

$$\begin{aligned} H_{\mathcal{D}_k}(\bar{x}_k) &\leq H_{\mathcal{D}_k}(x_k) + c_{min} q_{\mathcal{D}_k}^{\bar{\alpha}}(v_k^{(\mathcal{D}_k)}) + C_{max}\zeta_k \\ &\leq H_{\mathcal{D}_k}(x_k) - \frac{c_{min}}{2\bar{\alpha}}\|v_k^{(\mathcal{D}_k)} - x_k\|^2 + C_{max}\zeta_k \end{aligned} \tag{14}$$

holds for all possible realizations of $\mathcal{D}_k$ and for all $k \geq k_1$. Thus, we conclude that for every $i = 1, 2, ..., N$ and every $k \geq k_1$ a.s. we have

$$H_i(\overline{x}_k) \leq H_i(x_k) - \frac{c_{min}}{2\bar{\alpha}} \|v_k^{(i)} - x_k\|^2 + C_{max}\zeta_k,$$

where $H_i(x) = f_i(x) + R(x)$ and $v_k^{(i)} = \text{prox}_{\bar{\alpha}R}(x_k - \bar{\alpha}\nabla f_i(x_k))$. Indeed, if there exists $i \in \mathcal{N}$ that violates the previous inequality, then there would exist at least one realization of $\mathcal{D}_k$ (namely, $\mathcal{D}_k = \{i, i, ..., i\}$) that violates (14). Thus, a.s., for all $k \geq k_1$ we have

$$\begin{aligned} H_{\mathcal{N}}(\overline{x}_k) &= \frac{1}{N}\sum_{i=1}^{N} H_i(\overline{x}_k) \leq \frac{1}{N}\sum_{i=1}^{N}(H_i(x_k) - \frac{c_{min}}{2\bar{\alpha}}\|v_k^{(i)} - x_k\|^2 + C_{max}\zeta_k) \\ &= H_{\mathcal{N}}(x_k) - \frac{c_{min}}{2\bar{\alpha}}\frac{1}{N}\sum_{i=1}^{N}\|v_k^{(i)} - x_k\|^2 + C_{max}\zeta_k. \end{aligned}$$

□

Theorem 2.1. *Suppose that the Assumptions 1 and 2 hold and $N_k < N$, for all $k \in \mathbb{N}$. Let $\{x_k\}$ be a sequence generated by Algorithm 1. Then, a.s., any limit point of the sequence $\{x_k\}$ is a stationary point for problem (1).*

Proof Lemma 2.3 and $\bar{\alpha} \in [\alpha_{min}, \alpha_{max}]$ imply that there exists $k_1 \in \mathbb{N}$ such that a.s. the following inequality holds for all $k \geq k_1$

$$H_{\mathcal{N}}(x_{k+1}) = H_{\mathcal{N}}(\bar{x}_k) \leq H_{\mathcal{N}}(x_k) - \frac{c_{min}}{2\alpha_{max}}\frac{1}{N}\sum_{i=1}^{N}\|x_k - v_k^{(i)}\|^2 + C_{max}\zeta_k, \quad (15)$$

where the equality comes from the fact that $\mathcal{D}_k^- = \emptyset$ and thus the candidate point is accepted. By subtracting $\bar{H}_{\mathcal{N}}$ to both members of the previous inequality and applying the conditional expected value with respect to the σ-algebra generated by $k_1, \ldots, k$ we get

$$\begin{aligned} \mathbb{E}\left[H_{\mathcal{N}}(x_{k+1}) - \bar{H}_{\mathcal{N}} \mid \mathcal{F}_{k_1}^k\right] \leq H_{\mathcal{N}}(x_k) - \bar{H}_{\mathcal{N}} + \\ - \frac{c_{min}}{2\alpha_{max}}\frac{1}{N}\sum_{i=1}^{N}\|x_k - v_k^{(i)}\|^2 + C_{max}\zeta_k. \end{aligned}$$

where $\mathcal{F}_{k_1}^k$ denotes the the σ-algebra generated by $k_1, \ldots, k$. We note that both x_k and $v_k^{(i)}$ do not depend on $\mathcal{N}_k$ and hence are $\mathcal{F}_{k_1}^k$-measurable. In view of the Robbins-Siegmund lemma [44, Lemma 11], we can conclude that

$$\sum_{k=k_1}^{+\infty}\sum_{i=1}^{N}\|x_k - v_k^{(i)}\|^2 < +\infty, \quad a.s.$$

and, hence,

$$\lim_{k\to+\infty}\sum_{i=1}^{N}\|x_k - v_k^{(i)}\|^2 = 0, \quad a.s.$$

As a consequence, we claim that $\forall i = 1, \ldots, N$,

$$\lim_{k\to+\infty}\|x_k - v_k^{(i)}\|^2 = 0, \quad a.s. \quad (16)$$

Let us suppose that there exists a subsequence of $\{x_k\}$ that converges a.s. to $\bar{x}$, namely there exists $\mathcal{K} \subseteq \mathbb{N}$ such that

$$\lim_{k\to\infty,\, k\in\mathcal{K}} x_k = \bar{x} \text{ a.s.}$$

Moreover, the continuity of both the proximal operator and $\nabla f_i(\cdot)$ with respect to all their arguments, implies that, in view of (16),

$$\lim_{k\to\infty,\, k\in\mathcal{K}} v_k^{(i)} = \operatorname{prox}_{\bar{\alpha}R}(\bar{x} - \bar{\alpha}\nabla f_i(\bar{x})) = \bar{x}, \quad \forall i = 1, \ldots, N.$$

As a consequence, $-\nabla f_i(\bar{x}) \in \partial R(\bar{x})$, $\forall i = 1, \ldots, N$, and due to convexity of the subdifferential we conclude that $-\nabla f_{\mathcal{N}}(\bar{x}) = -\frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\bar{x}) \in \partial R(\bar{x})$, namely that $\bar{x}$ is a stationary point for $H_{\mathcal{N}}$ a.s.

□

By considering additional assumptions on the objective function, the convergence of both the sequence of the objective function values and the sequence of the iterates can be proved.

Theorem 2.2. *Suppose that the Assumptions 1 and 2 hold, $N_k < N$ for all $k \in \mathbb{N}$, and the objective function is convex. If the sequence $\{x_k\}$ generated by Algorithm 1 is bounded, then, a.s., the sequence of the objective function values $\{H_{\mathcal{N}}(x_k)\}$ converges to the minimum value $H_{\mathcal{N}}^*$ of $H_{\mathcal{N}}$.*

Proof By subtracting $H_{\mathcal{N}}^*$ to both sides of (15), the Robbins-Siegmund lemma implies that there exists a positive random variable Z such that $\{H_{\mathcal{N}}(x_k) - H_{\mathcal{N}}^*\}$ a.s. converges to Z. Since the sequence $\{x_k\}$ is bounded, it admits a subsequence $\{x_{k_j}\}$ that converges to some $\bar{x}$, which, according to the previous theorem, is a stationary point. Since the objective function is convex, $\bar{x}$ is a minimum point and $H_{\mathcal{N}}(\bar{x}) = H_{\mathcal{N}}^*$. By the continuity of $H_{\mathcal{N}}$, it follows that $H_{\mathcal{N}}(x_{k_j})$ converges to $H_{\mathcal{N}}(\bar{x}) = H_{\mathcal{N}}^*$. We can conclude that $Z = 0$ and that $\{H_{\mathcal{N}}(x_k)\}$ converges to $H_{\mathcal{N}}^*$ a.s. □

Theorem 2.3. *Suppose that the Assumptions 1 and 2 hold, $N_k < N$ for all $k \in \mathbb{N}$, and the objective function is $\mu_{\mathcal{N}}$-strongly convex. If the sequence $\{x_k\}$ generated by Algorithm 1 is bounded, then, a.s., $\{x_k\}$ converges to the unique solution x^* of problem (1).*

Proof If $H_{\mathcal{N}}$ is $\mu_{\mathcal{N}}$-strongly convex, problem (1) has a unique solution x^*. By denoting with $H_{\mathcal{N}}^*$ the value of the objective function in the solution, namely $H_{\mathcal{N}}(x^*) = H_{\mathcal{N}}^*$, in view of the strong convexity assumption for $H_{\mathcal{N}}$, the following inequality holds for any $x \in \mathbb{R}^d$:

$$\frac{\mu_{\mathcal{N}}}{2}\|x - x^*\|^2 \leq H_{\mathcal{N}}(x) - H_{\mathcal{N}}^*. \tag{17}$$

Since the sequence $\{x_k\}$ is bounded, Theorem 2.2 ensures that the sequence $\{H_{\mathcal{N}}(x_k) - H_{\mathcal{N}}^*\}$ converges to zero. As a consequence, taking x_k instead of x in (17) and letting $k \to \infty$, we obtain the convergence of the sequence generated by the algorithm to the unique solution x_* of problem (1). □

Theorem 2.4. *Suppose that the Assumption 2 holds and the full sample is reached. Then, any limit point of the sequence* $\{x_k\}$ *generated by Algorithm 1 is a stationary point. If moreover the function* $H_{\mathcal{N}}$ *is convex and the sequence* $\{S_k\} \subset \mathcal{M}_\mu$ *satisfies the following additional assumption*

$$S_{k+1} \preceq (1+\vartheta_k)S_k, \quad \{\vartheta_k\} \subset \mathbb{R}_+, \quad \sum_{k=0}^{+\infty} \vartheta_k < +\infty, \tag{18}$$

then the sequence $\{x_k\}$ *converges to a solution of* (1).

Proof If there exists $\bar{k}$ such that $N_k \geq N$, $\forall k \geq \bar{k}$, then Algorithm 1 reduces to a deterministic variable metric forward-backward method, whose convergence results are well-known in the literature. The reader is referred for example to [12, Theorem 3.1] and [12, Theorem 3.3]. □

Condition (18) states that the sequence $\{S_k\}_{k\in\mathbb{N}}$ asymptotically approaches a constant matrix [14, Lemma 2.3]. A possibility to practically fulfill this condition (see [12]) is to impose that

$$\{S_k\} \subseteq \mathcal{M}_{\mu_k}, \quad \text{where} \quad \mu_k^2 = 1+\xi_k, \quad \{\xi_k\} \subset \mathbb{R}_+, \quad \sum_{k=0}^{+\infty} \xi_k < +\infty. \tag{19}$$

Remark 2.1. We stress that the convergence analysis provided in this section has been developed without assuming the Lipschitz continuity of the gradient of the differentiable part of the objective function. Neither the definition of Lipschitz continuity nor any properties that follow from it were employed in the proofs.

2.3 Possible practical strategies to select α_k and S_k

A possibility to select the learning rate α_k consists in following the idea suggested in [41]. In particular the Barzilai-Borwein (BB) rules can be adopted in all those iterations where the mini-batch does not change. In this way, such rules are employed to efficiently minimize the approximation $H_{\mathcal{N}_k}(\cdot)$ of the objective function. In more detail, let us consider the scenario where the same mini-batch, $\mathcal{N}_k$, is held constant for $m \leq m(\mathcal{N}_k)$ iterations, specifically from iteration k to $k+m-1$. In this case, the learning rate α_j, for $j = k+1, \ldots, k+m-2$, can be chosen using one of the following BB selection rules, which account for the presence of a scaling matrix S_j^{-1} multiplying $\nabla f_{\mathcal{N}_k}(x_j)$:

$$\begin{aligned} \alpha_j^{BB1} &= \frac{z_{j-1}^T S_j z_{j-1}}{z_{j-1}^T y_{j-1}}, \\ \alpha_j^{BB2} &= \frac{z_{j-1}^T y_{j-1}}{y_{j-1}^T S_j^{-1} y_{j-1}}, \end{aligned} \tag{20}$$

where $z_{j-1} = x_j - x_{j-1}$ and $y_{j-1} = \nabla f_{\mathcal{N}_k}(x_j) - \nabla f_{\mathcal{N}_k}(x_{j-1})$. It is evident that, to compute the BB rules, two successive gradients related to the same mini-batch $\mathcal{N}_k$ are required. For this reason, when $j = k$, the learning rate α_j must be set equal to a different predefined value, such as

$$\alpha_j = \frac{1}{\|\nabla f_{\mathcal{N}_k}(x_j)\|}. \tag{21}$$

In the deterministic setting, many variants of the BB rules defined in (20) have been developed to make optimization gradient algorithms more effective. We recall here the so called ABB_{min} strategy [45] which is based on properly alternating the standard BB rules in (20); particularly, for $j = k+1, \ldots, m$, we get

$$\alpha_j^{ABB_{min}} = \begin{cases} \min\{\alpha_i^{BB2} \,|\, i = \max(1, j - M_\alpha), \ldots, j\} & \text{if} \frac{\alpha_j^{BB2}}{\alpha_j^{BB1}} < \tau \\ \alpha_j^{BB1} & \text{otherwise} \end{cases} \tag{22}$$

where $M_\alpha > 0$ is a prefixed integer constant and $\tau \in (0.5, 1)$. Every time the mini-batch changes, the BB rules can not be applied and the learning rate must be fixed differently. A possibility is to follow a strategy similar to (21). Algorithm 2 summarizes the learning rate selection technique just described. It is thought to be integrated in Step 1 of Algorithm 1.

