sciendo

CAIM
ISSN 2038-0909

Commun. Appl. Ind. Math. 11 (1), 2020, 19–34 DOI: 10.2478/caim-2020-0002

Research Article

Subsampled Nonmonotone Spectral Gradient Methods

Stefania Bellavia[2*], Nataša Krklec Jerinkić[3], Greta Malaspina[3]

[2]Department of Industrial Engineering, University of Florence, Italy.
Member of the INdAM Research Group GNCS.

[3]Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Serbia

[*]Email address for correspondence: stefania.bellavia@unifi.it

Communicated by Michele Benzi

Received on 12 10, 2018. Accepted on 11 13, 2019.

Abstract

This paper deals with subsampled spectral gradient methods for minimizing finite sums. Subsample function and gradient approximations are employed in order to reduce the overall computational cost of the classical spectral gradient methods. The global convergence is enforced by a nonmonotone line search procedure. Global convergence is proved provided that functions and gradients are approximated with increasing accuracy. R-linear convergence and worst-case iteration complexity is investigated in case of strongly convex objective function. Numerical results on well known binary classification problems are given to show the effectiveness of this framework and analyze the effect of different spectral coefficient approximations arising from the variable sample nature of this procedure.

Keywords: `spectral gradient methods, subsampling strategies, global convergence, nonmonotone line search.`

AMS subject classification: `49M37, 65K05, 68T05, 68W40`

1. Introduction

The aim of this work is to present a class of first-order iterative methods for optimization problems where the objective function is given as the mean of a large number of functions, i.e.,

$$\min_{x\in\mathbb{R}^n} f_{\mathcal{N}}(x), \qquad f_{\mathcal{N}}(x) = \frac{1}{N}\sum_{j=1}^{N} f_j(x). \tag{1}$$

The functions $f_j : \mathbb{R}^n \to R$ are assumed to be continuously differentiable for $j = 1, \ldots, N$ and $\mathcal{N}$ denotes the set of indices $\{1, \ldots, N\}$. The motivation for studying problems of this form comes from machine learning. Indeed, the training phase of a neural network requires to solve problems of the form (1) where the number N of functions is generally large enough to prevent the employment of classical optimization methods, thus leading to the necessity of developing different strategies. One possible approach is to employ a so-called mini-batch or subsampling strategy [1–16]. That is, an iterative optimization method is applied but instead of considering the whole sum (1), at the beginning of every iteration functions and/or gradient and/or Hessian are approximated using only a subset of the functions $\{f_1, \ldots, f_N\}$.

In this paper we focus on variable sample variants of spectral gradient methods with nonmonotone line search. Spectral gradient methods, originally proposed in [17] for the solution of unconstrained optimization problems have been widely used and developed (see [18–22] and references therein). In these approaches the steplength is adaptively chosen and they showed very good practical performance. Here, we combine the spectral gradient method with a nonmonotone line search and a variable sample strategy with a twofold aim: retain robustness and adaptive steplength choice of spectral gradient methods and reduce the overall computational cost of solving (1) by approximating functions and gradients with increasing accuracy.

© 2020 Stefania Bellavia,Nataša Krklec Jerinkić,Greta Malaspina, licensee De Gruyter Open.
This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 License.

Given the sample size, the subsample is randomly generated from $\{1, \dots, N\}$ and we consider two kinds of subsampling. As we will see, different options arise for the subsampling approximation of the relevant quantities characterizing this class of methods. These options give rise to four variants of the subsampling spectral gradient method with line search and we analyze the global convergence of the unifying framework they belong to. We remark that, thanks to the globalization strategy, the methods show global convergence and do not require to choose the steplength by trial, which can be time consuming in practice. Focusing on the strongly convex case we prove R-linear convergence of the generated sequence to the minimizer of (1). We also provide iteration complexity of the methods and show that the complexity bound of the corresponding exact method is retained, despite inaccuracy in functions and gradients.

Finally, we present some numerical results on binary classification problems, showing the effectiveness of our approach.

Spectral gradient methods for problem (1) have been investigated in [15,16]. In [15] they are used in combination with the stochastic gradient method, while in [16] a mini-batch strategy is employed. In both papers convergence is proved assuming to employ the full gradient every m iterations (at each outer iteration).

We differ from this latter approach as we embed the variable sample spectral method in a nonmonotone line search strategy using approximate functions and gradients. In the convergence analysis we have also to take into account inaccuracy in function values and not only in gradient, while in [15,16] only inaccuracy in gradients needs to be handled as the function values are unused. On the other hand, the employment of the line search procedure allow us to obtain global convergence to a stationary point irrespectively of convexity of the objective function, differently from [15,16]. Nonconvex problems arise in neural networks training, so this relaxation can be of great interest.

Taking different samples and/or sample sizes in different iterations affects the spectral coefficients, so, besides the convergence analysis, the main goal of this paper is to investigate the performance of different choices of spectral coefficient calculations as they affect the cost per iteration as well. Notice that in [15] the spectral coefficient is computed at the outer iterations using the full gradient and it is taken constant along the inner iterations. In [16] it is updated also in the inner iterations using a subsample of the dataset whose dimension depends on the condition number of the problem. In [23] spectral gradient methods for problems with objective function given in form of mathematical expectation have been analyzed, however the effect of the choice of the spectral coefficient has not been investigated.

This paper is structured as follows. In Section 1 we recall the classical nonmonotone spectral gradient method, introduce the subsampling strategy and embed it in the framework of the classical method. In Section 2 we study the convergence behavior of the obtained subsampling procedure providing convergence and complexity results. In Section 3 we report the results of numerical tests we carried out. We focus on the comparison between the subsampled method and the classical counterpart and on the influence of some critical parameters over the performance of the methods.

2. The Method

In this section we describe the subsampled spectral gradient framework we are involved with. Spectral gradient methods are first-order iterative procedures employing the gradient vector as a search direction. A crucial role is played by the steplength that is chosen in such a way to inject some second-order information into the methods.

At a generic iteration, given the current point x_k, the new iterate is computed as

$$x_{k+1} = x_k + \alpha_k d_k$$

where

$$d_k = -\sigma_{k-1}^{-1} \nabla f_{\mathcal{N}}(x_k) \tag{2}$$

and α_k is chosen through a line search strategy. The scalar σ_k is called *spectral coefficient*.

Here we adopt the classical Barzilai-Borwein choice given in [17], i.e.

$$\sigma_k = \begin{cases} \dfrac{s_k^t y_k}{s_k^t s_k} & \text{if } \dfrac{s_k^t y_k}{s_k^t s_k} \in [\sigma_m, \sigma_M] \\ 1 & \text{otherwise} \end{cases}$$

with

$$s_k = x_{k+1} - x_k$$
$$y_k = \nabla f_{\mathcal{N}}(x_{k+1}) - \nabla f_{\mathcal{N}}(x_k)$$

and $0 < \sigma_m < 1 < \sigma_M < +\infty$ given safeguards. Notice that this guaranties that the search direction is a descent direction with respect to the current objective function. Therefore, provided that the function is bounded from below on the line segment $[x_k, x_k + d_k]$, there exists a steplength interval that satisfies Wolfe conditions. In practice, Wolfe conditions are often replaced by the backtracking technique combined with the first Wolfe condition, i.e. the Armijo condition. Nonmonotone line search in general allows even larger step sizes, so the line search remains well defined.

