ORIGINAL PAPER

Random-reshuffled SARAH does not need full gradient computations

Aleksandr Beznosikov[1,2] · Martin Takáč[2]

Abstract

The StochAstic Recursive grAdient algoritHm (SARAH) algorithm is a variance reduced variant of the Stochastic Gradient Descent algorithm that needs a gradient of the objective function from time to time. In this paper, we remove the necessity of a full gradient computation. This is achieved by using a randomized reshuffling strategy and aggregating stochastic gradients obtained in each epoch. The aggregated stochastic gradients serve as an estimate of a full gradient in the SARAH algorithm. We provide a theoretical analysis of the proposed approach and conclude the paper with numerical experiments that demonstrate the efficiency of this approach.

Keywords Stochastic optimization · Finite-sum problems · Variance reduction · Convex optimization

1 Introduction

In this paper, we address the problem of minimizing a finite-sum problem of the form

$$\min_{w \in \mathbb{R}^d} \left\{ P(w) := \frac{1}{n} \sum_{i=1}^{n} f_i(w) \right\}, \tag{1}$$

where $\forall i \in [n] := \{1, 2, \dots, n\}$ the f_i is a convex function. We will further assume that $w^* = \arg \min P(w)$ exists.

✉ Aleksandr Beznosikov
   anbeznosikov@gmail.com

   Martin Takáč
   takac.mt@gmail.com

1   Moscow Institute of Physics and Technology (MIPT), Moscow, Russian Federation

2   Mohamed bin Zayed University of Artificial Intelligence (MBZUAI), Masdar City, Abu Dhabi, UAE

Problems of this form are very common in e.g., supervised learning [42]. Let a training dataset consists of n pairs, i.e., $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$ is a feature vector for a datapoint i and y_i is the corresponding label. Then for example, the least squares regression problem corresponds to (1) with $f_i(w) = \frac{1}{2}(x_i^T w - y_i)^2$. If $y_i \in \{-1, 1\}$ would indicate a class, then a logistic regression is obtained by choosing $f_i(w) = \log(1 + \exp(-y_i x_i^T w))$.

Recently, many algorithms have been proposed for solving (1). In this paper, we are interested in a subclass of these algorithms that fall into a stochastic gradient descent (SGD) framework originating from the work of Robbins and Monro in '50 s [40]. Let v_s will be some sort of (possibly stochastic and very rough) approximation of $\nabla P(w_s)$, then many SGD type algorithms update the w as follows:

$$w_{s+1} = w_s - \eta_s v_s, \tag{2}$$

where $\eta_s > 0$ is a predefined step-size. The classical SGD defines $v_s = \nabla f_i(w_s)$, where $i \in [n]$ is chosen randomly [43] or its mini-batch version [47], where $v_s = \frac{1}{|S|} \sum_{i \in S} \nabla f_i(w_s)$, with $S \subset [n]$. Even with an unbiased gradient estimates of SGD, where $\mathbb{E}[v_s | w_s] = \nabla P(w_s)$, the variance of v_s is the main source of slower convergence [6, 19, 33].

1.1 Brief literature review

Recently, many variance-reduced variants of SGD have been proposed, including SAG/SAGA [10, 38, 41], SVRG [3, 18, 51], MISO [25], SARAH [15, 31, 32, 34], SPIDER [11], STORM [9], PAGE [22], and many others. Generally speaking, the variance-reduced variants of SGD still aim to sample $\mathcal{O}(1)$ functions and use their gradients to update v_s. For example, SVRG [18] will fix a point $\tilde{w}$, in which a full gradient $\nabla P(\tilde{w})$ is computed and subsequently stochastic gradient is defined as $v_s = \nabla f_i(w_s) - \nabla f_i(\tilde{w}) + \nabla P(\tilde{w})$, where $i \in [n]$ is picked at random.