Algorithm 2 BB-like learning rate selection rule

if $flag > 0$ **then**
  Compute $z_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = \nabla f_{\mathcal{N}_k}(x_k) - \nabla f_{\mathcal{N}_k}(x_{k-1})$.
  Compute α_k by means of (20) or (22).
else
  $\alpha_k = \dfrac{1}{\|\nabla f_{\mathcal{N}_k}(x_k)\|}$
end if
$\alpha_k = \min\left(\max\left(\alpha_{min}, \alpha_k\right), \alpha_{max}\right)$

As for the selection of the variable metric, we borrow the ideas of adaptive stochastic gradient methods such as AdaGrad [46], Adam [47] and AdaBelief [48]. The general update iteration of these algorithms can be written as

$$\begin{aligned} x_{k+1} &= x_k - \alpha_k S_k^{-1} m_k \\ m_k &= \gamma m_{k-1} + (1-\gamma)\nabla f_{\mathcal{N}_k}(x_k) \end{aligned}$$

where α_k is a positive learning rate, S_k is a preconditioning matrix and $\gamma \in [0, 1)$ is a constant momentum parameter. Adaptive gradient methods typically differ in how their preconditioners are constructed and whether or not they include the momentum term. For the selection of an appropriate scaling matrix

in Algorithm 1, we can consider the preconditioning matrices used in Adam, AdaBelief and AdaGrad, defined respectively as

$$\text{AdaBelief} \begin{cases} M_k = \beta_1 M_{k-1} + (1-\beta_1)\nabla f_{\mathcal{N}_k}(x_k) \\ g_k = \nabla f_{\mathcal{N}_k}(x_k) - M_k \\ S_k = \left(\frac{\beta_2 S_{k-1} + (1-\beta_2) diag\left(g_k g_k^T\right) + \varepsilon I}{1-\beta_2^{\tilde{k}}} \right)^{1/2} \end{cases} \tag{23}$$

$$\text{Adam} \left\{ S_k = \left(\frac{\beta S_{k-1} + (1-\beta) diag\left(\nabla f_{\mathcal{N}_k}(x_k) \nabla f_{\mathcal{N}_k}(x_k)^T\right) + \varepsilon I}{1-\beta^{\tilde{k}}} \right)^{1/2} \right. \tag{24}$$

$$\text{AdaGrad} \left\{ S_k = \left(S_{k-1} + diag\left(\nabla f_{\mathcal{N}_k}(x_k) \nabla f_{\mathcal{N}_k}(x_k)^T\right) + \varepsilon I \right)^{1/2} \right. \tag{25}$$

where $S_0 = 0$, $M_0 = 0$, $\beta, \beta_1, \beta_2 \in [0, 1)$, $\tilde{k} = k$, $\varepsilon > 0$ and I is the identity matrix. We remark that all these matrices are diagonal with positive diagonal elements, therefore satisfying the requirement to be symmetric and positive definite as needed by Algorithm 1. However, the convergence of Algorithm 1 is guaranteed if the eigenvalues of S_k lie within a suitable interval $\left[\frac{1}{\mu}, \mu\right]$, with $\mu > 0$. As a consequence, the diagonal elements of the matrices S_k in (24), (23) and (25) must be properly thresholded when employed in Algorithm 1. Since condition (18) is needed for the convergence of Algorithm 1 when the full sample is reached, we derive bounds for the scaling matrices based on (19). In particular, given s_k the diagonal of the matrix S_k defined according to one of the definitions (24)-(25), and a summable sequence $\{\xi_k\}$ of non-negative elements, we could impose that

$$s_k = \min\left(\mu_k, \max\left(s_k, \frac{1}{\mu_k}\right)\right) \tag{26}$$

where $\mu_k^2 = 1 + \xi_k$. Actually, requirement (26) is too restrictive to be forced at every iteration of Algorithm 1. Indeed when the mini-batch changes, the bounds for the diagonal elements of S_k can be widened again. The variable *flag* in Algorithm 1 accounts for the number of iterations with the same mini-batch and, hence, can be employed to reset the bounds for the scaling matrices. A similar strategy can also be adopted to set $\tilde{k}$ in (23) and (24) in order to strengthen the effect of the scaling matrix when the mini-batch changes. In Algorithm 3, we outline our resulting proposal to select the scaling matrix in Step 1 of Algorithm 1.

3 Numerical experiments on binary classification

In this section, we perform several numerical experiments on binary classification problem to evaluate the behaviour of the different versions of the

Algorithm 3 Scaling matrix selection rule

Define S_k by means of (23) with $\tilde{k} = flag$ or (24) with $\tilde{k} = flag$ or (25).
$\mu_k^2 = 1 + \xi_{flag}$
$\text{diag}(S_k) = \min\left(\mu_k, \max\left(\text{diag}(S_k), \frac{1}{\mu_k}\right)\right)$

Prox-SAM method. We consider four datasets: *w8a*, *IJCNN*, *RCV1* (downloadable from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/) and *MNIST* (available at https://yann.lecun.com/exdb/mnist/). For the *MNIST* dataset, we adapted this for binary classification by dividing it into the odd and even digit classes.

dataset	d	#training set (N)	#testing set
MNIST	784	60000	10000
w8a	300	44774	4975
IJCNN	22	49990	91701
RCV1	47236	20242	10000

Table 1: Dataset features.

In Table 1 the features of the considered datasets are summarized. Let us assume that (a_i, b_i), for $i = 1, ...N$, denotes the pair consisting of the feature vector $a_i \in \mathbb{R}^d$ and the class label $b_i \in \{1, -1\}$ of the $i-$th example. We consider two cases for the function $f_{\mathcal{N}}(x)$; in the first case, the terms of the finite sum are convex logistic regression (LR) loss functions:

$$f_i(x) = \log\left[1 + e^{-b_i a_i^T x}\right],$$

whereas, in the second case, we take into account a finite sum of non-convex loss functions in 2-layer neural networks (NN):

$$f_i(x) = \left(1 - \frac{1}{1 + e^{-b_i a_i^T x}}\right)^2.$$

The regularization term in the objective function $H_{\mathcal{N}}(x)$ has been chosen in two different ways:

- the L_1 norm, $R(x) = \lambda\|x\|_1$; the proximal operator of $\alpha R(x)$ in the S-norm (with S diagonal and positive definite matrix) is given by

$$\text{prox}_{\alpha R}^S(x) = sign(x) \cdot \max((|x| - \alpha\lambda S\mathbf{1}), 0)$$

  where $\mathbf{1}$ is a column vector of d ones, and the product and the absolute value function are intended component-wise;

- the squared L_2 norm, $R(x) = \frac{\lambda}{2}\|x\|_2^2$; the proximal operator of $\alpha R(x)$ in the S-norm (with S diagonal and positive definite matrix) is given by

$$\text{prox}_{\alpha R}^{S}(x) = \frac{x}{\mathbf{1} + \alpha\lambda S\mathbf{1}}$$

  where the quotient is intended component-wise.

Consequently, the objective function $H_{\mathcal{N}}(x)$ can take four forms, hereafter denoted as LR-L_1, NN-L_1, LR-L_2 and NN-L_2. The regularization parameter is fixed as $\lambda = 10^{-4}$ in all the test problems. For any numerical test, we performed 10 runs, leaving the possibility to the random number generator to vary. Therefore, the performance measures used to evaluate the obtained results are the average values of the following quantities:

- the optimality gap $H_{\mathcal{N}}(x_k) - H_{\mathcal{N}}^*$, computed on the training set at the end of any epoch. Here, $H_{\mathcal{N}}^*$ is an estimate of the minimum value, obtained by running a gradient iterative method for a large number of iterations;
- the accuracy computed on the testing set at the end of any epoch;
- the increase of the mini-batch size with respect to the iterations.

As a measure of computational complexity, we refer to an epoch. By epoch we mean the set of the computational resources and memory management required to compute a full gradient of the function $f_{\mathcal{N}}(x)$ or, equivalently, N individual gradients $\nabla f_i(x)$. For instance, the evaluation of $f_{\mathcal{N}_k}(x)$ or $\nabla f_{\mathcal{N}_k}(x)$ incurs a complexity of N_k/N epoch.
In all experiments, we set the following parameters: $C_{max} = 10^8$, $c_{min} = 10^{-4}$, $\eta = 0.4$, $\beta = 0.5$, $\zeta_k = 0.99^k$ for $k \geq 0$, the initial mini-batch size $N_0 = 10$, except when S_k is fixed to the identity matrix, in which case $N_0 = 1$; $\alpha_{min} = 10^{-8}$, $\alpha_{max} = 10^2$, $\bar{\alpha} = 1$ and $m(N_k) = N_k$. The mini-batch size N_k increases according to the rule $N_{k+1} = N_k + 1$.

3.1 L_1-regularized test problems

In this section, we present the results obtained by the **Prox-SAM** method for the test-problems LR-L_1 and NN-L_1. The stopping criterion is met when the total number of evaluations of the loss terms and their gradients reaches at least $N \cdot maxit$, where $maxit$ is the maximum number of allowed epochs. Here, $maxit$ is set to 20. We also report the execution times for each version of the method to highlight its effectiveness each version of **Prox-SAM** and its potential competitors.

3.1.1 Evaluating hyperparameter settings in Algorithm 1

The first numerical experiment compares different versions of **Prox-SAM** by varying the selection of either the learning rate α_k or the scaling matrix S_k. Specifically, we consider the following schemes:

- **Prox-SAM-BB**: Algorithm 1 with S_k equal to the identity matrix and α_k defined by Algorithm 2 equipped by (22);
- **Prox-SAM-I**: Algorithm 1 with S_k equal to the identity matrix and $\alpha_k = 1, \forall k$;
- **Prox-SAM-$S_k^{(1)}$**: Algorithm 1 with S_k selected by Algorithm 3 equipped by (23), $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-16}$ and $\alpha_k = 0.5$, $\forall k$;
- **Prox-SAM-$S_k^{(2)}$**: Algorithm 1 with S_k selected by Algorithm 3 equipped by (24), $\beta = 0.999$, $\varepsilon = 10^{-16}$ and $\alpha_k = 0.5, \forall k$;
- **Prox-SAM-$S_k^{(3)}$**: Algorithm 1 with S_k selected by Algorithm 3 equipped by (25), $\varepsilon = 10^{-16}$ and $\alpha_k = 0.5, \forall k$.

For all the scaled versions of **Prox-SAM** the sequence $\{\xi_k\}$ has been chosen as $\left\{\frac{10^5}{(k+1)^{2.1}}\right\}$.
Tables 2-3 report the averaged optimality gap, averaged accuracy along with the related standard deviations (STD) and averaged execution time (in seconds) over 10 runs for the LR-L_1 and NN-L_1 test problems, respectively. Based on these results we can conclude that, generally, the **Prox-SAM** algorithm with non-trivial scaling matrices exhibits the best effectiveness. In terms of numerical performance, the least effective version of **Prox-SAM** is **Prox-SAM-I**. Indeed, between the two non-scaled versions of **Prox-SAM**, the one combined with the BB-like rules is more efficient. The final accuracy reached by all the variants of **Prox-SAM** is comparable.

Figures 1-2 show the behaviour of the averaged optimality gap, the averaged accuracy and the increase of the averaged mini-batch size over 20 epochs. We can observe that, for the dataset *MNIST* and both the test problems, the configuration **Prox-SAM-$S_k^{(3)}$** appears very efficient. The accuracy is quite similar across all configurations, and the increase in mini-batch size is limited for all settings, even in relation to the size of the training set. Figures 3-4 show the comparison for the *IJCNN* dataset and the two test problems LR-L_1 and NN-L_1, respectively. We observe that the use of the scaling techniques is efficient also in this case. In the following comparison with other state-of-the-art algorithms, the **Prox-SAM-$S_k^{(3)}$** version of the proposed method is adopted.