Specifically, we here assume that the steplength α_k satisfies the Li-Fukushima [24] nonmonotone descent condition and the Wolfe condition, i.e. it satisfies the following two inequalities:

$$\begin{aligned} f_{\mathcal{N}}(x_k + \alpha_k d_k) &\leq f_{\mathcal{N}}(x_k) + c_1 \alpha_k \nabla f_{\mathcal{N}}(x_k)^t d_k + \zeta_k \\ \nabla f_{\mathcal{N}}(x_k + \alpha_k d_k)^t d_k &\geq c_2 \nabla f_{\mathcal{N}}(x_k)^t d_k \end{aligned} \tag{3}$$

where $0 < c_1 \leq c_2 < 1$, $\zeta_k \geq 0$ such that $\sum_{k\geq 0} \zeta_k < +\infty$.

The following algorithm summarizes a generic iteration of the spectral gradient method we are considering.

Algorithm 2.1. *(k-th iteration of spectral gradient method)*
Input: $x_k \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$, $\zeta_k > 0$, $0 < \sigma_m < 1 < \sigma_M$

set $g_k = \nabla f_{\mathcal{N}}(x_k)$
set $s_{k-1} = x_k - x_{k-1}$
set $y_{k-1} = g_k - g_{k-1}$
set $\sigma_{k-1} = \dfrac{s_{k-1}^t y_{k-1}}{s_{k-1}^t s_{k-1}}$
if $\sigma_{k-1} \notin [\sigma_m, \sigma_M]$
  set $\sigma_{k-1} = 1$
end if
set $d_k = -\sigma_{k-1}^{-1} g_k$
compute α_k *such that* (3) *holds*
set $x_{k+1} = x_k + \alpha_k d_k$

In order to reduce the overall computational cost of the procedure, at each iteration we choose a subsample $\mathcal{N}_k \subseteq \{1, \ldots, N\}$ of size N_k and we employ a variable sample strategy. That is, we approximate the objective function and its gradient as follows:

$$f_{\mathcal{N}_k}(x) := \frac{1}{N_k} \sum_{j \in \mathcal{N}_k} f_j(x)$$

$$\nabla f_{\mathcal{N}_k}(x) := \frac{1}{N_k} \sum_{j \in \mathcal{N}_k} \nabla f_j(x).$$

Given an increasing sequence $\{N_k\}$ of sample sizes, we consider two different kinds of subsampling:

- *nested subsamples* $\mathcal{N}_{k-1} \subseteq \mathcal{N}_k$;
  we take $\mathcal{N}_k$ as the union of $\mathcal{N}_{k-1}$ and a set of $(N_k - N_{k-1})$ randomly-chosen indices in $\mathcal{N} \setminus \mathcal{N}_{k-1}$;
- *non-nested subsamples* such that $\mathcal{N}_{k-1} \cap \mathcal{N}_k \neq \emptyset$;
  we take one index j_1 randomly chosen in $\mathcal{N}_{k-1}$ to assure a non-empty intersection, then we take $\mathcal{N}_k$ as the union of j_1 and a set of $(N_k - 1)$ randomly chosen indices in $\mathcal{N} \setminus \{j_1\}$. Notice that we enforce non empty intersection between consecutive subsamples, but since we randomly choose the indices in $\mathcal{N} \setminus \{j_1\}$, we don't have any control over the actual size of the intersection.

In the definition (2) of the direction d_k we replace the gradient $\nabla f_{\mathcal{N}}(x_k)$ with $\nabla f_{\mathcal{N}_k}(x_k)$, while for the gradient displacement vector y_{k-1} we have different possible choices that we are now going to present. First, let us assume that we are in the nested case (i.e. $\mathcal{N}_{k-1} \subseteq \mathcal{N}_k$). Since the subsample that we are considering at the current iteration is $\mathcal{N}_k$, the first option is to replace in the computation of y_k both $\nabla f_{\mathcal{N}}(x_k)$ and $\nabla f_{\mathcal{N}}(x_{k-1})$ with the approximation given by the current subsample, getting

$$y_{k-1}^{(1)} := \nabla f_{\mathcal{N}_k}(x_k) - \nabla f_{\mathcal{N}_k}(x_{k-1}). \tag{4}$$

On the other hand we already approximated $\nabla f_{\mathcal{N}}(x_{k-1})$ at the previous iteration as $\nabla f_{\mathcal{N}_{k-1}}(x_{k-1})$, therefore the second option is to use

$$y_{k-1}^{(2)} := \nabla f_{\mathcal{N}_k}(x_k) - \nabla f_{\mathcal{N}_{k-1}}(x_{k-1}). \tag{5}$$

Assuming we are in the non-nested case, the definition (4) is still possible, and denoting with $\mathcal{I}_k$ the intersection of the current and the previous subsample we can also take

$$y_{k-1}^{(3)} := \nabla f_{\mathcal{I}_k}(x_k) - \nabla f_{\mathcal{I}_k}(x_{k-1}). \tag{6}$$

Each of these choices allows us to exploit a different amount of information in the approximation of y_{k-1} and therefore of the spectral coefficient σ_{k-1}, but it also requires a different amount of computation in terms of number of component gradient evaluations. We already noticed that at every iteration we need to evaluate $\nabla f_{\mathcal{N}_k}(x_k)$, then at iteration $k-1$ we have evaluated $\nabla f_j(x_{k-1})$ for every $j \in \mathcal{N}_{k-1}$. In the nested case one can store only the previous (average) gradient and the previous sample size and the computation of $y_{k-1}^{(2)}$ does not require extra computation, while the computation of $y_{k-1}^{(1)}$ requires $(N_k - N_{k-1})$ new evaluations of component gradients. In the non-nested case, the storage of all the component gradients from the previous iteration is needed to compute $y_{k-1}^{(1)}$ and $y_{k-1}^{(3)}$. The evaluation of $y_{k-1}^{(1)}$ also requires $(N_k - |\mathcal{I}_k|)$ evaluations of the component gradients that have not been already evaluated at the previous iteration. If the storage of all the component gradients is not feasible, one can use a fixed mini-batch for the intersection and use it only for obtaining the spectral coefficient.

The employment of the subsampling scheme to the line search conditions (3) is immediate and leads to:

$$f_{\mathcal{N}_k}(x_k + \alpha_k d_k) \leq f_{\mathcal{N}_k}(x_k) + c_1 \alpha_k \nabla f_{\mathcal{N}_k}(x_k)^t d_k + \zeta_k \tag{7}$$

$$\nabla f_{\mathcal{N}_k}(x_k + \alpha_k d_k)^t d_k \geq c_2 \nabla f_{\mathcal{N}_k}(x_k)^t d_k \tag{8}$$

with all the requests over c_1, c_2 and $\{\zeta_k\}$ unchanged.
In Table 1 we summarize all the possible combinations for the definitions of the vector y_k together with the two kinds of subsampling, then in Algorithm 2.2 we present the general structure of the k-th iteration of these methods. The input variable *NM* denotes the name of the method, according to Table 1.