SARAH Algorithm The SARAH algorithm [31], on the other hand, updates v_s recursively. It starts with a full gradient computation $v_0 = \nabla P(w_0)$, then taking a step (2) and updating the gradient estimate recursively as $v_s = \nabla f_i(w_s) - \nabla f_i(w_{s-1}) + v_{s-1}$. For a smooth and strongly convex problem, the procedure highlighted above converges, but not to the optimal solution w^*. Therefore, similarly to SVRG, the process is restarted after i) a predefined number of iterations, ii) randomly [21–23], or iii) decided in run-time by computing the ratio $\|v_s\| / \|v_0\|$ (SARAH+ [31]), and a new full gradient estimate has to be computed. To elevate this issue, e.g., in [34], they proposed inexact SARAH (iSARAH), where the full gradient estimate is replaced by a mini-batch gradient estimate $v_0 = \frac{1}{|S|} \sum_{i \in S} f_i(w_0)$, where $S \subset [n]$. To find a point $\hat{w}$ such that $\mathbb{E}[\|\nabla P(\hat{w})\|^2] \leq \epsilon$, the mini-batch size has to be chosen as $|S| \sim \mathcal{O}(\frac{1}{\epsilon})$, and the step-size will be $\eta \sim \mathcal{O}(\frac{\epsilon}{L})$.

There are a few variants of SARAH that do not need any restart and full-gradient estimate. E.g., the Hybrid Variance-Reduced variant [24] defines

$$v_s = \beta \nabla f_i(w_s) + (1 - \beta)\big(\nabla f_i(w_s) - \nabla f_i(w_{s-1}) + v_{s-1}\big), \tag{3}$$

where $\beta \in (0, 1)$ is a hyperparameter. The STORM variant [9] uses (3) not with a fixed value of parameter β, but in STORM, the value of β_s is diminishing to 0. The ZeroSARAH [23] is another variant, where the v_s is a combination of (3) with SAG/SAGA.

Random sampling versus random reshuffle All the stochastic algorithms discussed so far sample function f_i randomly. However, it is standard practice, for a finite-sum problem, not to choose functions f_i randomly with replacement, but rather make a data permutation/shuffling and then choose the f_is in a cyclic fashion [5, 39]. This is primarily because methods involving data shuffling are more cache-friendly, which leads to a significant acceleration in their implementation, as noted in [4]. From learning and optimization perspectives, a cyclic pass through the data appears to be more beneficial as it ensures that each data point is utilized an equal number of times throughout the training process. Hence, shuffling strategies are frequently used for real-world practical problems, including those encountered in deep learning, as outlined in [45].

The basic shuffling approaches include

- *Random Reshuffling (RR)*—reshuffle data before each epoch;
- *Shuffle-Once (SO)*—shuffle data only once before optimizing;
- *Incremental Gradient (IG)*—access data in a cycling fashion over the given dataset.

In recent years, numerous studies have offered theoretical analyses of various shuffling techniques, both in the context of classical SGD [1, 17, 20, 27, 35, 53] and in variance reduction methods [13, 16, 26, 28, 36, 46, 49, 52].

In Table 1 we summarize the results from the literature on stochastic methods with independent generation of batch numbers and with shuffling techniques. In the case of independent generation, the results do not claim to be complete and are given rather for information and comparison.

1.2 Contribution

The main contribution of this paper is the modification of the SARAH algorithm to remove the requirement of computing a full gradient $\nabla P(w)$, while achieving a linear convergence with a fixed step-size for strongly convex objective. The crucial algorithmic modification that was needed to achieve this goal, was to replace the random selection of functions by either of the shuffle options (**RR**, **SO**, **IG**) and designing a mechanism that can build a progressively better approximation of a full gradient $\nabla P(w_s)$ as $w_s \to w^*$.

One can note that, according to Table 1, our Algorithm 1 is one of the few methods with shuffling that simultaneously does not compute full gradients, does not use large amounts of additional memory, and converges linearly for smooth strongly convex finite-sum problems (1). Moreover, with a sufficiently small δ (see the discussion after Assumption 1 and at the end of Sect. 3), Algorithm 1 has state-of-the-art guarantees of convergence among methods using the shuffling technique.