3.1.2 Comparison of Prox-SAM-$S_k^{(3)}$ with other methods

The second numerical experiment concerns the comparison between **Prox-SAM-$S_k^{(3)}$** and several methods that represent the state-of-the-art in both deterministic and stochastic contexts. Specifically, among the deterministic methods, we consider the well-known **FISTA** method [49, 50] to solve the convex test problem LR-L_1 and the standard forward-backward or proximal gradient iterative scheme (**Prox-FB**) [51] for the non-convex NN-L_1 problem. For both methods, the version with backtracking is used; at each iteration, the initial trial value of the step length for this procedure is set equal to $\min(\alpha_0 \equiv 1/L, 2\alpha_{k-1})$, $k > 0$, where L is the Lipschitz constant of $\nabla f_{\mathcal{N}}$, which can be estimated for the considered datasets. The accelerated

Method		*MNIST*	*w8a*	*IJCNN*	*RCV1*
Prox-SAM-BB	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0320 $\pm$0.0203	0.0069 $\pm$0.0015	0.0074 $\pm$0.0036	0.0154 $\pm$0.0009
	$A(\bar{x})$ $\pm STD$	0.8854 $\pm$0.0095	0.9023 $\pm$0.0016	0.9192 $\pm$0.0020	0.9411 $\pm$0.0009
	Time (s)	4.3989	1.7123	0.6528	3.6745
Prox-SAM-I	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.1302 $\pm$0.0118	0.0142 $\pm$0.0018	0.0039 $\pm$0.0004	0.0865 $\pm$0.0012
	$A(\bar{x})$ $\pm STD$	0.8832 $\pm$0.0030	0.9001 $\pm$0.0011	0.9152 $\pm$0.0010	0.9396 $\pm$0.0013
	Time (s)	5.1700	1.4906	0.6037	3.9884
Prox-SAM-$S_k^{(1)}$	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0112 $\pm$0.0037	0.0092 $\pm$0.0013	0.0012 $\pm$0.0005	0.0294 $\pm$0.0009
	$A(\bar{x})$ $\pm STD$	0.8956 $\pm$0.0024	0.9014 $\pm$0.0013	0.9190 $\pm$0.0011	0.9395 $\pm$0.0012
	Time (s)	5.7853	1.6218	0.7124	4.2884
Prox-SAM-$S_k^{(2)}$	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0103 $\pm$0.0024	0.0098 $\pm$0.0019	0.0011 $\pm$0.0004	0.0315 $\pm$0.0010
	$A(\bar{x})$ $\pm STD$	0.8964 $\pm$0.0015	0.9007 $\pm$0.0015	0.9183 $\pm$0.0011	0.9392 $\pm$0.0019
	Time (s)	5.7489	1.6252	0.6741	4.3169
Prox-SAM-$S_k^{(3)}$	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0105 $\pm$0.0017	0.0115 $\pm$0.0010	0.0013 $\pm$0.0004	0.0339 $\pm$0.0013
	$A(\bar{x})$ $\pm STD$	0.8968 $\pm$0.0007	0.8997 $\pm$0.0010	0.9179 $\pm$0.0014	0.9383 $\pm$0.0012
	Time (s)	5.7707	1.5883	0.6746	4.2041

Table 2: Results obtained for the test problem LR-L_1 with different versions of **Prox-SAM**.

behaviour of the **FISTA** scheme is achieved as specified in [50] (with $a = 2.1$, using the notation of the cited paper).

In the stochastic framework, we compare the **Prox-SAM-$S_k^{(3)}$** method with the following methods: **Prox-SARAH** [28], **Prox-Spider-boost** [52] and **Prox-LISA** [22]. These algorithms are equipped with the hyperparameter settings specified in the cited papers. For the sake of completeness, we report

Method		*MNIST*	*w8a*	*IJCNN*	*RCV1*
Prox-SAM-BB					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0091 $\pm$0.0035	0.0018 $\pm$0.0004	0.0009 $\pm$0.0005	0.0031 $\pm$0.0002
	$A(\bar{x})$ $\pm STD$	0.8902 $\pm$0.0061	0.8961 $\pm$0.0022	0.9244 $\pm$0.0023	0.9290 $\pm$0.0013
	Time (s)	5.2144	1.8716	0.7267	3.7748
Prox-SAM-I					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0082 $\pm$0.0007	0.0024 $\pm$0.0005	0.0051 $\pm$0.0007	0.0296 $\pm$0.0002
	$A(\bar{x})$ $\pm STD$	0.8945 $\pm$0.0022	0.8974 $\pm$0.0014	0.9120 $\pm$0.0015	0.9329 $\pm$0.0013
	Time (s)	6.0872	1.3674	0.6446	4.0937
Prox-SAM-$S_k^{(1)}$					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0026 $\pm$0.0011	0.0026 $\pm$0.0004	0.0004 $\pm$0.0001	0.0082 $\pm$0.0004
	$A(\bar{x})$ $\pm STD$	0.8990 $\pm$0.0016	0.8948 $\pm$0.0012	0.9258 $\pm$0.0021	0.9287 $\pm$0.0010
	Time (s)	6.1572	1.7564	0.7865	4.3621
Prox-SAM-$S_k^{(2)}$					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0024 $\pm$0.0007	0.0028 $\pm$0.0006	0.0004 $\pm$0.0001	0.0090 $\pm$0.0004
	$A(\bar{x})$ $\pm STD$	0.8980 $\pm$0.0021	0.8944 $\pm$0.0013	0.9242 $\pm$0.0024	0.9289 $\pm$0.0015
	Time (s)	5.9182	1.8103	0.7804	4.4766
Prox-SAM-$S_k^{(3)}$					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0024 $\pm$0.0009	0.0029 $\pm$0.0006	0.0004 $\pm$0.0001	0.0099 $\pm$0.0005
	$A(\bar{x})$ $\pm STD$	0.8993 $\pm$0.0010	0.8946 $\pm$0.0012	0.9243 $\pm$0.0014	0.9286 $\pm$0.0012
	Time (s)	6.0052	1.7864	0.7411	4.4195

Table 3: Results obtained for the test problem NN-L_1 with different versions of **Prox-SAM**.

these values in Appendix B. As in the previous experiment, for any numerical test we performed 10 runs and reported averaged results. Due to significant differences in the nature of the considered iterative schemes, we allocated a time budget of 15 seconds per run and analyzed the behaviour of the averaged metrics over this period. We remind that **Prox-SARAH** and **Prox-Spider-boost** are hybrid schemes based on outer-inner iterations. In each outer

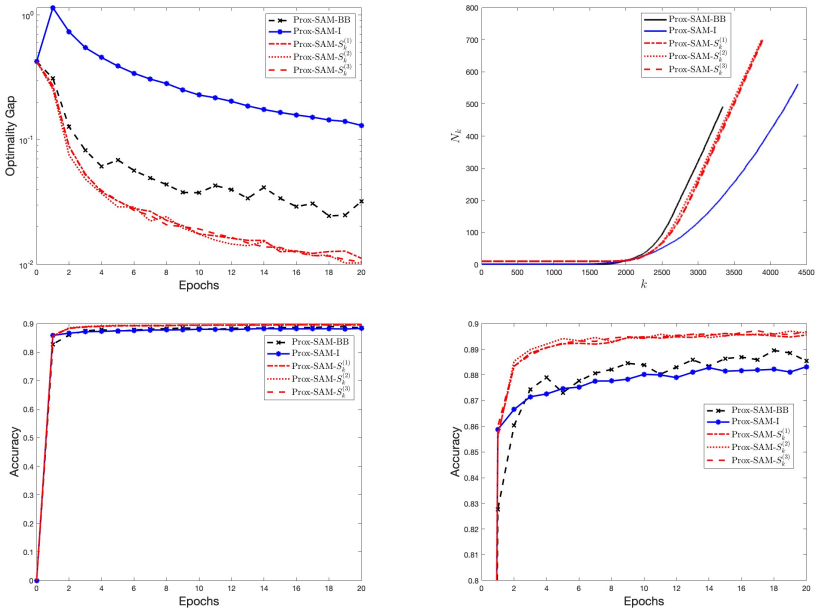

Fig. 1: Test problem LR-L_1 for the *MNIST* dataset - First row: averaged optimality gap computed on the training set (left panel) and increase of the averaged mini-batch size (right panel). Second row: averaged accuracy evaluated on the test set (left panel) and a zoom of the accuracy in the interval $[0.8, 0.9]$ (right panel).

iteration, the full gradient at the current iterate (or an estimate based on a large mini-batch) is computed and used to evaluate the stochastic gradients during the subsequent inner steps m, where m is of the order of N. Our implementation of these methods is based on the codes available for download at https://github.com/unc-optimization/StochasticProximalMethods. Here, at each outer iteration, the computation of the full gradient is performed.
On the other hand, the **Prox-LISA** algorithm is a stochastic gradient method that uses a line search technique to select the learning rate and includes a test to manage the stochastic gradient variance by appropriately increasing the mini-batch size.

In Tables 4 and 5 we report the averaged optimality gap and the averaged accuracy achieved after 15 seconds of runtime by the considered methods for the two test problems LR-L_1 and NN-L_1, respectively. Moreover, to evaluate the complexity of the different methods, we also report the number of epochs processed within the 15-second time budget. The results in the tables highlight that, given the same time budget, the **Prox-SAM-**$S_k^{(3)}$ method generally outperforms both the deterministic scheme (**FISTA** or **Prox-FB**), and the hybrid methods (**Prox-SARAH** and **Prox-Spider-boost**) in terms of

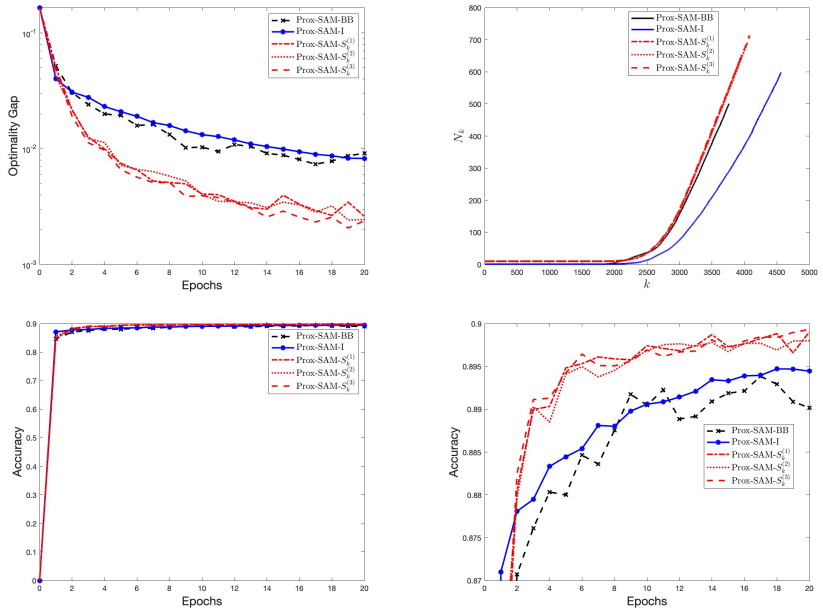

Fig. 2: Test problem NN-L_1 for the *MNIST* dataset - First row: averaged optimality gap computed on the training set (left panel) and increase of the average of the mini-batch size (right panel). Second row: averaged accuracy evaluated on the test set (left panel) and a zoom of the accuracy in the interval $[0.87, 0.90]$ (right panel).

optimality gap values. Specifically, **Prox-SARAH** and **Prox-Spider-boost** process fewer epochs due to their periodic need for computationally expensive full-gradient evaluations. In contrast, **Prox-SAM-**$S_k^{(3)}$ performs comparably to **Prox-LISA**, which, however, can process more epochs. Notably, within the given time budget, **Prox-SAM-**$S_k^{(3)}$ and **FISTA/Prox-FB** process a similar number of epochs, but the stochastic behaviour of **Prox-SAM-**$S_k^{(3)}$ enables it to achieve greater efficiency. Finally, we remark that the performance of **Prox-LISA** depends on multiple hyperparameters (for example, for variance control tests) that, while robust, still require tuning. On the other hand, **Prox-SAM-**$S_k^{(3)}$ has fewer key hyperparameters to select initially (such as N_0).

In Figure 5 the averaged optimality gap, the averaged accuracy and the averaged increase of the mini-batch size related to the test problem LR-L_1 for the *RCV1* dataset are shown. In the optimality gap plot, the STD is represented by the shaded area around the curves. We observe that, in terms of the optimality gap, the behaviour of the considered methods is similar at the end. In the initial seconds, as is typical for stochastic schemes, **Prox-SAM-**$S_k^{(3)}$ and **Prox-LISA** achieve a more rapid reduction in the optimality gap compared

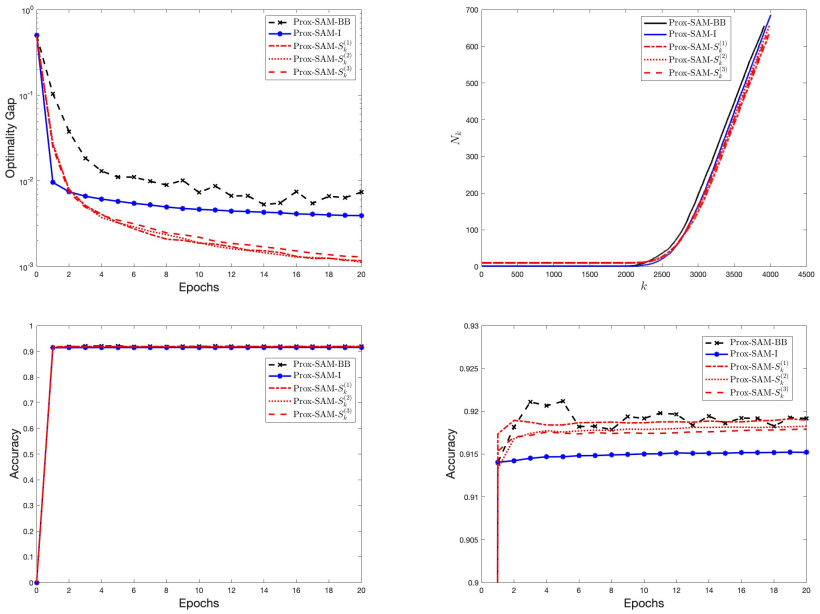

Fig. 3: Test problem LR-L_1 for the *IJCNN* dataset - First row: averaged optimality gap computed on the training set (left panel) and increase of the averaged mini-batch size (right panel). Second row: averaged accuracy evaluated on the test set (left panel) and a zoom of the accuracy in the interval $[0.90, 0.93]$ (right panel).

to **FISTA**. The final accuracy values for **FISTA** and **Prox-LISA** are slightly higher than those obtained by **Prox-SAM-**$S_k^{(3)}$.