Algorithm 2.2. *(Framework of variable sample spectral methods, k-th iteration)*
Input: $x_k \in \mathbb{R}^n$, $\{f_j\}_{j \in \mathcal{N}}$, N_0 *initial size,* NM, $\zeta_k > 0$, $0 < \sigma_m < 1 < \sigma_M$

set $N_k \geq N_{k-1}$
generate $\mathcal{N}_k$ *according to* NM *and the second column of Table 1*
set $g_k = \nabla f_{\mathcal{N}_k}(x_k)$
set $s_{k-1} = x_k - x_{k-1}$
compute y_{k-1} *according to* NM *and the third column of Table 1*
set $\sigma_{k-1} = \dfrac{s_{k-1}^t y_{k-1}}{s_{k-1}^t s_{k-1}}$
if $\sigma_{k-1} \notin [\sigma_m, \sigma_M]$
  set $\sigma_{k-1} = 1$
end if
compute $d_k = -\sigma_{k-1}^{-1} g_k$
compute α_k *such that* (7) *and* (8) *hold*
set $x_{k+1} = x_k + \alpha_k d_k$

Table 1. Subsampled spectral methods

Name	Subsample	y_k
SG_N_1	Nested	$y_k^{(1)}$ given in (4)
SG_N_2	Nested	$y_k^{(2)}$ given in (5)
SG_I_1	Non-Nested	$y_k^{(1)}$ given in (4)
SG_I_3	Non-Nested	$y_k^{(3)}$ given in (6)

3. Global Convergence

We assume that the objective function $f_{\mathcal{N}}$ is bounded from below and continuously differentiable in $\mathbb{R}^n$, and that each gradient ∇f_j is Lipschitz-continuous. We define the errors of approximation ν_k and η_k as follows:

$$\nu_k := \max\{|f_{\mathcal{N}_k}(x_k) - f_{\mathcal{N}}(x_k)|, |f_{\mathcal{N}_k}(x_{k+1}) - f_{\mathcal{N}}(x_{k+1})|\} \tag{9}$$

$$\eta_k := \max\left\{\left|\|\nabla f_{\mathcal{N}}(x_k)\|^2 - \|\nabla f_{\mathcal{N}_k}(x_k)\|^2\right|, \left|\|\nabla f_{\mathcal{N}}(x_{k+1})\|^2 - \|\nabla f_{\mathcal{N}_k}(x_{k+1})\|^2\right|\right\}. \tag{10}$$

We prove here that any limit point of the sequence generated by the method is a stationary point of (1) and that when the objective function is strongly convex, R-linear convergence to the solution holds.

Theorem 3.1. *Let* $\{x_k\}$ *be the sequence of the iterates computed by Algorithm 2.2 with* $\sum_{k\geq 0}\zeta_k = \bar{\zeta} < +\infty$. *Let us assume that* $\sum_{k\geq 0}\nu_k = \bar{\nu} < +\infty$ *and that* η_k *goes to 0 as* k *goes to* $+\infty$. *If* $f_{\mathcal{N}}$ *is bounded from below over* $\mathbb{R}^n$ *and continuously-differentiable, and the gradients* ∇f_j *are Lipschitz-continuous with Lipschitz constant* L, *then:*

$$\lim_{k\to+\infty} \|\nabla f_{\mathcal{N}}(x_k)\| = 0.$$

Proof. *Subtracting* $\nabla f_{\mathcal{N}_k}(x_k)^t d_k$ *from both sides of* (8) *and using the Lipschitz-continuity of the gradient we get*

$$(c_2 - 1)\nabla f_{\mathcal{N}_k}(x_k)^t d_k \leq (\nabla f_{\mathcal{N}_k}(x_k + \alpha_k d_k) - \nabla f_{\mathcal{N}_k}(x_k))^t d_k \leq$$
$$\leq \|\nabla f_{\mathcal{N}_k}(x_k + \alpha_k d_k) - \nabla f_{\mathcal{N}_k}(x_k))\|\|d_k\| \leq L\alpha_k\|d_k\|^2.$$

By the definition of d_k

$$\nabla f_{\mathcal{N}_k}(x_k)^t d_k = -\sigma_{k-1}^{-1}\|\nabla f_{\mathcal{N}_k}(x_k)\|^2 \tag{11}$$

hence, from the previous inequality,

$$\alpha_k \geq -\frac{\sigma_{k-1}(c_2-1)}{L}.$$

Let us define $C := -\frac{c_1(c_2-1)}{L}$*, from* (7)*,* (11) *and the last inequality we have*

$$\begin{aligned} f_{\mathcal{N}_k}(x_{k+1}) &= f_{\mathcal{N}_k}(x_k + \alpha_k d_k) \\ &\leq f_{\mathcal{N}_k}(x_k) + \frac{c_1(c_2-1)}{L}\|\nabla f_{\mathcal{N}_k}(x_k)\|^2 + \zeta_k \\ &\leq f_{\mathcal{N}_k}(x_k) - C\|\nabla f_{\mathcal{N}_k}(x_k)\|^2 + \zeta_k. \end{aligned} \tag{12}$$

From this inequality and (9) *we have*

$$\begin{aligned} f_{\mathcal{N}}(x_{k+1}) &\leq f_{\mathcal{N}_k}(x_{k+1}) + \nu_k \\ &\leq f_{\mathcal{N}_k}(x_k) - C\|\nabla f_{\mathcal{N}_k}(x_k)\|^2 + \zeta_k + \nu_k \\ &\leq f_{\mathcal{N}}(x_k) - C\|\nabla f_{\mathcal{N}_k}(x_k)\|^2 + \zeta_k + 2\nu_k \\ &\leq f_{\mathcal{N}}(x_0) - C\sum_{j=0}^{k}\|\nabla f_{\mathcal{N}_j}(x_j)\|^2 + \sum_{j=0}^{k}(\zeta_j + 2\nu_j), \end{aligned} \tag{13}$$

thus

$$\sum_{j=0}^{k}\|\nabla f_{\mathcal{N}_j}(x_j)\|^2 \leq \frac{f_{\mathcal{N}}(x_0) - f_{\mathcal{N}}(x_{k+1})}{C} + \frac{1}{C}\sum_{j=0}^{k}(\zeta_j + 2\nu_j)$$

and taking the limit for $k \to +\infty$ *we get*

$$\sum_{k\geq 0}\|\nabla f_{\mathcal{N}_k}(x_k)\|^2 \leq \frac{f_{\mathcal{N}}(x_0) - \lim_{k\to\infty} f_{\mathcal{N}}(x_{k+1})}{C} + \frac{1}{C}(\bar{\zeta} + 2\bar{\nu}).$$

Since $f_{\mathcal{N}}$ *is bounded from below we get*

$$\sum_{k\geq 0}\|\nabla f_{\mathcal{N}_k}(x_k)\|^2 < +\infty. \tag{14}$$

By (10) *we thus have*

$$\lim_{k\to+\infty}\|\nabla f_{\mathcal{N}}(x_k)\|^2 \leq \lim_{k\to+\infty}\left(\|\nabla f_{\mathcal{N}_k}(x_k)\|^2 + \eta_k\right) = 0$$

and hence the thesis follows.

□

In the next theorem we will show that our procedure needs at most $O(\varepsilon^{-2})$ iterations to provide $\|\nabla f_{\mathcal{N}_k}(x_k)\| \leq \varepsilon$. We note that the worst case iteration complexity is of the same order of that of nonmonotone spectral gradient methods shown in [25].

Theorem 3.2. *Let* $\{x_k\}$ *be the sequence of the iterates computed by Algorithm 2.2 with* $\sum_{k\geq 0}\zeta_k = \bar{\zeta} < +\infty$*. If we assume that* $\sum_{k\geq 0}\nu_k = \bar{\nu} < +\infty$ *and that the gradients* ∇f_j *are Lipschitz-continuous with constant* L*, then for any given* $\varepsilon > 0$ *the generated sequence satisfies*

$$\|\nabla f_{\mathcal{N}_k}(x_k)\| \leq \varepsilon$$

in at most

$$k_\varepsilon = \lceil (f_{\mathcal{N}}(x_0) - f_{\mathcal{N}}(x_*) + \bar{\zeta} + 2\bar{\nu}) C^{-1} \varepsilon^{-2} \rceil$$

iterations, where the constant C is given by $C := -\frac{c_1(c_2-1)}{L}$.