Table 1 Summary of results on solving a smooth strongly convex finite-sum optimization problem

	Method	References	No Full Grad.?	Memory	Lin. Conv.?	Complexity
Independent	GD	[37]	✗	$\mathcal{O}(d)$	✓	$\mathcal{O}\left(n\frac{L}{\mu}\log\frac{1}{\varepsilon}\right)$
	AGD	[30]	✗	$\mathcal{O}(d)$	✓	$\mathcal{O}\left(n\sqrt{\frac{L}{\mu}}\log\frac{1}{\varepsilon}\right)$
	SGD	[40, 29, 12, 44]	✓	$\mathcal{O}(d)$	✗	$\mathcal{O}\left(\frac{L}{\mu}\log\frac{1}{\varepsilon}+\frac{\sigma_*^2}{\mu^2\varepsilon}\right)$
	ASGD	[8, 50]	✓	$\mathcal{O}(d)$	✗	$\mathcal{O}\left(\sqrt{\frac{L}{\mu}}\log\frac{1}{\varepsilon}+\frac{\sigma_*^2}{\mu^2\varepsilon}\right)$
	SAGA	[10]	✓	$\mathcal{O}(nd)$	✓	$\mathcal{O}\left(\left[n+\frac{L}{\mu}\right]\log\frac{1}{\varepsilon}\right)$
	SVRG	[18]	✗	$\mathcal{O}(d)$	✓	$\mathcal{O}\left(\left[n+\frac{L}{\mu}\right]\log\frac{1}{\varepsilon}\right)$
	SARAH	[31]	✗	$\mathcal{O}(d)$	✓	$\mathcal{O}\left(\left[n+\frac{L}{\mu}\right]\log\frac{1}{\varepsilon}\right)$
	Katyusha	[2]	✗	$\mathcal{O}(d)$	✓	$\mathcal{O}\left(\left[n+\sqrt{n\frac{L}{\mu}}\right]\log\frac{1}{\varepsilon}\right)$
Shuffling	SGD	[53, 17, 1, 27, 35]	✓	$\mathcal{O}(d)$	✗	$\mathcal{O}\left(\frac{L}{\mu}\log\frac{1}{\varepsilon}+\sqrt{\frac{L\sigma_*^2}{\mu^3 n\varepsilon}}\right)$
	IAG	[49, 13]	✓	$\mathcal{O}(nd)$	✓	$\mathcal{O}\left(n\frac{L}{\mu}\log\frac{1}{\varepsilon}\right)$
	DIAG	[28]	✓	$\mathcal{O}(nd)$	✓	$\mathcal{O}\left(n\frac{L}{\mu}\log\frac{1}{\varepsilon}\right)$
	SAGA	[46, 36, 52]	✓	$\mathcal{O}(nd)$	✓	$\mathcal{O}\left(n\frac{L^2}{\mu^2}\log\frac{1}{\varepsilon}\right)$
	SVRG	[46, 26]	✗	$\mathcal{O}(d)$	✓	$\mathcal{O}\left(n\frac{L^{3/2}}{\mu^{3/2}}\log\frac{1}{\varepsilon}\right)$
	AVRG	[52]	✓	$\mathcal{O}(d)$	✓	$\mathcal{O}\left(n\frac{L^2}{\mu^2}\log\frac{1}{\varepsilon}\right)$
	Prox-DFinito	[16]	✓	$\mathcal{O}(nd)$	✓	$\mathcal{O}\left(n\frac{L}{\mu}\log\frac{1}{\varepsilon}\right)$
	Algorithm 1	this paper	✓	$\mathcal{O}(d)$	✓	$\mathcal{O}\left(\left[n\frac{L}{\mu}+n^2\frac{\delta}{\mu}\right]\log\frac{1}{\varepsilon}\right)$
	Algorithm 2	this paper	✗	$\mathcal{O}(d)$	✓	$\mathcal{O}\left(\left[n\frac{L}{\mu}+n^2\frac{\delta}{\mu}\right]\log\frac{1}{\varepsilon}\right)$