Figure 6 shows the results obtained for the test problem NN-L_1 with the *IJCNN* dataset. We observe that **Prox-SAM-**$S_k^{(3)}$ performs better than the other methods from the very beginning of the process.

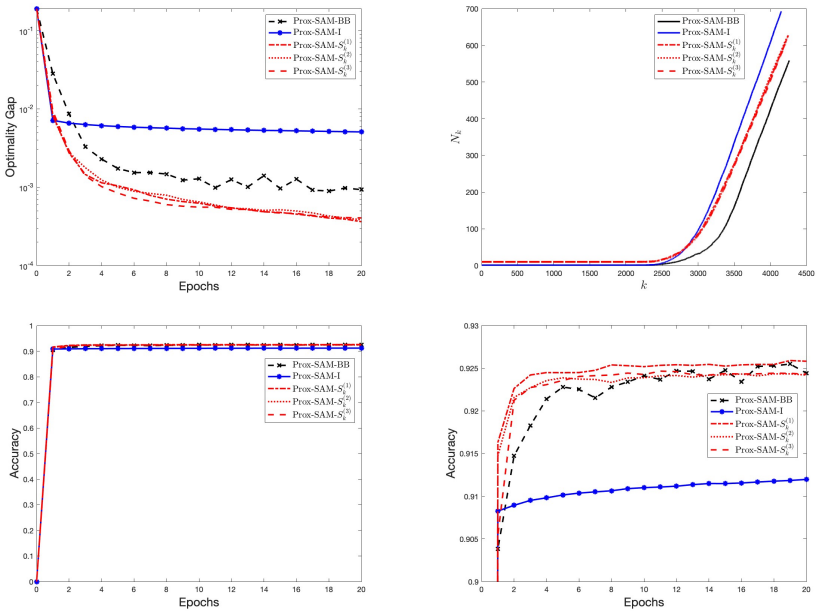

Fig. 4: Test problem NN-L_1 for the *IJCNN* dataset - First row: averaged optimality gap computed on the training set (left panel) andincrease of the average of the mini-batch size (right panel). Second row: averaged accuracy evaluated on the test set (left panel) and a zoom of the accuracy in the interval $[0.90, 0.93]$ (right panel).

Method		*MNIST*	*w8a*	*IJCNN*	*RCV1*
Prox-SAM-$S_k^{(3)}$					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$ $\pm STD$	0.0057 $\pm$0.0015	0.0035 $\pm$0.0003	$5.40e^{-5}$ $\pm 2.06e^{-5}$	0.0142 $\pm$0.0003
	$A(\bar{x})$ $\pm STD$	0.8977 $\pm$0.0013	0.9029 $\pm$0.0010	0.9188 $\pm$0.0002	0.9432 $\pm$0.0007
	Epochs	48	186	900	228
Prox-SARAH					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$ $\pm STD$	0.0055 $\pm$ 0.0001	0.0072 $\pm$ 0.0003	0.0017 $\pm$ 0.0001	0.0863 $\pm$ 0.0011
	$A(\bar{x})$ $\pm STD$	0.8975 $\pm$0.0004	0.9003 $\pm$0.0006	0.9162 $\pm$ 0.0001	0.9391 $\pm$0.0009
	Epochs	32	65	27	5
Prox-Spider-boost					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$ $\pm STD$	0.0050 $\pm$ 0.0001	0.0171 $\pm$ 8.73e^{-6}	0.0011 $\pm$ 8.04e^{-5}	0.0834 $\pm$ 0.0011
	$A(\bar{x})$ $\pm STD$	0.8978 $\pm$0.0005	0.8983 $\pm$0.0000	0.9168 $\pm$ 8.23e^{-5}	0.9393 $\pm$0.0009
	Epochs	34	295	30	5
Prox-LISA					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$ $\pm STD$	0.0014 $\pm$0.0004	0.0018 $\pm$ 0.0001	$7.66e^{-6}$ $\pm$ 2.77e^{-6}	0.0295 $\pm$0.0003
	$A(\bar{x})$ $\pm STD$	0.8982 $\pm$0.0007	0.9037 $\pm$0.0003	0.9190 $\pm$0.0001	0.9449 $\pm$0.0008
	Epochs	70	221	455	23
FISTA					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$ $A(\bar{x})$	0.1120 0.8395	0.0318 0.8953	$1.37e^{-6}$ 0.9190	0.0053 0.9474
	Epochs	47	243	943	373

Table 4: Results for the LR-L_1 test problem after 15 seconds of runtime.

Method		*MNIST*	*w8a*	*IJCNN*	*RCV1*
Prox-SAM-$S_k^{(3)}$					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$	0.0009	0.0012	$3.62e^{-5}$	0.0025
	$\pm STD$	± 0.0002	± 0.0003	$\pm 1.40e^{-5}$	$\pm 8.43e^{-5}$
	$A(\bar{x})$	0.8999	0.8979	0.9252	0.9317
	$\pm STD$	± 0.0014	± 0.0012	± 0.0008	± 0.0005
	Epochs	53	183	1016	292
Prox-SARAH					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$	0.0024	0.0017	0.0202	0.0226
	$\pm STD$	± 0.0001	± 0.0003	$\pm$ 0.0152	$\pm$ 0.0023
	$A(\bar{x})$	0.8976	0.8970	0.9223	0.9291
	$\pm STD$	± 0.0005	± 0.0008	$\pm$ 0.0093	± 0.0038
	Epochs	37	70	33	5
Prox-Spider-boost					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$	0.0051	0.0032	0.0214	0.0248
	$\pm STD$	$\pm 7.62e^{-6}$	$\pm 2.33e^{-6}$	$\pm$ 0.0153	$\pm$ 0.0026
	$A(\bar{x})$	0.8939	0.8968	0.9215	0.9283
	$\pm STD$	± 0.0002	± 0.0001	$\pm$ 0.0092	± 0.0038
	Epochs	100	330	32	5
Prox-LISA					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$	0.0003	0.0004	0.0004	0.0090
	$\pm STD$	± 0.0001	$\pm 3.84e^{-5}$	$\pm$ 2.99e^{-5}	$\pm 7.15e^{-5}$
	$A(\bar{x})$	0.9004	0.9000	0.9209	0.9355
	$\pm STD$	± 0.0008	± 0.0004	± 0.0003	± 0.0006
	Epochs	68	225	483	24
Prox-FB					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$	0.1178	0.0599	0.0184	0.0532
	$A(\bar{x})$	0.8111	0.8941	0.9049	0.9257
	Epochs	52	240	915	321

Table 5: Results for NN-L_1 test problem after 15 seconds of runtime.

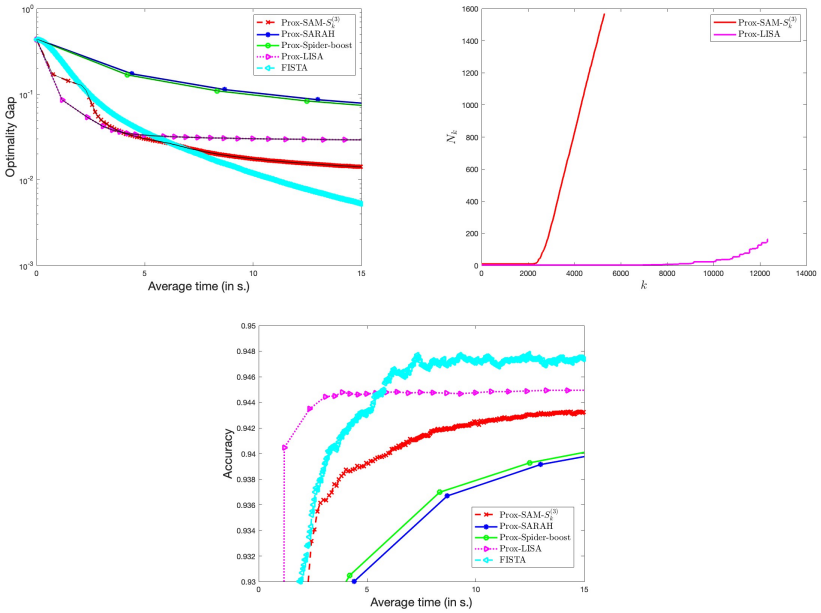

Fig. 5: Behaviour of the methods in a time budget of 15 seconds for the test problem LR-L_1 and the *RCV1* dataset - First row: averaged optimality gap computed on the training set (left panel) and increase of the averaged mini-batch size (right panel). Second row: averaged accuracy evaluated on the test set.

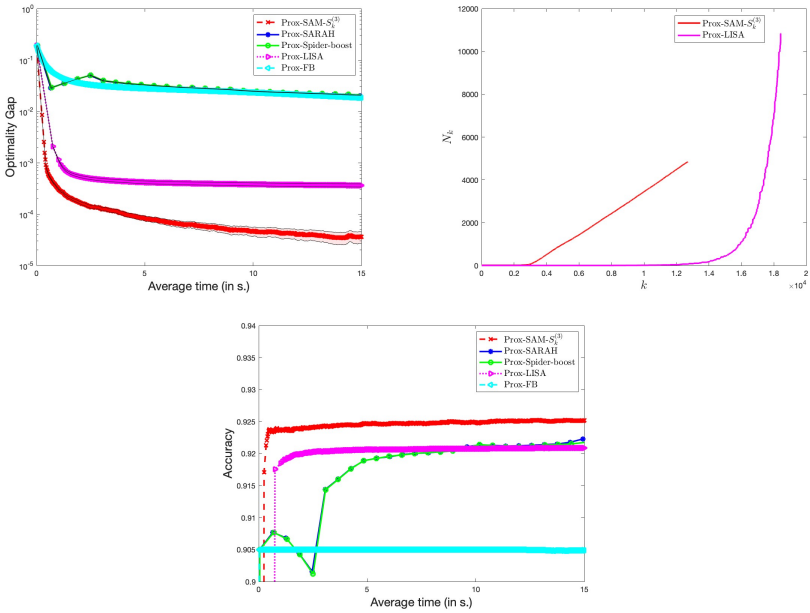

Fig. 6: Behaviour of the methods in a time budget of 15 seconds for the test problem NN-L_1 and the *IJCNN* dataset - First row: averaged optimality gap computed on the training set (left panel) and increase of the averaged mini-batch size (right panel). Second row: averaged accuracy evaluated on the test set.

3.2 L_2-regularized test problems

In this section we present the results obtained for test problems with regularization term equal to $\frac{\lambda}{2}\|x\|_2^2$. Specifically, the first experiment compares the last four versions of **Prox-SAM** introduced in Section 3.1 with the **LSNM-BB** algorithm [41]. We remark that while **Prox-SAM** exploits the closed form of the proximal operator related to the squared L_2 norm, **LSNM-BB** does not include a proximal step to handle the regularization term but relies on the use of the gradient of the entire objective function.
As in Section 3.1, the methods are stopped when the total number of evaluations of the loss terms and their gradients is greater than or equal to $N \cdot maxit$ and $maxit$ is set equal to 20.
In Tables 6 and 7 we report the averaged optimality gap and the averaged accuracy obtained by the **LSNM-BB** algorithm and the different versions of **Prox-SAM** for the two test problems LR-L_2 and NN-L_2, respectively. In general, the non-scaled version of **Prox-SAM** is the least efficient, while the scaled versions perform efficiently across all metrics compared to **LSNM-BB**.