Proof. *Let us denote with k_ε the first iteration such that $\|\nabla f_{\mathcal{N}_{k_\varepsilon}}(x_{k_\varepsilon})\| < \varepsilon$. By (13) we get, for every $k < k_\varepsilon$*

$$f_{\mathcal{N}_k}(x_k) - f_{\mathcal{N}_k}(x_{k+1}) + \zeta_k \geq C \|\nabla f_{\mathcal{N}_k}(x_k)\|^2 \geq C\varepsilon^2 \tag{15}$$

and by (9)

$$f_{\mathcal{N}}(x_k) - f_{\mathcal{N}}(x_{k+1}) + \zeta_k + 2\nu_k \geq C\varepsilon^2. \tag{16}$$

Taking the sum for k from 0 to $k_\varepsilon - 1$ we get

$$\begin{aligned} k_\varepsilon C\varepsilon^2 &\leq \sum_{k=0}^{k_\varepsilon - 1} (f_{\mathcal{N}}(x_k) - f_{\mathcal{N}}(x_{k+1}) + \zeta_k + 2\nu_k) \leq \\ &\leq f_{\mathcal{N}}(x_0) - f_{\mathcal{N}}(x_{k_\varepsilon}) + \bar{\zeta} + 2\bar{\nu} \end{aligned} \tag{17}$$

and hence the thesis. □

In the sequel we will make use of the next two lemmas.

Lemma 3.1. *[7] If $f_{\mathcal{N}}$ is a continuously differentiable function from $\mathbb{R}^n$ to $\mathbb{R}$, strongly convex with constant c, then $f_{\mathcal{N}}$ has an unique minimizer x_* and, for every $x \in \mathbb{R}^n$, the following inequality holds:*

$$2c\,(f_{\mathcal{N}}(x) - f_{\mathcal{N}}(x_*)) \leq \|\nabla f_{\mathcal{N}}(x)\|^2.$$

Lemma 3.2. *[12] If a sequence $\{a_k\}$ converges to 0 R-linearly then, for every $\rho \in (0,1)$ the sequence $\{A_k\}$ given by*

$$A_k := \sum_{j=0}^{k} \rho^j a_{k-j}$$

converges to 0 R-linearly.

We now focus on the strongly convex case. Denoting with x_* the unique minimizer of $f_{\mathcal{N}}$, we prove that the optimality gap $f_{\mathcal{N}}(x_k) - f_{\mathcal{N}}(x_*)$ tends to zero R- linearly, and consequently the method drives the optimality gap below ε, even if approximated gradients and functions are used.

Theorem 3.3. *Let $\{x_k\}$ be the sequence of the iterates computed by Algorithm 2.2. Assume that $f_{\mathcal{N}}$ is a strongly convex function from $\mathbb{R}^n$ to $\mathbb{R}$ and that the gradients ∇f_j are Lipschitz-continuous with constant L. Then:*

(i) There exists a constant $\rho \in (0,1)$ such that for every index k the optimality gap satisfies

$$\begin{aligned} f_{\mathcal{N}}(x_{k+1}) - f_{\mathcal{N}}(x_*) \leq \rho^{k+1} (f_{\mathcal{N}}(x_0) - f_{\mathcal{N}}(x_*)) + \sum_{j=0}^{k} \rho^j \zeta_{k-j} + \\ + \sum_{j=0}^{k} \rho^j (2\nu_{k-j} + C\eta_{k-j}) \end{aligned}$$

where $C = -\frac{c_1(c_2-1)}{L}$.

(ii) If the three sequences $\{\zeta_k\}$, $\{\nu_k\}$, $\{\eta_k\}$ tend to 0 R-linearly as k goes to $+\infty$, then $f_{\mathcal{N}}(x_{k+1}) - f_{\mathcal{N}}(x_*)$ converges to 0 R-linearly.

Proof.
By (13) and (10) we have

$$f_{\mathcal{N}}(x_{k+1}) - f_{\mathcal{N}}(x_*) \leq f_{\mathcal{N}}(x_k) - f_{\mathcal{N}}(x_*) - C\|\nabla f_{\mathcal{N}_k}(x_k)\|^2 + \zeta_k + 2\nu_k \leq f_{\mathcal{N}}(x_k) - f_{\mathcal{N}}(x_*) - C\|\nabla f_{\mathcal{N}}(x_k)\|^2 + \zeta_k + 2\nu_k + C\eta_k. \tag{18}$$

By Lemma 3.1 we have that

$$\|\nabla f_{\mathcal{N}}(x_k)\|^2 \geq \gamma(f_{\mathcal{N}}(x_k) - f_{\mathcal{N}}(x_*))$$

for every $0 < \gamma < \min\{2c, L\}$. Note that $(1 - C\gamma) \in (0, 1)$. Thus, from (18), we get

$$f_{\mathcal{N}}(x_{k+1}) - f_{\mathcal{N}}(x_*) \leq (1 - C\gamma)(f_{\mathcal{N}}(x_k) - f_{\mathcal{N}}(x_*)) + \zeta_k + 2\nu_k + C\eta_k.$$

Iteratively applying this inequality and denoting with ρ the quantity $(1 - C\gamma)$, we get

$$f_{\mathcal{N}}(x_{k+1}) - f_{\mathcal{N}}(x_*) \leq \rho^{k+1}\left(f_{\mathcal{N}}(x_0) - f_{\mathcal{N}}(x_*)\right) + \sum_{j=0}^{k} \rho^j \zeta_{k-j} + \sum_{j=0}^{k} \rho^j (2\nu_{k-j} + C\eta_{k-j})$$

and thus (i) holds.

Let us define $\omega_j := 2\nu_j + C\eta_j$. By the R-linear convergence of $\{\nu_k\}$ and $\{\eta_k\}$ we have that ω_k converges to 0 R-linearly and thus by inequality (i) and Lemma 3.2 we have (ii).

□

Theorem 3.4. *Suppose that assumptions of Theorem 3.3 hold. Then, for every $\varepsilon \in (0, e^{-1})$, there exist $\hat{\rho} \in (0, 1)$ and $Q > 0$ such that Algorithm 2.2 achieves $f_{\mathcal{N}}(x_k) - f_{\mathcal{N}}(x_*) < \varepsilon$ in at most k_ε iterations, where*

$$k_\varepsilon = \left\lceil \frac{\log(f_N(x_0) - f_N(x_*) + Q) + 1}{|\log(\hat{\rho})|} \log(\varepsilon^{-1}) + 1 \right\rceil. \tag{19}$$

Proof. *We follow the proof of Theorem 6 in [25].*
In Theorem 3.3 we proved that the following inequality holds

$$f_{\mathcal{N}}(x_k) - f_{\mathcal{N}}(x_*) \leq \rho^k \left(f_{\mathcal{N}}(x_0) - f_{\mathcal{N}}(x_*)\right) + \sum_{j=0}^{k-1} \rho^j (\zeta_{k-j} + \omega_{k-j}) \tag{20}$$

where $\omega_j = 2\nu_j + C\eta_j$. Moreover, under our assumptions the last sum converges to 0 R-linearly and hence there there exist $\bar{\rho} \in (0, 1)$ and $Q > 0$ such that

$$\sum_{j=0}^{k-1} \rho^j (\zeta_{k-j} + \omega_{k-j}) \leq Q\bar{\rho}^k. \tag{21}$$