The results are given both for independent generation of batch numbers and for different shuffling techniques. In the case of independent generation, the results are given for reference and make no claim to completeness

Columns No Full Grad.? = whether the method computes full gradients, Memory = number of additional memory, Lin. Conv.? = whether the method converges linearly to the solution w^*, Complexity = oracle complexity on the terms f_i to find the solution (give the best result of the works listed in References)

Notation μ = constant of strong monotonicity of P, L = Lipschitz constant of ∇f_i, δ = similarity parameter, $\sigma_*^2 = \frac{1}{n}\sum_{i=1}^{n}\|\nabla f_i(w^*)\|^2$ (in [27], the authors use a slightly different definition), σ^2 = upper bound for $\|\nabla f_i(w) - \nabla P(w)\|^2$, n = size of the dataset, d = dimension of the problem, ε = accuracy of the solution

2 Shuffled-SARAH

2.1 Building gradient estimate while optimizing

An intuition By accessing data in a cyclic order, employing any of the alternative methods mentioned earlier, we can approximate the full gradient $\tilde{v} \approx \nabla P$. In fact, if

we set the step-size η_s in (2) to zero and let v_s represent a stochastic gradient ∇f_i, then by averaging all the stochastic gradients in a single pass, we would obtain the exact full gradient $\nabla P(w)$. As we increase the value of η, the stochastic gradients would be computed at different points

$$\tilde{v} = \frac{1}{n} \sum_{i=1}^{n} \nabla f_{\pi^i}(w_i), \tag{4}$$

and hence we would not obtain the exact full gradient of $P(w)$ but rather just a rough estimate. But is it just the η that affects how good the $\tilde{v}$ will be? Of course not, as w_s is updated using (2), one can see that the radius of a set of ws that are used to compute gradient estimates is dependent on v_s. Ideally, as we will converge go w^*, then also $v_s \rightarrow \nabla P(w^*) = 0$ and hence $\tilde{v}$ will be getting closer to $\nabla P(w_s)$.

Building the gradient estimate Our proposed approach to eliminate the need to compute the full gradient is based on a simple recursive update. Let us initialize $\tilde{v}_0 = \mathbf{0} \in \mathbb{R}^d$. Then while making a pass $i = \{1, 2, \ldots, n\}$ over the data, we will keep updating $\tilde{v}$ using the gradient estimates as follows

$$\tilde{v}_i = \frac{i-1}{i} \tilde{v}_{i-1} + \frac{1}{i} \nabla f_{\pi^i}(w_i), \quad \text{for } i \in \{1, 2, \ldots, n\}.$$

It is an easy exercise to see that $\tilde{v}_i$ will be the average of gradients seen so far, and moreover, after n updates, it will be exactly as in (4). Let us note that making the pass over the dataset is crucial to build a good estimate of the gradient and random selection of functions would not achieve this goal.

The algorithm We are now ready to explain the `Shuffled-SARAH` algorithm, as outlined in Algorithm 1. The algorithm commences with the selection of an initial solution w^-, which can be randomly chosen or simply set to, for example, $\mathbf{0}$. Subsequently, we define $v_0 = \mathbf{0}$, which will consistently serve as an estimate of the full gradient ∇P.

In line 5, we assign $\tilde{v}$ to share the same memory address as v_0. Essentially, this implies that during the first pass $s = 0$, v_0 and $\tilde{v}$ will always be identical, and any modification to $\tilde{v}$ will concurrently affect v_0. After executing lines 18, 19, v_s and $\tilde{v}$ will evolve into two distinct vectors.

The reason we have included the term *in place* in line 13 is solely to ensure that for $s = 0$, both v_0 and $\tilde{v}$ will remain the same.