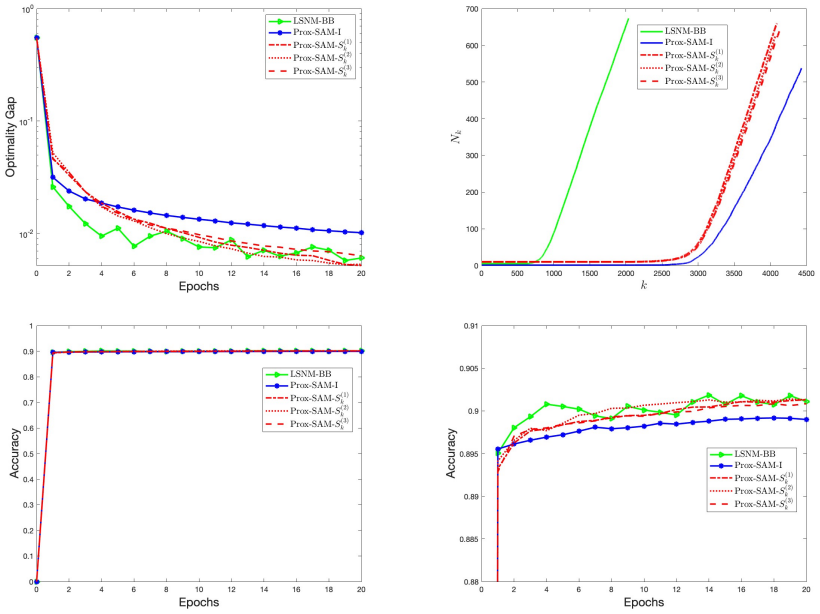

Fig. 7: LR-L_2 test problem for the *w8a* dataset - First row: averaged optimality gap computed on the training set (left panel) and increase of the averaged mini-batch size (right panel). Second row: averaged accuracy evaluated on the test set (left panel) and a zoom of the accuracy in the interval $[0.88, 0.91]$ (right panel).

Method		*MNIST*	*w8a*	*IJCNN*	*RCV1*
LSNM-BB					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0343 $\pm$0.0217	0.0061 $\pm$0.0015	0.0053 $\pm$0.0014	0.0321 $\pm$0.0074
	$A(\bar{x})$ $\pm STD$	0.8859 $\pm$0.0094	0.9011 $\pm$0.0016	0.9158 $\pm$0.0018	0.9493 $\pm$0.0018
	Time (s)	5.7553	1.6268	0.6155	2.9684
Prox-SAM-I					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0994 $\pm$0.0107	0.0102 $\pm$0.0014	0.0017 $\pm$0.0003	0.0330 $\pm$0.0009
	$A(\bar{x})$ $\pm STD$	0.8851 $\pm$0.0027	0.8990 $\pm$0.0017	0.9146 $\pm$0.0012	0.9427 $\pm$0.0019
	Time (s)	5.7899	1.5858	0.6360	4.1773
Prox-SAM-$S_k^{(1)}$					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0086 $\pm$0.0009	0.0052 $\pm$0.0010	0.0008 $\pm$0.0004	0.0080 $\pm$0.0005
	$A(\bar{x})$ $\pm STD$	0.8969 $\pm$0.0020	0.9013 $\pm$0.0011	0.9165 $\pm$0.0007	0.9548 $\pm$0.0008
	Time (s)	6.2626	1.7721	0.6997	4.5637
Prox-SAM-$S_k^{(2)}$					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0098 $\pm$0.0009	0.0053 $\pm$0.0012	0.0009 $\pm$0.0005	0.0090 $\pm$0.0013
	$A(\bar{x})$ $\pm STD$	0.8964 $\pm$0.0017	0.9013 $\pm$0.0007	0.9162 $\pm$0.0005	0.9543 $\pm$0.0010
	Time (s)	6.0336	1.6865	0.6982	4.5370
Prox-SAM-$S_k^{(3)}$					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0109 $\pm$0.0020	0.0063 $\pm$0.0011	0.0009 $\pm$0.0004	0.0104 $\pm$0.0019
	$A(\bar{x})$ $\pm STD$	0.8957 $\pm$0.0027	0.9008 $\pm$0.0009	0.9160 $\pm$0.0007	0.9536 $\pm$0.0016
	Time (s)	5.9906	1.6770	0.7032	5.0806

Table 6: Results obtained for the test problem LR-L_2 with different versions of **Prox-SAM** and **LSNM-BB**.

In Figure 7 the averaged optimality gap, the averaged accuracy and the averaged increase of the mini-batch size for the test problem LR-L_2 with the *w8a* dataset are shown. We can observe that the three scaled versions of **Prox-SAM** (S_k different to the identity matrix) are very efficient. The increase of the mini-batch size is very similar for all the versions of **Prox-SAM**. The final accuracy is around 0.9 in all cases. Figure 8 shows the results obtained for the

Method		*MNIST*	*w8a*	*IJCNN*	*RCV1*
LSNM-BB					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0106 $\pm$0.0052	0.0014 $\pm$0.0005	0.0019 $\pm$0.0013	0.0097 $\pm$0.0030
	$A(\bar{x})$ $\pm STD$	0.8879 $\pm$0.0056	0.9011 $\pm$0.0010	0.9148 $\pm$0.0023	0.9487 $\pm$0.0027
	Time (s)	5.4880	1.7148	0.5530	2.8257
Prox-SAM-I					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0096 $\pm$0.0008	0.0024 $\pm$0.0004	0.0015 $\pm$0.0005	0.0130 $\pm$0.0003
	$A(\bar{x})$ $\pm STD$	0.8943 $\pm$0.0016	0.8985 $\pm$0.0013	0.9113 $\pm$0.0017	0.9412 $\pm$0.0018
	Time (s)	5.4716	1.8529	0.6698	3.8381
Prox-SAM-$S_k^{(1)}$					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0028 $\pm$0.0006	0.0022 $\pm$0.0005	0.0001 $\pm$ 4.64e^{-5}	0.0019 $\pm$0.0002
	$A(\bar{x})$ $\pm STD$	0.8990 $\pm$0.0019	0.8992 $\pm$0.0015	0.9166 $\pm$0.0004	0.9542 $\pm$0.0014
	Time (s)	5.8628	1.8786	0.7428	4.1673
Prox-SAM-$S_k^{(2)}$					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0032 $\pm$0.0010	0.0026 $\pm$0.0007	0.0002 $\pm$ 6.89e^{-5}	0.0025 $\pm$0.0006
	$A(\bar{x})$ $\pm STD$	0.8987 $\pm$0.0022	0.8984 $\pm$0.0016	0.9165 $\pm$0.0005	0.9530 $\pm$0.0016
	Time (s)	5.6565	1.9515	0.7504	4.2096
Prox-SAM-$S_k^{(3)}$					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H^*_{\mathcal{N}}\rvert$ $\pm STD$	0.0029 $\pm$0.0008	0.0028 $\pm$0.0004	0.0001 $\pm$ 2.98e^{-5}	0.0025 $\pm$0.0005
	$A(\bar{x})$ $\pm STD$	0.9001 $\pm$0.0025	0.8978 $\pm$0.0011	0.9165 $\pm$0.0004	0.9537 $\pm$0.0011
	Time (s)	5.7120	1.9208	0.7739	4.0003

Table 7: Results obtained for the test problem NN-L_2 with different versions of **Prox-SAM** and **LSNM-BB**.

NN-L_2 objective function with the *MNIST* dataset. Similar considerations to those made for Figure 7 can be deduced.

3.2.1 Comparison of Prox-SAM-$S_k^{(3)}$ with other methods

As in Section 3.1.2 for the L_1 norm, we now discuss the results of a numerical comparison between **Prox-SAM-**$S_k^{(3)}$ and several state-of-the-art methods in

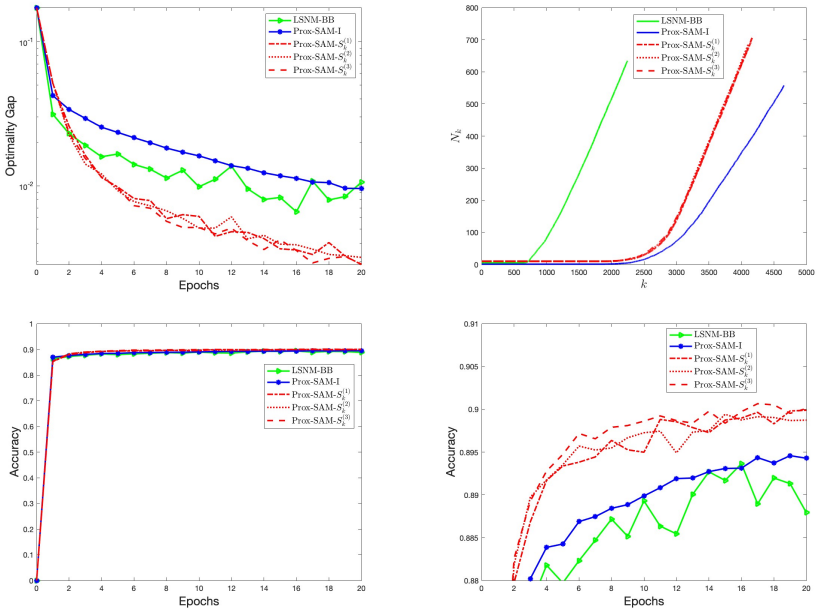

Fig. 8: Test problem NN-L_2 for the *MNIST* dataset - First row: averaged optimality gap computed on the training set (left panel) and increase of the averaged mini-batch size (right panel). Second row: averaged accuracy evaluated on the test set (left panel) anda zoom of the accuracy in the interval $[0.88, 0.91]$ (right panel).

both deterministic and stochastic settings. In particular, in the deterministic context, **FISTA** was considered for the LR-L_2 test problem and **Prox-FB** for the NN-L_2 test problem. In the stochastic framework, in addition to **Prox-SARAH**, **Prox-Spider-boost** and **Prox-LISA** methods, the **ADAGRAD** algorithm [46] was also considered. As for the other methods, the hyperparameter values are shown in the Appendix B. Indeed, since the entire objective function for both LR-L_2 and NN-L_2 is differentiable, the convergence of the method is guaranteed [53]. We recall that the key feature of the **ADAGRAD** method is its rescaling of each coordinate based on the sum of squared past stochastic gradients. However, the accumulation of these past gradients significantly limits its ability to adapt to local variations in function smoothness.

In Tables 8 and 9 we report the averaged optimality gap and the averaged accuracy obtained after 15 seconds of runtime by the considered methods for the two test problems LR-L_2 and NN-L_2, respectively. We report also the number of epochs processed within 15-seconds time budget. In general, **Prox-SAM-**$S_k^{(3)}$ appears to be more efficient than deterministic schemes, particularly during the initial phase of execution, as confirmed by the scientific literature on stochastic methods. Considering the number of epochs processed,

Method		*MNIST*	*w8a*	*IJCNN*	*RCV1*
Prox-SAM-$S_k^{(3)}$					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$ $\pm STD$	0.0053 $\pm$0.0004	0.0012 $\pm$0.0002	$4.75e^{-5}$ $\pm 1.49e^{-5}$	0.0009 $\pm 7.18e^{-5}$
	$A(\bar{x})$ $\pm STD$	0.8978 $\pm$0.0018	0.9034 $\pm$0.0011	0.9162 $\pm$0.0001	0.9573 $\pm$0.0006
	Epochs	47	175	914	236
Prox-SARAH					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$ $\pm STD$	0.0061 $\pm$ 0.0001	0.0036 $\pm$ 0.0001	0.0001 $\pm$ 1.45e^{-5}	0.0298 $\pm$ 0.0006
	$A(\bar{x})$ $\pm STD$	0.8980 $\pm$0.0005	0.9004 $\pm$0.0005	0.9157 $\pm$ 5.21e^{-5}	0.9461 $\pm$0.0007
	Epochs	33	58	29	5
Prox-Spider-boost					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$ $\pm STD$	0.0051 $\pm$ 0.0001	0.0128 $\pm$ 1.02e^{-5}	0.0002 $\pm$ 1.74e^{-5}	0.0279 $\pm$ 0.0006
	$A(\bar{x})$ $\pm STD$	0.8985 $\pm$0.0004	0.8985 $\pm$0.0000	0.9157 $\pm$ 3.75e^{-5}	0.9464 $\pm$0.0006
	Epochs	38	284	28	5
Prox-LISA					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$ $\pm STD$	0.0013 $\pm$0.0004	0.0001 $\pm$ 1.84e^{-5}	$2.14e^{-6}$ $\pm$ 6.89e^{-7}	0.0020 $\pm 3.97e^{-5}$
	$A(\bar{x})$ $\pm STD$	0.8990 $\pm$0.0008	0.9038 $\pm$0.0002	0.9164 $\pm 4.38e^{-5}$	0.9561 $\pm$0.0007
	Epochs	76	218	430	24
ADAGRAD					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$ $\pm STD$	0.0029 $\pm$ 3.94e^{-5}	0.0032 $\pm$ 1.18e^{-5}	0.0197 $\pm$ 7.83e^{-5}	0.0217 $\pm 3.62e^{-5}$
	$A(\bar{x})$ $\pm STD$	0.8988 $\pm$0.0005	0.8999 $\pm 6.36e^{-5}$	0.9076 $\pm 6.49e^{-5}$	0.9516 $\pm$0.0004
	Epochs	82	249	200	77
FISTA					
	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$ $A(\bar{x})$	0.1064 0.8431	0.0324 0.8949	$9.06e^{-7}$ 0.9164	$5.62e^{-5}$ 0.9573
	Epochs	53	221	970	373

Table 8: Results for LR-L_2 test problem after 15 seconds of runtime.