Replacing this inequality in (20) and denoting with $\hat{\rho}$ the maximum between ρ and $\bar{\rho}$, we get

$$f_{\mathcal{N}}(x_k) - f_{\mathcal{N}}(x_*) \leq \hat{\rho}^k \left(f_{\mathcal{N}}(x_0) - f_{\mathcal{N}}(x_*) + Q\right). \tag{22}$$

We denote with k_ε the first iteration such that $f_{\mathcal{N}}(x_k) - f_{\mathcal{N}}(x_*) < \varepsilon$, so that by (22) we have

$$\varepsilon < f_{\mathcal{N}}(x_{k_\varepsilon - 1}) - f_{\mathcal{N}}(x_*) \leq \hat{\rho}^{k_\varepsilon - 1} \left(f_{\mathcal{N}}(x_0) - f_{\mathcal{N}}(x_*) + Q \right) \tag{23}$$

and hence,

$$\log(f_{\mathcal{N}}(x_0) - f_{\mathcal{N}}(x_*) + Q) + \log(\varepsilon^{-1}) > -(k_\varepsilon - 1)\log(\hat{\rho}) = (k_\varepsilon - 1)|\log(\hat{\rho})|.$$

Rearranging this expression, since $\log(\varepsilon^{-1}) > 1$ we get

$$k_\varepsilon - 1 < \frac{\log(f_N(x_0) - f_N(x_*) + Q) + 1}{|\log(\hat{\rho})|} \log(\varepsilon^{-1}) \tag{24}$$

and this completes the proof. □

4. Numerical Results

In this section we report on the numerical experimentation we carried out over the methods introduced in Section 2. Our main goals in this section are the following: to asses whether the subsampling procedure provides a reduction in the overall computational cost, and to provide an indication of which could be the best combination of choices for the subsampling schedule and the gradient displacement vector. Then, the subsampled methods will also be compared with the spectral gradient method without subsampling (SGFull).

We considered binary classification problems [7]. Given a dataset made of $\hat{N}$ of pairs $\{(a_j, b_j)\}_{j=1}^{\hat{N}}$ with $a_j \in \mathbb{R}^n$ and labels $b_j \in \{-1, 1\}$, we use the 95% of data as training set and the remaining 5% as validation set. Then, letting $\mathcal{N}$ and $\mathcal{V}$ be the sets of indices of the data in the training and in the validation set, respectively, we define the objective function $f_{\mathcal{N}}$ as

$$f_{\mathcal{N}}(x) = \frac{1}{N} \sum_{j \in \mathcal{T}} f_j(x), \qquad f_j(x) = \log\left(1 + \exp(-b_j a_j^t x)\right) + \lambda \|x\|^2 \tag{25}$$

where N is the cardinality of $\mathcal{N}$ and the regularization parameter λ is set to N^{-1}.

Note that the main cost in the computation of each function f_j is the evaluation of $\exp(-b_j a_j^t x_j)$. Computing the gradient of f_j for a generic index j, we get

$$\nabla f_j(x) = \frac{\exp(-b_j a_j^t x)}{1 + \exp(-b_j a_j^t x)} b_j a_j + 2\lambda x, \tag{26}$$

thus the evaluation of the gradient comes for free from the evaluation of the corresponding function. Relying on this remark, at the beginning of every iteration the values $\exp(-b_j a_j^t x_k)$ are computed and then exploited during the execution to evaluate both the sampled function $f_{\mathcal{N}_k}(x_k)$ and the sampled gradient $\nabla f_{\mathcal{N}_k}(x_k)$.

Given an initial size $N_0 \geq 1$ and a growth factor $\tau > 1$, we set $N_k = \lceil \tau^k N_0 \rceil$ if this quantity is smaller than the full sample size N, and $N_k = N$ otherwise. We employ (7) with $c_1 = 10^{-4}$ and $\zeta_k = 10^2 k^{-1.1}$ and we compute α_k through a backtracking strategy. As a result we have $\alpha_k = \beta^{-\bar{\jmath}}$ where $\beta = 0.5$ and $\bar{\jmath}$ is the smallest positive integer such that (7) is satisfied, provided $\bar{\jmath} \leq 15$. When such a $\bar{\jmath}$ does not exist, if the full subsample has not been reached yet, we set $x_{k+1} = x_k$ and we proceed to the next iteration enlarging the subsample. If we are working with the full sample, we declare failure. We do not explicitly check if Wolfe condition (8) holds at the chosen α_k since the backtracking strategy and the safeguard $\bar{\jmath} \leq 15$ prevent the step size to become too small.

We consider three problems of the form (25), corresponding to the datasets given in Table 2, along with their dimension $\hat{N}$ (number of data points including both training and validation set) and n (dimension of the decision variable).

Table 2. Datasets: number of samples $\hat{N}$, problem dimension n.

Dataset	$\hat{N}$	n
CINA0 [26]	16033	132
MNIST [27]	60000	784
Mushrooms [27]	5000	112

4.1. *Comparison of the Subsampled Methods*

We report the results of the comparison among the performance of all the subsampled methods summarized in Table 1 on the CINA0 dataset. We underline that the reported statistics are representative of the results that we obtained also with the other datasets. We add comparison with the method employing exact functions and gradients (SGFull) for completeness.

We run each method with $\tau = 1.1$, $N_0 = 3$, and we stop the procedure when the norm of the gradient is smaller than 10^{-4}, provided that the full sample is reached. We count the number of iterations performed, the number of scalar products computed for the evaluation of each term $\exp(-b_j a_j^t x_j)$ with $j \in \mathcal{N}_k$, the number of functions evaluations and gradients evaluations. We recall that since the subsamples are chosen randomly the obtained sequence is not deterministic and therefore for each method we perform 100 runs and we report the averages.

In Table 3 we report the obtained results. The averages obtained are divided by the size N of the training set, so that the evaluation of the full gradient and function counts 1. For the number of gradients evaluations, we consider two countings. In column GE_1 we count only the computation of gradients $\nabla f_j(x_k)$ corresponding to functions f_j that have not been previously evaluated at the same point x_k, while in GE_2 we count every gradient evaluations irrespectively to the evaluation of the functions. The motivation behind this choice is the remark that we made at the beginning of this section about the particular form of the gradients for the problems we are considering. Since the evaluation of the gradient through equation (26) does not require additional computation with respect to the corresponding evaluation of the function, we can count only the gradients evaluations involving new scalar products (that is, the gradients required to compute the vector y_k); we choose to report also the full count of the gradients to better understand what would be the computational cost if we could not rely on this property of the gradients, that is in case we consider a sum of functions f_j such that there is not this strict relationship between gradients and functions.
In the table the headers of the columns have the following meaning: IT is the number of iterations performed, SP is the number of scalar products computed, FE is the number of functions evaluations and GE_1 and GE_2 are the gradients evaluations explained above.