The random permutation in line 11 could have one of the three options mentioned in Sect. 1.1. For **RR**, we will permute the $[n]$ each time, for **SO** we will only shuffle once for $s = 0$ and define $\pi_s = \pi_0$ for any $s > 0$. In **IG** option we have $\pi_s = (1, 2, \ldots, n) \, \forall s$.

```
 1  Input: 0 < η step-size
 2  choose w⁻ ∈ ℝᵈ
 3  w = w⁻
 4  v₀ = 0 ∈ ℝᵈ
 5  ṽ = &(v₀)        // ṽ will point to v₀
 6  Δ = 0 ∈ ℝᵈ
 7  for s = 0, 1, 2, … do
 8  │   define wₛ := w
 9  │   w⁻ = w
10  │   w = w − ηvₛ
11  │   obtain permutation πₛ = (π¹ₛ, …, πⁿₛ) of [n] by some rule
12  │   for i = 1, 2, …, n do
13  │   │   ṽ = (i−1)/i ṽ + 1/i ∇f_{π^i_s}(w)
14  │   │   Δ = Δ + ∇f_{π^i_s}(w) − ∇f_{π^i_s}(w⁻)
15  │   │   w⁻ = w
16  │   │   w = w − η(vₛ + Δ)
17  │   end
18  │   v_{s+1} = ṽ
19  │   ṽ = 0 ∈ ℝᵈ
20  │   Δ = 0 ∈ ℝᵈ
21  end
22  Return:  w
```

Algorithm 1 `Shuffled-SARAH`

3 Theoretical analysis

Before presenting our theoretical results we introduce some notations and assumptions.

We use $\langle x, y \rangle := \sum_{i=1}^{d} x_i y_i$ to define standard inner product of $x, y \in \mathbb{R}^d$. It induces ℓ_2-norm in $\mathbb{R}^d$ in the following way $\|x\| := \sqrt{\langle x, x \rangle}$.

Assumption 1 For problem (1) the following hold:

(i) Each $f_i : \mathbb{R}^d \to \mathbb{R}$ is convex and twice differentiable, with L-Lipschitz gradient:

$$\|\nabla f_i(w_1) - \nabla f_i(w_2)\| \le L\|w_1 - w_2\|,$$

for all $w_1, w_2 \in \mathbb{R}^d$;

(ii) $P(w)$ is a μ-strongly convex function with minimizer x^* and optimal value P^*;

(iii) Each f_i is δ-similar with P, i.e. for all $w \in \mathbb{R}^d$ it holds that

$$\|\nabla^2 f_i(w) - \nabla^2 P(w)\| \le \delta/2,$$

where $\| \cdot \|$ is the operator norm between matrices.

The last assumption pertains to the Hessian similarity of $\{f_i\}_{i=1}^{n}$. This effect is prominently seen when the data with $n \cdot b$ samples is randomly partitioned across n batches $\{\{z_j^i\}_{j=1}^{b}\}_{i=1}^{n}$ and f_i is defined as $f_i(w) = \frac{1}{b} \sum_{j=1}^{b} \ell(w, z_j^i)$, where ℓ is a L-smooth and convex loss function. One can see that

$P(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w) = \frac{1}{nb} \sum_{i=1}^{n} \sum_{j=1}^{b} \ell(w, z_j^i)$ forms the total loss function over the whole dataset.

Subsequently, one can use Hoeffding's inequality for matrices [48] and obtain that with probability $1 - p$ it holds

$$\|\nabla^2 f_i(w) - \nabla^2 P(w)\| \leq \sqrt{\frac{32L^2 \log(d/p)}{b}}.$$

It turns out that we can consider that $\delta \sim \frac{L}{\sqrt{b}}$. Moreover, in the case when ℓ is a quadratic function $\delta \sim \frac{L}{b}$ [14].

The following theorem presents the convergence guarantees of `Shuffled-SARAH`.