Prox-SAM-$S_k^{(3)}$, **Prox-LISA** and **ADAGRAD** require shorter execution compared to **Prox-SARAH** and **Prox-Spider-boost**. We also observe that, even in the presence of the L_2 regularization term, **Prox-SAM-$S_k^{(3)}$** provides more accurate results than **Prox-SARAH** and **Prox-Spider-boost**. The results of **ADAGRAD** are fairly aligned with those of **Prox-SAM-$S_k^{(3)}$**; on the other hand, **ADAGRAD** also performs a scaling of the stochastic gradient; however, the scheme requires the differentiability of the objective function

Method		*MNIST*	*w8a*	*IJCNN*	*RCV1*
Prox-SAM-$S_k^{(3)}$	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$ $\pm STD$	0.0012 $\pm$0.0004	0.0009 $\pm$0.0003	$3.44e^{-5}$ $\pm 9.13e^{-6}$	0.0002 $\pm 2.49e^{-5}$
	$A(\bar{x})$ $\pm STD$	0.9014 $\pm$0.0009	0.9005 $\pm$0.0006	0.9167 $\pm$0.0002	0.9555 $\pm$0.0003
	Epochs	52	158	106	299
Prox-SARAH	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$ $\pm STD$	0.0028 $\pm$ 7.19e^{-5}	0.0014 $\pm$ 0.0003	$8.96e^{-11}$ $\pm$ 1.86e^{-10}	0.0470 $\pm$ 0.0016
	$A(\bar{x})$ $\pm STD$	0.8992 $\pm$0.0006	0.8987 $\pm$0.0007	0.9172 $\pm$ 1.17e^{-16}	0.9432 $\pm$0.0009
	Epochs	38	62	33	5
Prox-Spider-boost	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$ $\pm STD$	0.0063 $\pm$ 1.23e^{-5}	0.0038 $\pm$ 3.72e^{-6}	$8.20e^{-11}$ $\pm$ 1.51e^{-10}	0.0525 $\pm$ 0.0019
	$A(\bar{x})$ $\pm STD$	0.8940 $\pm$0.0001	0.8973 $\pm$0.0000	0.9172 $\pm$ 1.17e^{-16}	0.9423 $\pm$0.0009
	Epochs	102	293	33	7
Prox-LISA	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$ $\pm STD$	0.0003 $\pm 3.46e^{-5}$	$8.57e^{-5}$ $\pm$ 1.80e^{-5}	$1.36e^{-6}$ $\pm$ 4.72e^{-7}	0.0006 $\pm 1.36e^{-5}$
	$A(\bar{x})$ $\pm STD$	0.9022 $\pm$0.0009	0.9007 $\pm$0.0002	0.9171 $\pm 6.82e^{-5}$	0.9538 $\pm$0.0006
	Epochs	68	210	472	24
ADAGRAD	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$ $\pm STD$	0.0009 $\pm$ 2.40e^{-5}	0.0002 $\pm$ 4.68e^{-6}	0.0024 $\pm$ 9.87e^{-6}	0.0022 $\pm 7.44e^{-6}$
	$A(\bar{x})$ $\pm STD$	0.9017 $\pm$0.0004	0.9003 $\pm 6.36e^{-5}$	0.9107 $\pm 5.68e^{-5}$	0.9531 $\pm$0.0005
	Epochs	84	248	216	82
Prox-FB	$\lvert H_{\mathcal{N}}(\bar{x}) - H_{\mathcal{N}}^*\rvert$ $A(\bar{x})$	0.1243 0.8110	0.0677 0.8947	0.0119 0.9049	0.0328 0.9351
	Epochs	52	219	902	345

Table 9: Results for NN-L_2 test problem after 15 seconds of runtime.

- an assumption that can be avoided by incorporating the proximal step in **Prox-SAM-$S_k^{(3)}$**. Finally, compared to **Prox-LISA**, the behaviour of **Prox-SAM-$S_k^{(3)}$** appears similar and, for certain test problem/dataset combinations, even more efficient.

In Figure 9 the averaged optimality gap, the averaged accuracy and the averaged increase of the mini-batch size for the test problem LR-L_2 with the *IJCNN* dataset are shown. We observe that, after few epochs, **Prox-SAM-$S_k^{(3)}$** provides a rapid decrease in the optimality gap compared to the other methods.

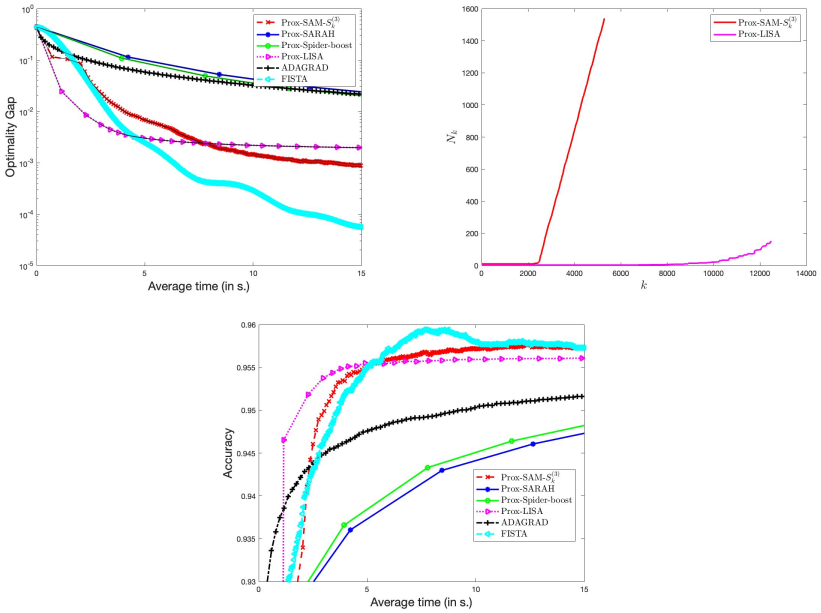

Fig. 9: Behaviour of the methods in a time budget of 15 seconds for the test problem LR-L_2 and the *RCV1* dataset - First row: averaged optimality gap computed on the training set (left panel) and increase of the averaged mini-batch size (right panel). Second row: averaged accuracy evaluated on the test set.

The final accuracy is very similar across all methods, and although the mini-batch size increase for **Prox-SAM-**$S_k^{(3)}$ is larger than for **Prox-LISA**, it remains smaller than the size of the training set. Similar observations can be made for Figure 10, which shows the same metrics for the *RCV1* dataset with the NN-L_2 objective function.

4 Numerical experiments on nonlinear regression

As a further numerical application, we focus on a non-convex minimization problem arising in the context of nonlinear regression. Consider a training set $\{(a_i, b_i)\}_{i=1}^N$, where $a_i \in \mathbb{R}^d$ represents the feature vector and $b_i \in \mathbb{R}$ is the target variable for the i-th example. Our goal is to solve a problem of the form

$$\min_{x\in\mathbb{R}^d} H_{\mathcal{N}}(x) := \frac{1}{N}\sum_{i=1}^{N}(b_i - h(a_i;\, x))^2 + R(x)$$

where $h(\cdot;\, x) : \mathbb{R}^d \longrightarrow \mathbb{R}$ is a nonlinear prediction function.

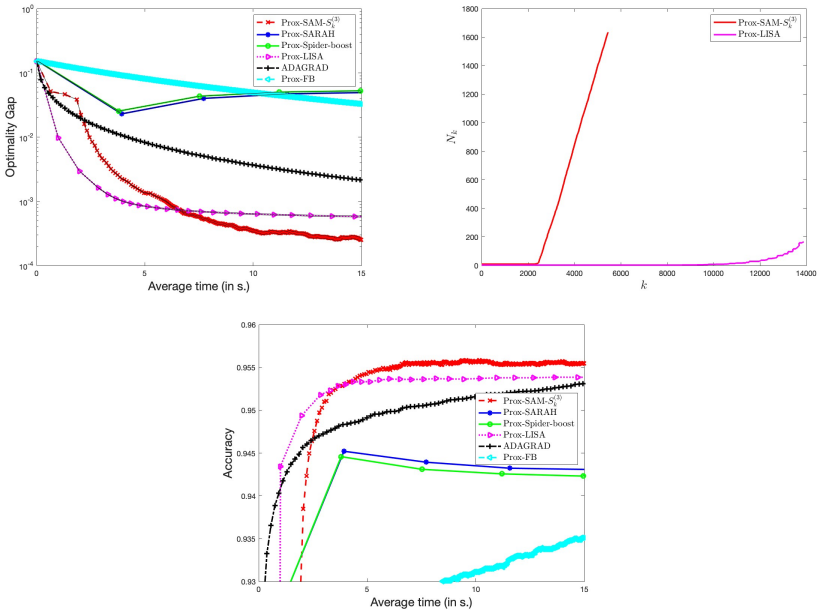

Fig. 10: Behaviour of the methods in a time budget of 15 seconds for the test problem NN-L_2 and the *RCV1* dataset - First row: averaged optimality gap computed on the training set (left panel) and increase of the averaged mini-batch size (right panel). Second row: averaged accuracy evaluated on the test set.

Here we use the *AIR* dataset [54], which includes 9358 instances of hourly averaged pollutant concentrations along with temperature and air humidity (both relative and absolute) recorded each hour between March 2004 and February 2005 at a device located in a polluted area of an Italian city. The task, as in [55], is to predict the benzene concentration based on seven input features: carbon monoxide, nitrogen oxides, ozone, non-methane hydrocarbons, nitrogen dioxide, air temperature, and relative air humidity. Following [56], the prediction function $h(\cdot, x)$ has been modeled using a $7 \times 5 \times 1$ feed-forward neural network, where the two hidden layers use a linear activation function, and the output layer uses a sigmoid activation function. Similarly to the numerical experiments on binary classification, we report the results obtained by considering either $R(x) = \lambda \|x\|_1$ (L_1) or $R(x) = \frac{\lambda}{2}\|x\|_2^2$ (L_2). In both cases, $\lambda = 10^{-4}$.

According to [56], instances with missing benzene concentration values were removed from the dataset, reducing its size from 9358 to 8991 records. For the training phase, we used $N = 6294$ samples, with the remaining 2697 used for testing. Since the concentration values are recorded hourly, the training data covers the first nine months, while the testing data corresponds to the last three months. Lastly, all data values were normalized to the interval $[0, 1]$.

To evaluate the effectiveness of **Prox-SAM-**$S_k^{(3)}$, the nonlinear regression test problem is solved by comparing the behaviour of the following methods:

- **Prox-SAM-**$S_k^{(3)}$ with the same hyperparameters setting detailed in Section 3;
- **Prox-LISA** with the same hyperparameters setting detailed in Section 3;
- **Prox-SARAH** with $\gamma = 0.99$, $\alpha = 0.01$, $\overline{N} = 1$ and $m = 2N$;
- **Prox-Spider-boost** with $\alpha = 0.01$, $\overline{N} = 1$ and $m = 2N$;
- **Prox-FB** with $\alpha_0 = 10$.

For each numerical test, we performed 10 runs, each with a time budget of 15 seconds. In Table 10 we report the averaged value of the loss function over the training set, the averaged value of the loss function over the testing set and the averaged number of epochs processed by the methods on both the L_1 and L_2 regularized problems. In terms of loss function reduction, the performance

Method	L_1			L_2		
	training loss ± STD	testing loss ± STD	epoch	training loss ± STD	testing loss ± STD	epoch
Prox-SAM-$S_k^{(3)}$	$0.0032 \pm 5.89e^{-5}$	0.0045 ± 0.0002	161	$0.0028 \pm 1.32e^{-5}$	0.0036 ± 0.0002	195
Prox-LISA	$0.0031 \pm 1.95e^{-5}$	0.0046 ± 0.0002	128	$0.0028 \pm 3.03e^{-6}$	0.0039 ± 0.0002	129
Prox-SARAH	$0.0044 \pm 5.59e^{-5}$	$0.0061 \pm 5.30e^{-5}$	47	$0.0030 \pm 2.17e^{-5}$	$0.0043 \pm 5.97e^{-5}$	45
Prox-Spider-boost	$0.0043 \pm 4.05e^{-5}$	$0.0061 \pm 4.22e^{-5}$	45	$0.0030 \pm 2.74e^{-6}$	$0.0043 \pm 5.69e^{-5}$	48
Prox-FB	0.0053	0.0088	69	0.0035	0.0069	76

Table 10: Results after 15 seconds of runtime for the nonlinear regression problem equipped with L_1 or L_2 regularization term.

of **Prox-SAM-**$S_k^{(3)}$ is comparable to that of **Prox-LISA** and slightly outperforms both the hybrid methods and **Prox-FB**. Notably, **Prox-SAM-**$S_k^{(3)}$ is the algorithm that can process the highest number of epochs within the given time budget. Figures 11 and 12 show the results of the nonlinear regression problem with L_1 and L_2 regularization terms, respectively. Specifically, for all the compared methods, we report the decrease in the averaged values of the loss functions on both the training and testing sets, as well as the averaged increase in the mini-batch size. For each plot, the standard deviation is also displayed and appears limited in all cases. From these figures, we can conclude that **Prox-SAM-**$S_k^{(3)}$ performs efficiently in reducing the loss function, with a limited increase in the mini-batch size.