Table 3. Statistics of the runs

METHOD	IT	SP	FE	GE_1	GE_2
SG_N_1	101	67.6	66.5	1.1	31.3
SG_N_2	106	80.1	80.1	0	35.9
SG_I_1	109	91.6	85.7	5.7	44.7
SG_I_3	105	93.6	93.6	0	34.7
SGFull	52	115	115	0	52

We can notice from Table 3 that the nested subsampling methods are in general more efficient than their non-nested counterparts and that the definition of y_k that uses the same subset $\mathcal{N}_k$ to approximate the gradient both at x_k and at x_{k-1} seems to perform better than the alternatives, both in the nested and in the non-nested case. In fact, our numerical experience suggests that computing the gradient displacement vector using all the current information, i.e. using the gradient sampled in the same set $\mathcal{N}_k$ where the function and the search direction are sampled, allows to obtain a less expensive method in terms of weighted number of functions evaluations. Then, even if choices (5) and (6) do not require extra scalar products for computing y_{k-1}, the gain obtained is not enough to compensate the larger number of functions evaluations required. Finally, notice that the results indicate that using the subsampled approach is overall more efficient than SGFull.

4.2. *Influence of the Parameter τ*

In this subsection we focus on the role of the growth factor τ. The influence of this parameter over the efficiency of the methods may be high: a bigger value of τ means a more accurate approximation of the objective function and its gradient from the beginning of the algorithm, but it also causes a higher per-iteration cost during the initial phase. We consider the methods SG_N_1 and SG_I_1 which from the previous section appear to be the best among the nested and the non-nested methods, respectively.

In Figure 1 we report the number of scalar products required by the two methods for different values of $\tau \in \{1.1, 1.2, ..., 1.9, 2, 2.25, 2.5, ..., 4.5, 5\}$. For each τ we consider 100 runs of each method on the CINA0 dataset and we take the average number of scalar products computed excluding the worst 20 and the best 20 results, then we divide by N. We compare the overall cost in terms of scalar products with that of SGFull.
In Figure 2 we plot the difference between the smallest and the biggest number of scalar products required among the runs of each of the two methods: this can be taken as a first indication of the variance in the efficiency of the methods corresponding to the chosen τ.

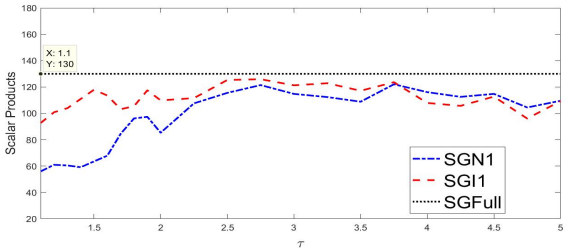

Figure 1.

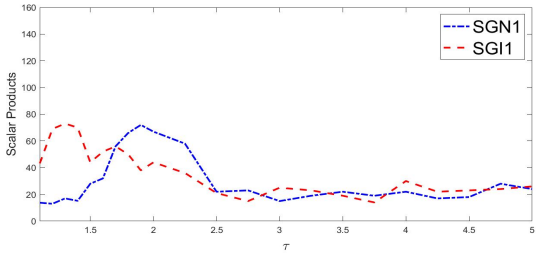

Figure 2.

As we can see from Figure 1, among the considered values of τ the initial choice $\tau = 1.1$ seems to be the optimal one for both the methods and, for small values of the growth factor SG_N_1 seems to be more efficient than SG_I_1. As τ grows the computational cost of the nested method tends to increase (for $\tau > 3.5$ it becomes higher than the cost of SG_I_1). For all the tested values of τ the subsampled methods outperform the SG method with full function and gradient information. The gap is more evident for small values of τ. For larger values the benefits deriving from a more accurate representation of $f_{\mathcal{N}}$ do not seem to be enough to compensate for the bigger average number of per-iteration scalar products required. Figure 2 shows that the cost of the non-nested method seems to be overall less influenced by the choice of the growth factor. We considered also values of τ smaller than 1.1 and we observed that they give rise to procedures overall more expensive than that corresponding to $\tau = 1.1$ as the growth of the data set is too low and starting from $N_0 = 3$ the number of iterations needed to reach a reasonable value of the training loss is too high.

4.3. *Training Error*

We consider the three datasets reported in Table 2 and three methods: SGFull, SG_N_1 and SG_I_1 both with with $\tau = 1.1$. For every method and every dataset we study how the *training error* $f_{\mathcal{N}}(x_k) - f^*_{\mathcal{N}}$ changes as the number of iterations and scalar products performed grows. We stress that the number of performed scalar products can be considered a measure of the overall computational cost due to the form of functions f_j and their derivatives. The optimal value $f^*_{\mathcal{N}}$ has been computed running SGFull with termination condition $\|\nabla f_{\mathcal{N}}(x_k)\| \leq 10^{-7}$.

In Figures 3-5 we plot the training loss versus iterations on the left and versus scalar products on the

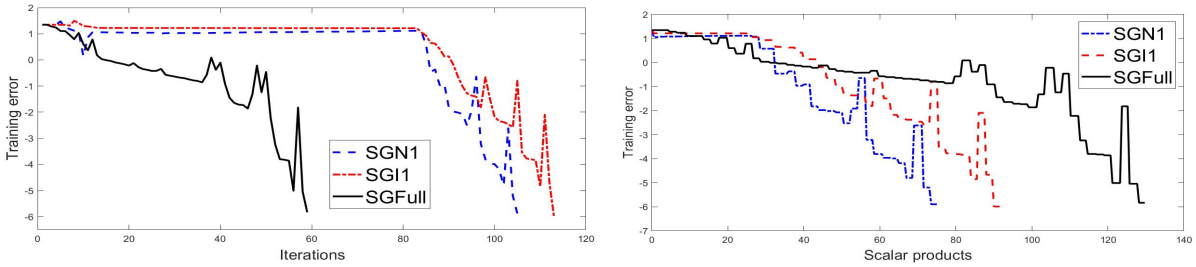

Figure 3. CINA0 dataset, training error versus iterations (left) and versus scalar products (right).

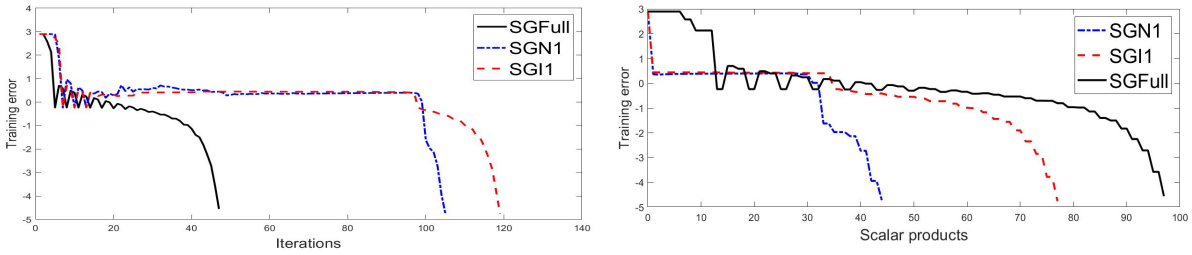

Figure 4. MNIST dataset, training error versus iterations (left) and versus scalar products (right).