Theorem 1 *Suppose that Assumption* 1 *holds. Consider* `Shuffled-SARAH` (*Algorithm* 1) *with the choice of η such that*

$$\eta \leq \min\left[\frac{1}{8nL}; \frac{1}{8n^2\delta}\right]. \tag{5}$$

Then, we have

$$P(w_{s+1}) - P^* + \frac{\eta(n+1)}{16}\|v_s\|^2$$
$$\leq \left(1 - \frac{\eta\mu(n+1)}{2}\right)\left(P(w_s) - P^* + \frac{\eta(n+1)}{16}\|v_{s-1}\|^2\right).$$

Hence, it is easy to obtain an estimate for the number of outer iterations in Algorithm 1.

Corollary 1 *Fix ε, and let us run* `Shuffled-SARAH` *with η from* (5). *Then we can obtain an ε-approximate solution* (*in terms of $P(w) - P^* \leq \varepsilon$*) *after*

$$S = \mathcal{O}\left(\left[n \cdot \frac{L}{\mu} + n^2 \cdot \frac{\delta}{\mu}\right] \log \frac{1}{\varepsilon}\right) \quad \text{calls of terms } f_i.$$

Difference between RR, SO, IG A crucial detail is that we achieve deterministic convergence estimates irrespective of the generation strategy π_s employed in line 11 of Algorithm 1. Concurrently, it's anticipated that the diverse characteristics of π_s may result in variations in theoretical convergence. Indeed, different shuffling techniques have a theoretical impact on the SGD method as discussed in the most recent research findings [20, 27]. However, a significant detail of these findings is that the differences only manifest in the portion that depends on the variance bound of the stochastic gradient σ^2. But SARAH, like other methods incorporating variance reduction, nullifies the effect of σ^2. The most credible results on the effect of variance reduction and shuffling techniques do not discern a considerable difference among **RR**, **SO**, and **IG** generally [16, 26]. Minor improvements

are detectable in specific cases [26] or include only logarithmic factors [16]. Consequently, the question of distinguishing between **RR**, **SO**, and **IG** in variance reduction algorithms remains unanswered.

Optimal choice of b In the case of $\delta \sim \frac{L}{\sqrt{b}}$, it is required to take $\frac{n}{\sqrt{b}} \sim 1$, i.e. $b = N^{2/3}$, where N is the total sample size: $N = nb$.

4 Numerical experiments

Trajectory We start with a toy experiment in $\mathbb{R}^2$ with a quadratic function. We compare the trajectories of the classical SARAH (two random and average), the average trajectory of the RR-SARAH (see Algorithm 2 in "Appendix 2"), and the random trajectory of Shuffled-SARAH with Random Reshuffling (Fig. 1).

Logistic regression Next, we consider the logistic regression problem with ℓ_2-regularization for binary classification with

$$f_i(w) = \frac{1}{b} \sum_{k=1}^{b} \log \left(1 + \exp \left(-y_k \cdot (X_b w)_k \right) \right) + \frac{\lambda}{2} \|w\|^2,$$
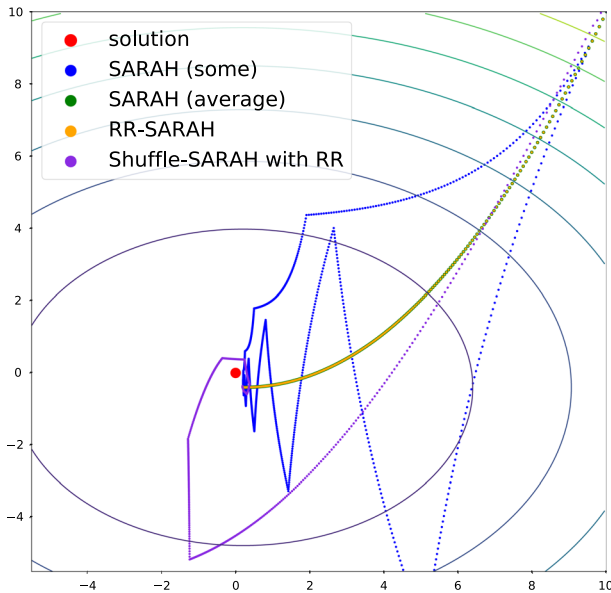
Fig. 1 Trajectories on quadratic function