Finally, Figure 13 shows the benzene concentration estimates provided by the algorithms against the true concentration (black line). All the algorithms achieve similar results.

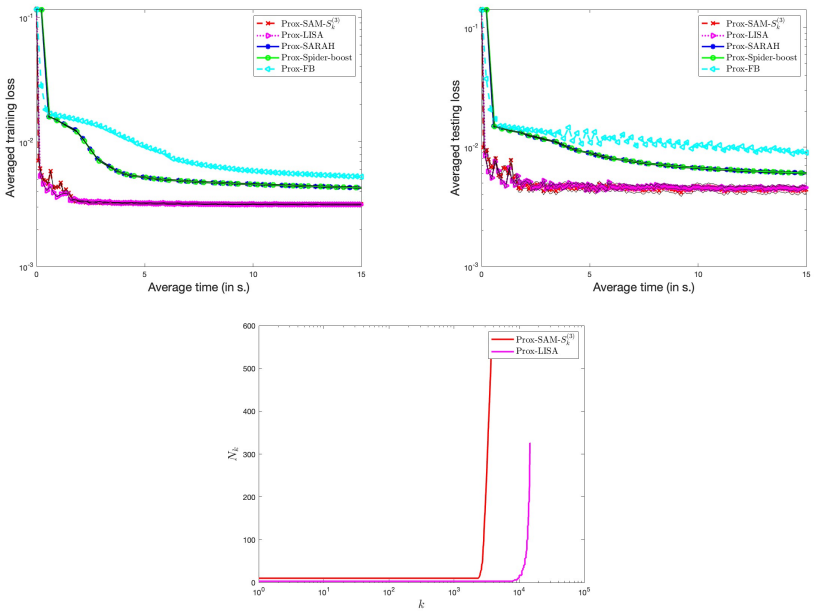

Fig. 11: Results for the nonlinear regression with L_1 regularization term - First row: averaged loss computed over the training set (left panel) and averaged loss computed over the testing set (right panel). Second row: increase of the averaged mini-batch size.

5 Conclusions

In this paper, we proposed a class of variable metric proximal stochastic gradient methods aimed at solving regularized empirical risk minimization problems. Besides the presence of a scaling matrix in the stochastic direction, our proposal is based on a line search, monitoring the decrease of the stochastic approximations of the objective function, and an increasing mini-batch size strategy, combined with an additional sampling procedure. Specifically, the mini-batch remains fixed until the additional sampling condition is no longer satisfied or a predefined number of iterations has been reached. We have studied the convergence properties of our proposed scheme for strongly convex, convex, and non-convex functions, notably without requiring Lipschitz continuity of the gradient for the differentiable part of the objective function.
Numerical experiments on binary classification and nonlinear regression tasks validate the effectiveness of our approach, showcasing its promising performance relative to existing state-of-the-art proximal stochastic gradient methods, without requiring the expensive selection of hyperparameters needed to achieve a efficient behaviour. Future work will focus on extending the approach to more complex tasks, such as multi-class classification and deep learning models.

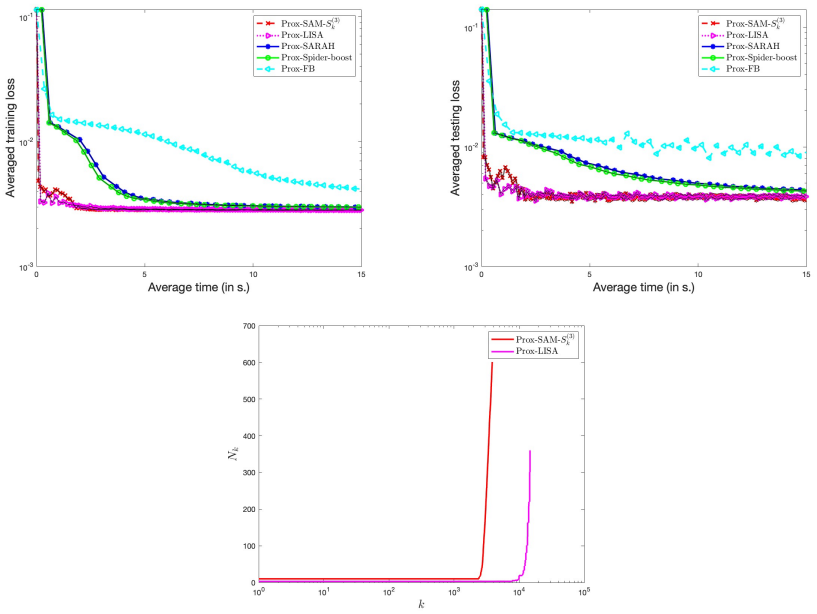

Fig. 12: Results for the nonlinear regression with L_2 regularization term - First row: averaged loss computed over the training set (left panel) and averaged loss computed over the testing set (right panel). Second row: increase of the averaged mini-batch size.

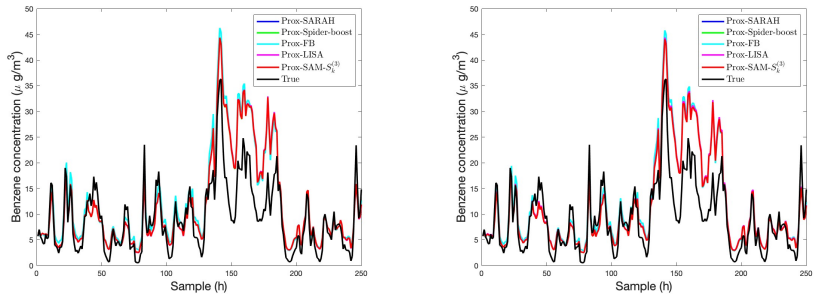

Fig. 13: Estimated benzene concentrations during 10 days (240 hours) obtained by solving the nonlinear regression problem with L_1 regularized problem (left panel) and the L_2 regularized problem (right panel) using the compared methods; the black line represents the true concentration.

Acknowledgments

This work was partially supported by "Gruppo Nazionale per il Calcolo Scientifico (GNCS-INdAM)" (Progetti 2024).

F. Porta is partially supported by the the Italian MUR through the PRIN 2022 Project "Numerical Optimization with Adaptive Accuracy and Applications to Machine Learning", project code: 2022N3ZNAX (CUP E53D2300 7700006), under the National Recovery and Resilience Plan (PNRR), Italy, Mission 04 Component 2 Investment 1.1 funded by the European Commission - NextGeneration EU programme.
F. Porta is partially supported by the the Italian MUR through the PRIN 2022 PNRR Project "Advanced optimization METhods for automated central veIn Sign detection in multiple sclerosis from magneTic resonAnce imaging (AMETISTA)", project code: P2022J9SNP (CUP E53D23017980001), under the National Recovery and Resilience Plan (PNRR), Italy, Mission 04 Component 2 Investment 1.1 funded by the European Commission - NextGeneration EU programme.
V. Ruggiero and I. Trombini were partially funded by European Union - NextGenerationEU through the Italian Ministry of University and Research as part of the PNRR – Mission 4 Component 2, Investment 1.3 (MUR Directorial Decree no. 341 of 03/15/2022), FAIR "Future" Partnership Artificial Intelligence Research", Proposal Code PE00000013 - CUP J33C22002830006).
Krklec Jerinkić is partially supported by the Science Fund of the Republic of Serbia, Grant no. 7359, Project LASCADO and partially by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia (Grants No. 451-03-137/2025-03/ 200125 & 451-03-136/2025-03/ 200125).

Availability of Data and Materials

The datasets analyzed during the current study are available in links given in the paper.

Conflict of interest

The authors have no relevant financial or non-financial interests to disclose.

References

[1] Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning: Data mining, inference, and prediction. Springer New York (2009)

[2] Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. SIAM Review **60**(2), 223–311 (2018)

[3] Bottou, L.: Online learning and stochastic approximations. Cambridge University Press, New York, NY, USA, 9–42 (1998)

[4] Goodfellow, I., Bengio, Y., Courville, A.: Deep learning, volume 1. MIT press Cambridge (2016)

[5] Sra, S., Nowozin, S., Wright, S.J.: Optimization for machine learning. Mit Press (2012)

[6] Bertero, M., Boccacci, P.: Introduction to inverse problems in imaging. Institute of Physics, Bristol, England (1998)

[7] Bertero, M., Boccacci, P., Ruggiero, V.: Inverse imaging with poisson data. Institute of Physics, Bristol, England (2018). https://doi.org/10.1088/2053-2563/aae109

[8] Combettes, P.L., Wajs, V.R.: Signal recovery by proximal forward-backward splitting. SIAM Multiscale Model. Simul. **4**, 1168–1200 (2005)

[9] Combettes, P.L., Pesquet, J.-C.: Proximal splitting methods in signal processing. In: Bauschke, H.H., Burachik, R.S., Combettes, P.L., Elser, V., Luke, D.R., Wolkowicz, H. (eds.) Fixed-point Algorithms for Inverse Problems in Science and Engineering. Springer Optimization and Its Applications, pp. 185–212. Springer, ??? (2011)

[10] Attouch, H., Bolte, J., Svaiter, B.F.: Convergence of descent methods for semi–algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods. Math. Program. **137**, 91–129 (2013)

[11] Bonettini, S., Porta, F., Prato, M., Rebegoldi, S., Ruggiero, V., Zanni, L.: Recent advances in variable metric first-order methods. In: M. Donatelli, S.S.-C. (ed.) Computational Methods for Inverse Problems in Imaging vol. 36, pp. 1–31. Springer, ??? (2019)

[12] Bonettini, S., Loris, I., Porta, F., Prato, M.: Variable metric inexact line-search based methods for nonsmooth optimization. SIAM Journal on Optimization **26**, 891–921 (2016)

[13] Chouzenoux, E., Pesquet, J.C., Repetti, A.: Variable metric forward-backward algorithm for minimizing the sum of a differentiable function and a convex function. J. Optim. Theory Appl. **162**, 107–132 (2014)

[14] Combettes, P.L., Vũ, B.: Variable metric quasi-féjer monotonicity. Nonlinear Anal. **78**, 17–31 (2013)

[15] Frankel, P., Garrigos, G., Peypouquet, J.: Splitting methods with variable metric for Kurdyka-Lojasiewicz functions and general convergence rates. J. Optim. Theory Appl. **165**, 874–900 (2015)

[16] Salzo, S.: The variable metric forward-backward splitting algorithm under mild differentiability assumptions. SIAM J. Optim. **27**(4), 2153–2181 (2017)

[17] Khaled, A., Sebbouh, O., Loizou, N., Gower, R.M., Richtárik, P.: Unified analysis of stochastic gradient methods for composite convex and smooth optimization. J. Optim. Theory Appl. **199**, 499–540 (2023)

[18] Duchi, J., Singer, Y.: Efficient online and batch learning using forward backward splitting. J. Mach. Learn. Res. **10**, 2899–2934 (2009)

[19] Bollapragada, R., Byrd, R., Nocedal, J.: Adaptive sampling strategies for stochastic optimization. SIAM Journal on Optimization **28**(4), 3312–3343 (2018)

[20] Richard H Byrd, R.H., Chin, G.M., Nocedal, J., Wu, Y.: Sample size selection in optimization methods for machine learning. Mathematical Programming **134**(1), 127–155 (2012)

[21] Cartis, C., Scheinberg, K.: Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. Mathematical Programming, 1–39 (2015)

[22] Franchini, G., Porta, F., Ruggiero, V., Trombini, I.: A line search based proximal stochastic gradient algorithm with dynamical variance reduction. Journal of Scientific Computing **94**, 23 (2023)

[23] Franchini, G., Porta, F., Ruggiero, V., Trombini, I., Zanni, L.: Learning rate selection in stochastic gradient methods based on line search strategies. Applied Mathematics in Science and Engineering **31**(1), 2164000 (2023)

[24] Friedlander, M.P., Schmidt, M.: Hybrid deterministic-stochastic methods for data fitting. SIAM Journal on Scientific Computing **34**(3), 1380–1405 (2012)

[25] Hashemi, F.H., Ghosh, S., Pasupathy, R.: On adaptive sampling rules for stochastic recursions. Simulation Conference (WSC), 2014 Winter, 3959–3970 (2014)

[26] Poon, C., Liang, J., Schoenlieb, C.: Local convergence properties of SAGA/Prox-SVRG and acceleration. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 4124–4132 (2018)

[27] Xiao, L., Zhang, T.: A proximal stochastic gradient method with progressive variance reduction. SIAM J. Optim. **24**(4), 2057–2075 (2014)

[28] Phamy, N.H., Nguyen, L.M., Phan, D.T., Tran–Dinh, Q.: Proxsarah: An effcient algorithmic framework for stochastic composite nonconvex optimization. Journal of Machine Learning Research **21**, 1–48 (2020)

[29] Wang, Z., Ji, K., Zhou, Y., Liang, Y., Tarokh, V.: Spiderboost and momentum: Faster stochastic variance reduction algorithms. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems, Curran Associates Inc., vol. 216, pp. 2406–2416 (2019)