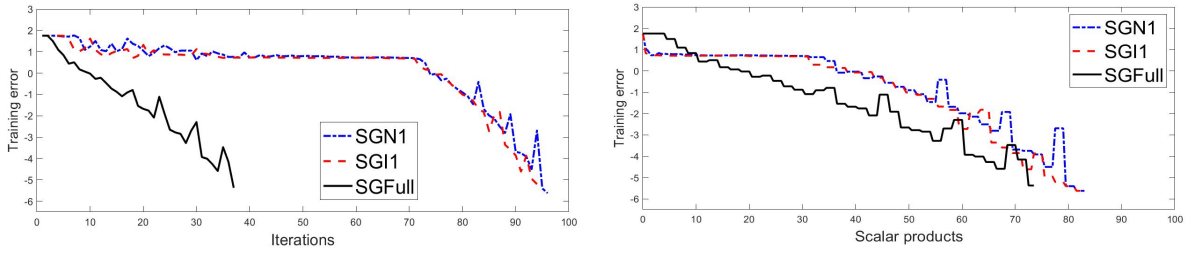

Figure 5. Mushrooms dataset, training error versus iterations (left) and versus scalar products (right).

right. The reported figures refer to a specific run out of 100 runs, representative of the average behavior of the methods. Plots are in logarithmic scale on the y-axes. As expected, for all the three datasets, the spectral gradient method without subsampling requires a much smaller number of iterations with respect to SG_N_1 and SG_I_1. Let us now consider the number of scalar products. Concerning CINA0 and MNIST dataset, both the subsampled methods are less expensive than SGFull, that is, they produce the same training error with a smaller number of scalar products. Moreover, the nested method SG_N_1 appears to be better than the non-nested one. Regarding the Mushrooms dataset, we have that SG_N_1 and SG_I_1 appear to behave very similarly and they are both less efficient than SGFull, therefore on this particular dataset the subsampling scheme does not seem to be convenient with respect to the method without subsampling. This is due to the fact that the size of the training set is small and therefore the gain obtained reducing the sample size is not enough to produce an overall saving.

We also notice that we do not expect an advantage in using the subsampling in all the situations where problems are easy enough (because the starting point is close to the solution, or due to the particular form of the functions f_j) to be solved by SGFull within a small number of iterations.

4.4. *Validation Loss*

In the previous subsections we tested the behavior of the Algorithm 2.2 using the deterministic stopping criterion - the norm of the full gradient. Then, we always reach the full sample and ask for an ϵ-accurate first-order method. However, in these applications a reasonable value of the validation error is needed rather than an accurate approximation of the minimizer of (25).

In order to do that, we show numerical results obtained using a stopping criterion related to the validation loss rather than to the training loss or to the full gradient. Notice that the testing data set is usually much smaller than the training set, in our runs it is constituted by the 5% of samples of the whole dataset. Letting

$$f_{\mathcal{V}}(x) = \frac{1}{|\mathcal{V}|}\sum_{j\in\mathcal{V}} f_j(x),$$

be the validation loss, we stop whenever the following condition is met:

$$f_{\mathcal{V}}(x_k) > 1.1 f_{\mathcal{V}}(x_{k-1}) \text{ or } |f_{\mathcal{V}}(x_{k-1}) - f_{\mathcal{V}}(x_k))| < 10^{-3}|f_{\mathcal{V}}(x_k))| \tag{27}$$

provided that $N_k \geq pN$, with $p \in (0,1)$. Thus, we stop whenever we monitor a 10% increase or a stallation of the validation loss, provided that the sample size is at least a fixed percentage of the full sample size.

In Figures 6-7 we plot the validation loss versus iterations and scalar products for both SG_N_1 and SG_I_1 using logarithmic scale on the y-axes. For the sake of comparison we also plot the validation loss of SGFull stopped when the norm of the gradient is smaller than 10^{-4}. This way we give a clear indication of the value of the validation loss that can be obtained by a method computing a 10^{-4}-accurate first-order critical point. The reported figures refer to a specific run out of 100 runs, representative of the general behavior of the methods. As a first comment we observe that in case of CINA0 dataset, we needed to reach the full sample in order to provide a validation error of the same order of that provided by SGFull. Therefore, for CINA0 dataset, we use $p = 1$ in (27). In fact, we can observe that the validation decreases in the very early stage of the iterative process, then it remains almost constant, and then rapidly drops to 0.4 as soon as the full sample is reached. However, notice that only a very few iterations are performed using the full sample as the average number of iterations is 86 for SG_N_1 and 90 for SG_I_1. Notice that the full sample is reached at iteration 82. On the other hand, considering Mushrooms and MNIST datasets a smaller number of samples are enough to provide an acceptable validation error. Thus, we set $p = 0.1$ in (27). In case of Mushrooms the average sample size at termination is 1164 and 1234 for SG_N_1 and SG_I_1, respectively. Both methods used around 25% of samples and reached a validation loss value of the order of 0.6 while SGFull reaches a value of about 0.3. Analogous behaviour can be observed in the solution of MNIST problem where the average sample size at termination is 7179 and 6520 for SG_N_1 and SG_I_1, respectively. Then, the 10% of samples are enough for this dataset.

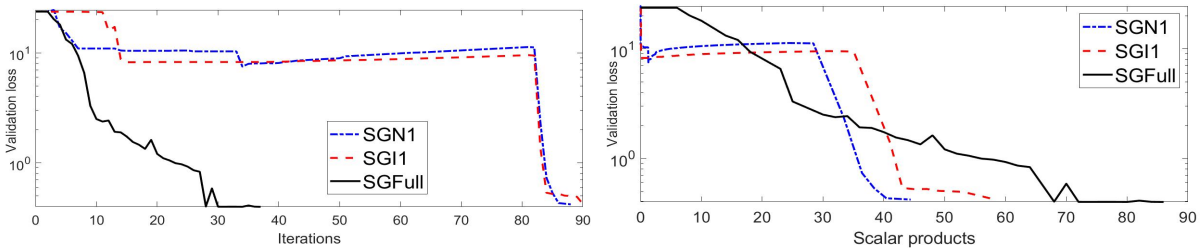

Figure 6. CINA0 dataset, validation loss versus iterations (left) and versus scalar products (right), stopping criterion (27) with $p = 1$

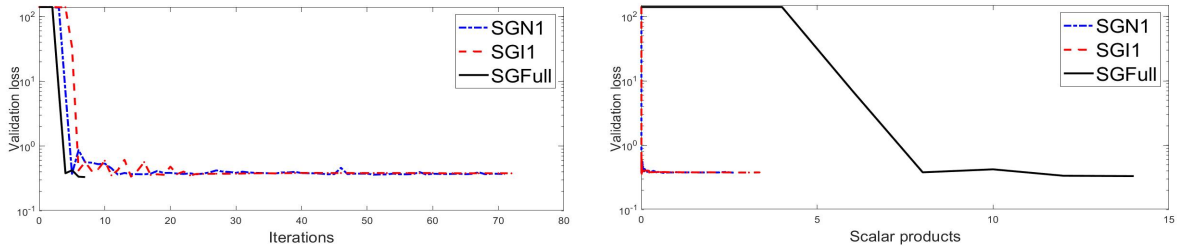

Figure 7. MNIST dataset, validation loss versus iterations (left) and versus scalar products (right), stopping criterion (27) with $p = 0.1$

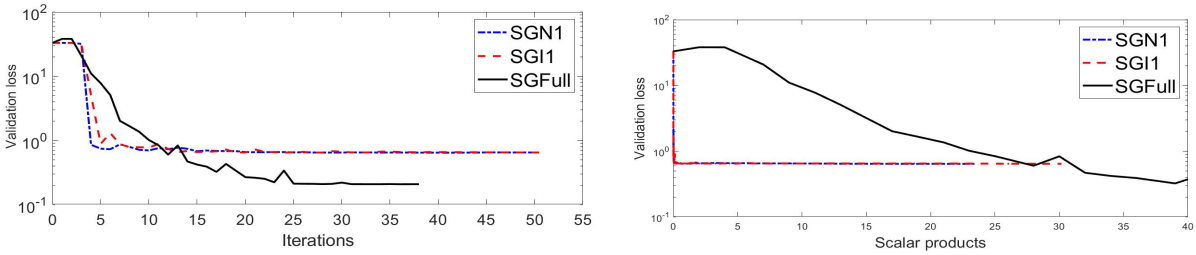

Figure 8. Mushrooms dataset, validation loss versus iterations (left) and versus scalar products (right), stopping criterion (27) with $p = 0.1$

5. Conclusions

We analyzed subsampled spectral gradient methods for solving large sum optimization problems with continuously-differentiable objective function. Our main aim was to provide initial understanding of the behavior of such methods with different kinds of spectral coefficients, or more precisely, with different kinds of gradient difference calculations used for obtaining spectral coefficients within the growing sample size framework.