[30] Fort, G., Moulines, E.: Stochastic variable metric proximal gradient with variance reduction for non-convex composite optimization. Stat. Comput. **33**, 65 (2023)

[31] Defazio, A., Bach, F., Lacoste-Julien, S.: Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In: Advances in Neural Information Processing Systems, pp. 1646–1654 (2014)

[32] Johnson, R., Zhang, T.: Accelerating stochastic gradient descent using predictive variance reduction. In: Advances in Neural Information Processing Systems, pp. 315–323 (2013)

[33] Bandeira, A.S., Scheinberg, K., Vicente, L.N.: Convergence of trust-region methods based on probabilistic models. SIAM J. Optim. **24**(3), 1238–1264 (2014)

[34] Blanchet, J., Cartis, C., Menickelly, M., Scheinberg, K.: Convergence rate analysis of a stochastic trust region method via submartingales. INFORMS J. Optim. **1**, 92–119 (2019)

[35] Chen, R., Menickelly, M., Scheinberg, K.: Stochastic optimization using a trust-region method and random models. Math. Program. **169**(2), 447–487 (2018)

[36] Xiaoyu, W., Yuan, Y.X.: Stochastic trust region methods with trust region radius depending on probabilistic models. J. Comput. Math. **40**(2), 294–334 (2022)

[37] Franchini, G., Porta, F., Ruggiero, V., Trombini, I., Zanni, L.: A stochastic gradient method with variance control and variable learning rate for deep learning. Journal of Computational and Applied Mathematics **451**, 116083 (2024)

[38] Paquette, C., Scheinberg, K.: A stochastic line search method with expected complexity analysis. SIAM Journal on Optimization **30**(1), 349–376 (2020)

[39] Vaswani, S., Mishkin, A., Laradji, I., Schmidt, M., Gidel, G., Lacoste-Julien, S.: Painless stochastic gradient; interpolation, line–search, and convergence rates. Advances in Neural Information Processing Systems

32, 3312–3343 (2018)

[40] di Serafino, D., Krejić, N., Jerinkić, N.K., Viola, M.: LSOS: Line-search second-order stochastic optimization methods for nonconvex finite sums. Mathematics of Computation **92**(341), 1273–1299 (2023)

[41] Krklec Jerinkić, N., Trombini, I., Ruggiero, V.: Spectral stochastic gradient method with additional sampling for finite and infinite sums. Comput Optim Appl (2025)

[42] Bellavia, S., Krejić, N., Jerinkić, N.K., Raydan, M.: Slises: Subsampled line search spectral gradient method for finite sums. Optimization Methods and Software, 1–26 (2024)

[43] Krejić, N., Krklec Jerinkić, N., Martínez, A., Yousefi, M.: A non-monotone trust-region method with noisy oracles and additional sampling. Comput Optim Appl **89**, 247–278 (2024)

[44] Polyak, B.T.: Introduction to optimization. Optimization Software (1987)

[45] Frassoldati, G., Zanghirati, G., Zanni, L.: New adaptive stepsize selections in gradient methods. J. Ind. Manag. Optim. **4**(2), 299–312 (2008)

[46] Duchi, J.C., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research **12**, 2121–2159 (2011)

[47] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR (2015)

[48] Zhuang, J., Tang, T., Ding, Y., Tatikonda, S.C., Dvornek, N., Papademetris, X., Duncan, J.: Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. Adv. Neural Inf. Process. Syst. **33** (2020)

[49] Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal on Imaging Sciences **2**(1), 183–202 (2009). https://doi.org/10.1137/080716542

[50] Chambolle, A., Dossal, C.H.: On the convergence of the iterates of "fista". Journal of Optimization Theory and Applications **166**(3), 25 (2015). https://doi.org/10.1137/080716542

[51] Passty, G.B.: Ergodic convergence to a zero of the sum of monotone operators in hilbert space. Journal of Mathematical Analysis and Applications **72**(1), 383–390 (1979)

[52] Wang, Z., Ji, K., Zhou, Y., Liang, Y., Tarokh, V.: SpiderBoost and

momentum: faster stochastic variance reduction algorithms. Curran Associates Inc., Red Hook, NY, USA (2019)

[53] Défossez, A., Bottou, L., Bach, F., Usunier, N.: A simple convergence proof of adam and adagrad. Transactions on Machine Learning Research (2022)

[54] Lichman, M.: UCI machine learning repository (2013). https://archive.ics.uci.edu/ml/index.php

[55] De Vito, S., Massera, E., Piga, M., Martinotto, L., Di Francia, G.: On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. Sens. Actuator B **129**, 750–757 (2008)

[56] Bellavia, S., Krejić, N., Morini, B., Rebegoldi, S.: A stochastic first-order trust-region method with inexact restoration for finite-sum minimization. Comput Optim Appl **84**, 53–84 (2023)

A Auxiliary results

A.1 Properties of the proximal operator

Lemma A.1 recalls well known results on the proximal operator (for the proof, see [9, 12] and references therein).

Lemma A.1. *Let $\alpha > 0$, S be a symmetric positive definite matrix, $x \in dom(H_\Sigma)$ with $\Sigma \subseteq \mathcal{N}$. Given the function*

$$q_\Sigma^{\alpha,S}(y) = (y-x)^T \nabla f_\Sigma(x) + \frac{1}{2\alpha}\|y-x\|_S^2 + R(y) - R(x),$$

the following statements hold true.

a. $\hat{y} = \mathrm{prox}_{\alpha R}^S(x - \alpha S^{-1}u)$ *if and only if* $\frac{1}{\alpha}S(x-\hat{y}) - u = w$, $w \in \partial R(\hat{y})$.
b. $q_\Sigma^{\alpha,S}(x) = 0$.
c. Given $v^{(\Sigma)} = \mathrm{prox}_{\alpha R}^S(x - \alpha S^{-1}\nabla f_\Sigma(x))$, $q_\Sigma^{\alpha,S}(v^{(\Sigma)}) \leq 0$ *and* $q_\Sigma^{\alpha,S}(v^{(\Sigma)}) = 0$ *if and only if* $v^{(\Sigma)} = x$.
d. x *is a stationary point for problem* $min_{y\in\mathbb{R}^d} f_\Sigma(y) + R(y)$ *if and only if* $x = v^{(\Sigma)}$ *if and only if* $q_\Sigma^{\alpha,S}(v^{(\Sigma)}) = 0$.

A.2 Proof of Lemma 2.1

Assume by contradiction that there exists a $k \in \mathbb{N}$ such that Step 4 in Algorithm 1 performs an infinite number of reductions. As a consequence, for any

$j \in \mathbb{N}$ we have

$$\begin{aligned}
\eta q_{\mathcal{N}_k}^{\alpha_k,S_k}(v_k^{(\mathcal{N}_k)}) &< \frac{H_{\mathcal{N}_k}(x_k + \beta^j d_k^{(\mathcal{N}_k)}) - H_{\mathcal{N}_k}(x_k)}{\beta^j} = \\
&= \frac{f_{\mathcal{N}_k}(x_k + \beta^j d_k^{(\mathcal{N}_k)}) - f_{\mathcal{N}_k}(x_k)}{\beta^j} + \frac{R(x_k + \beta^j d_k^{(\mathcal{N}_k)}) - R(x_k)}{\beta^j} \\
&\leq \frac{f_{\mathcal{N}_k}(x_k + \beta^j d_k^{(\mathcal{N}_k)}) - f_{\mathcal{N}_k}(x_k)}{\beta^j} + \\
&\quad + \frac{\beta^j R(x_k + d_k^{(\mathcal{N}_k)}) + (1-\beta^j)R(x_k) - R(x_k)}{\beta^j} \\
&= \frac{f_{\mathcal{N}_k}(x_k + \beta^j d_k^{(\mathcal{N}_k)}) - f_{\mathcal{N}_k}(x_k)}{\beta^j} + R(v_k^{(\mathcal{N}_k)}) - R(x_k),
\end{aligned}$$

where the second inequality follows by applying the convexity of function R. Taking the limit on the right-hand side for $j \to +\infty$, we obtain

$$\begin{aligned}
\eta q_{\mathcal{N}_k}^{\alpha_k,S_k}(v_k^{(\mathcal{N}_k)}) &\leq \nabla f_{\mathcal{N}_k}(x_k)^T d_k^{(\mathcal{N}_k)} + R(v_k^{(\mathcal{N}_k)}) - R(x_k) \\
&\leq \nabla f_{\mathcal{N}_k}(x_k)^T d_k^{(\mathcal{N}_k)} + R(v_k^{(\mathcal{N}_k)}) - R(x_k) + \frac{1}{2\alpha_k}\|v_k^{(\mathcal{N}_k)} - x_k\|_{S_k}^2 \\
&= q_{\mathcal{N}_k}^{\alpha_k,S_k}(v_k^{(\mathcal{N}_k)}).
\end{aligned}$$

Since $0 < \eta < 1$ and the line search is performed only if $q_{\mathcal{N}_k}^{\alpha_k,S_k}(v_k^{(\mathcal{N}_k)})$ is non-zero, this is an absurd.

A.3 Proof of Lemma 2.2

Assume that $N_k < N$ for all $k \in \mathbb{N}$. Since the sample size sequence $\{N_k\}$ in Algorithm 1 is non-decreasing, this means that there exists some $\overline{N} < N$ and $k_2 \in \mathbb{N}$ such that $N_k = \overline{N}$ for all $k \geq k_2$. Now, let us assume that there is no $k_1 \in \mathbb{N}$ such that $\mathcal{D}_k^- = \emptyset$ for all $k \geq k_1$. This means that there exists an infinite sub-sequence of iterations $K \subseteq \mathbb{N}$ such that $\mathcal{D}_k^- \neq \emptyset$ for all $k \in K$. Since $\mathcal{D}_k$ is chosen randomly and uniformly, with finitely many possible outcomes for each k, there exists some $q > 0$ such that $\mathbb{P}(\mathcal{D}_k \in \mathcal{D}_k^-) \geq q$ for all $k \in K$. So, we have

$$\mathbb{P}(\mathcal{D}_k \in \mathcal{D}_k^+, k \in K) \leq \Pi_{k \in K}(1-q) = 0;$$

this means that we will almost surely encounter an iteration at which the sample size will be increased due to violation of the additional sampling condition in Step 6 of Algorithm 1. This is a contradiction with the sample size being kept to $\overline{N}$ during the whole optimization process. Thus, we conclude that the statement holds.

B Hyperparameter settings for the compared methods

For the **Prox-SARAH** method we use the hyperparameter setting specified in [28] where, by borrowing the notation of the referred paper, $q = 2+0.01+(\frac{1}{100})$, $C = \frac{q^2}{(q^2+8)\hat{L}^2\gamma^2}$ and the values for the other hyperparameters are shown in Table 11.

For the **Prox-Spider-boost** method we use the hyperparameter setting specified in [52] and the values for hyperparameters are shown in Table 12.

Dataset	Loss	γ	α	$\overline{N}$	m
MNIST	LR	0.99	α_{opt}	1	2N
MNIST	NN	0.99	$\frac{0.1}{\hat{L}}$	1	2N
w8a	LR	0.99	$\frac{0.1}{\hat{L}}$	1	2N
w8a	NN	0.99	$\frac{0.1}{\hat{L}}$	1	2N
IJCNN	LR	0.99	α_{opt}	1	2N
IJCNN	NN	0.99	α_{opt}	1	2N
RCV1	LR	0.95	α_{opt}	1	2N
RCV1	NN	0.95	α_{opt}	1	2N

Table 11: Hyperparameter settings for Prox-SARAH [28].

α_{opt} is the best tuned value obtained after a time and resource consuming procedure of repeated trials. This setting is the same for both the regularization terms.

Dataset	Loss	α	$\overline{N}$	m
MNIST	LR	α_{opt}	1	2N
MNIST	NN	0.05	256	$\frac{2N}{256}$
w8a	LR	0.05	256	$\frac{2N}{256}$
w8a	NN	0.05	256	$\frac{2N}{256}$
IJCNN	LR	α_{opt}	1	2N
IJCNN	NN	α_{opt}	1	2N
RCV1	LR	α_{opt}	1	2N
RCV1	NN	α_{opt}	1	2N

Table 12: Hyperparameter settings for Prox-Spider-boost [52].

In the implementation of **Prox-LISA** method, the initial mini batch size is $N_0 = 3$ and the line search hyperparameter is $\beta = \frac{1}{2}$ in all experiments. The attempt value for the step length at Step 2 is, in general, $\alpha_0 = 1$ for the first iteration and $\alpha_k = \min(\alpha_0, \alpha_{k-1}\frac{1}{\beta})$ for the following iterations. Furthermore, we have the rule $\varepsilon_k = 100 \cdot 0.999^k$ for controlling the variance.

In the implementation of **ADAGRAD** method, the hyperparameters are set with the most commonly used values [46]; specifically, the learning rate α is 0.01, the mini-batch size is fixed to 50, and $\varepsilon = 10^{-8}$.