Although the safeguards for the spectral coefficients provide a descent direction regarding the current approximate function, globalization strategy relies on nonmonotone line search. The motivation for this is twofold - to the best of our knowledge, both spectral gradient methods and subsampling methods favor nonmonotone line search techniques. Moreover, if the objective function is assumed to be strongly convex, R-linear convergence is achieved. We provided complexity results for both convex and non-convex case. Further modifications of this approach could be devised in order to solve optimization problems arising in the training of neural network employing non-smooth activation functions.

According to the tests performed on three binary classification problems, our main conclusions are as the following: 1) nested (cumulative) samples perform better than non-nested ones considering costs of the algorithm in terms of scalar products; 2) employing the same subset $\mathcal{N}_k$ for computing the displacement vector y_k proved to be beneficial despite the needed additional cost; 3) subsampling seems to reduce the overall computational cost as the tested subsampling variants outperform the full spectral gradient methods except in Mushrooms dataset where the size of the training set is small.

Finally, investigating on the validation loss, we conclude that a mini-batch version of the proposed spectral gradient methods has a good potential. However, stopping criterion in this setting is a problem itself and it needs further analysis. A related question is how the proposed methods compete in the unbounded sample case. This will be one of the topics of our further research.

Acknowledgements

Indam-GNCS partially supported the first author under Progetti di Ricerca 2018. The second author was supported by the Serbian Ministry of Education, Science and Technological Development, grant no. 174030. The third author was supported by the European Union's Horizon 2020 programme under the Marie Skłodowska-Curie Grant Agreement No 812912. The authors are grateful to the anonymous referee whose suggestions led to significant

improvement of the manuscript.

References

1. S. Bellavia, G. Gurioli, and B. Morini, Theoretical study of an adaptive cubic regularization method with dynamic inexact hessian information, *arXiv:1808.06239*, 2018.
2. S. Bellavia, N. Krejić, and N. Krklec Jerinkić, Subsampled inexact newton methods for minimizing large sums of convex functions, *IMA J. Numerical Analysis*, 2019.
3. A. Berahas, R. Bollapragada, and J. Nocedal, An investigation of newton-sketch and subsampled newton methods, *arXiv:1705.06211v3*, 2018.
4. E. Birgin, N. Krejić, and J. Martínez, On the employment of inexact restoration for the minimization of functions whose evaluation is subject to programming errors, *Mathematics of Computation*, vol. 87, pp. 1307–1326, 2018.
5. D. Blatt, A. O. Hero, and H. Gauchman, A convergent incremental gradient method with a constant step size, *SIAM Journal of Optimization*, vol. 18, no. 1, pp. 29–51, 2007.
6. R. Bollapragada, R. R. Byrd, and J. Nocedal, Exact and inexact subsampled newton methods for optimization, *IMA Journal Numerical Analysis*, 2018.
7. L. Bottou, F. E. Curtis, and J. Nocedal, Optimization methods for large-scale machine learning, *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.
8. L. Bottou, Stochastic gradient learning in neural networks, *Proceedings of Neuro-Nimes*, vol. 91, no. 8, p. 12, 1991.
9. R. Byrd, G. Chin, J. Nocedal, and Y. Wu, Sample size selection in optimization methods for machine learning, *Mathematical Programming*, vol. 134, no. 1, pp. 127–155, 2012.
10. M. P. Friedlander and M. Schmidt, Hybrid deterministic-stochastic methods for data fitting, *SIAM Journal on Scientific Computing*, vol. 34, no. 3, pp. 1380–1405, 2012.
11. R. Johnson and T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction, *Advances in Neural Information Processing Systems*, 2013.
12. N. Krejić and N. Krklec Jerinkić, Nonmonotone line search methods with variable sample size, *Numerical Algorithms*, vol. 68, no. 4, pp. 711–739, 2015.
13. F. Roosta-Khorasani and M. Mahoney, Sub-sampled newton methods, *Mathematical Programming*, vol. 174, pp. 293–326, 2019.
14. N. N. Schraudolph, J. Yu, and S. Günter, A stochastic quasi-newton method for online convex optimization, *SAIS International Conference on Artificial Intelligence and Statistics*, pp. 436–443, 2007.
15. C. Tan, S. Ma, Y. H. Dai, and Y. Qian, Barzilai-borwein step size for stochastic gradient descent, *Advances in Neural Information Processing Systems*, vol. 29, pp. 685–693, 2016.
16. Z. Yang, C. Wang, Z. Zhang, and J. Li, Random barzilai-borwein step size for mini-batch algorithms, *Engineering Applications of Artificial Intelligence*, vol. 72, pp. 124–135, 2018.
17. J. Barzilai and J. M. Borwein, Two-point step size gradient method, *IMA J. Numerical Analysis*, vol. 8, no. 1, pp. 141–148, 1988.
18. E. G. Birgin, J. M. Martínez, and M. Raydan, Spectral projected gradient methods: Review and perspectives, *Journal of Statistical Software*, vol. 60, 2014.
19. Y. H. Dai, W. W. Hager, K. Schittkowski, and H. Zhang, The cyclic barzilai-borwein method for unconstrained optimization, *IMA Journal Numerical Analysis*, vol. 26, no. 3, pp. 604–627, 2006.
20. R. Fletcher, On the barzilai-borwein gradient method, in *Optimization and Control with Applications, Applied Optimization* (L. Qi, K. Teo, and X. Yang, eds.), vol. 96, pp. 235–256, Springer, 2005.
21. D. di Serafino, V. Ruggiero, G. Toraldo, and L. Zanni, On the steplength selection in gradient methods for unconstrained optimization, *Applied Mathematics and Computation*, vol. 318, pp. 176–195, 2006.
22. M. Raydan, The barzilai and borwein gradient method for the large scale unconstrained minimization problem, *SIAM Journal Optimization*, vol. 7, no. 1, pp. 26–33, 1997.

23. N. Krejić and N. Krklec Jerinkić, Spectral projected gradient method for stochastic optimization, *Journal of Global Optimization*, vol. 73, pp. 59–81, 2018.
24. D. H. Li and M. Fukushima, A derivative-free line search and global convergence of broyden-like method for nonlinear equations, *Optimization Methods and Software*, vol. 13, no. 3, pp. 181–201, 2000.
25. G. N. Grapiglia and E. Sachs, On the worst-case evaluation complexity of nonmonotone line search algorithms, *Computational Optimization and applications*, vol. 68, no. 3, pp. 555–577, 2017.
26. Causality workbench team, a marketing dataset. http://www.causality.inf.ethz.ch/data/CINA.html, 2008.
27. M. Lichman, Uci machine learning repository. https://archive.ics.uci.edu/ml/index.php, 2013.