Table 2 Summary of datasets

	N	b	d	L
mushrooms	8124	64	112	5,3
a9a	32,561	256	123	3,5
w8a	49,749	256	300	28,5

Table 3 Optimal parameters for SARAH

	m	η
mushrooms	$0.5 \cdot (L/\mu)$	$1/L$
a9a	$0.25 \cdot (L/\mu)$	$1/L$
w8a	L/μ	$1/L$
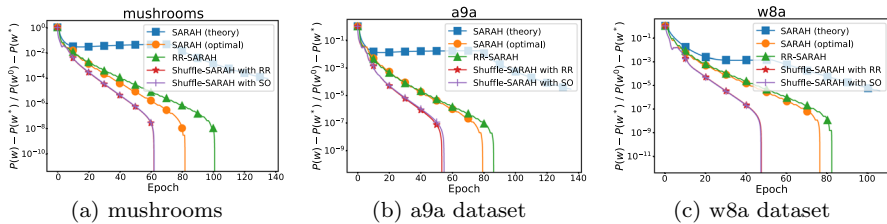
Fig. 2 Convergence of SARAH-type methods on various LiBSVM datasets. Convergence on the function

where $X_b \in \mathbb{R}^{b \times d}$ is a matrix of objects, $y_1, \dots, y_b \in \{-1, 1\}$ are labels for these objects, b is the size of the local datasets and $w \in \mathbb{R}^d$ is a vector of weights. We optimize this problem for mushrooms, a9a, w8a datasets from LIBSVM library[7]. Since we only have the estimate on b for our method (see the end of Sect. 3), it is not completely fair to use it for all methods, therefore we take as b the nearest power of two to $N/100$, where N is the full size of the data. More details on the dataset parameters can be found in Table 2. For each dataset we take the regularization parameter $\lambda = 0.001 \cdot L$. One can note that the strong convexity constant of the problem $\mu = \lambda$. We compare the following method settings: (1) SARAH with theoretical parameters $m = 4.5 \cdot (L/\mu), \eta = 1/(2L)$ (m is the number of iterations of the inner loop – see [31]), (2) SARAH with optimal parameters (is selected by brute force—see Table 3), (3) RR-SARAH with optimal tuned step-size, (4) Shuffled-SARAH (Random Reshuffling) with optimal tuned step-size, (5) Shuffled-SARAH (Shuffle Once) with optimal tuned step-size. All methods are run 20 times, and the convergence results are averaged. We are interested in how these methods converge in terms of the epochs number (1 epoch is a call of the full gradient of P). For results
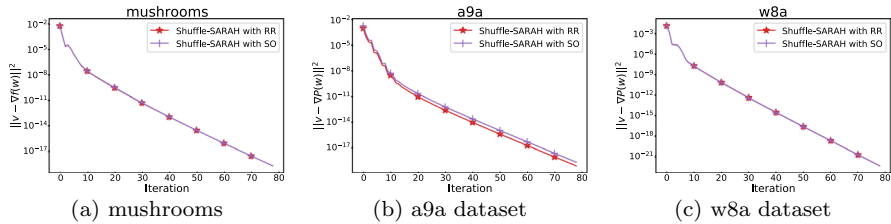
Fig. 3 $\|v_s - \nabla P(w_s)\|^2$ changes

see Figs. 2, 4 and 5. One can note that in these cases our new methods are superior to the original SARAH.

The v_s is getting closer to $\nabla P(w_s)$ The goal of this experiment is to show that v is a good approximation of ∇P and improves with each iteration. To do this, we analyze the changes of $\|v_s - \nabla P(w_s)\|^2$ on the logistic regression problem (see the previous paragraph). See the results in Fig. 3. It can be seen that the difference is decreasing $\|v_s - \nabla P(w_s)\|^2$.

Appendix 1: Additional experimental results

See Figs. 4 and 5.
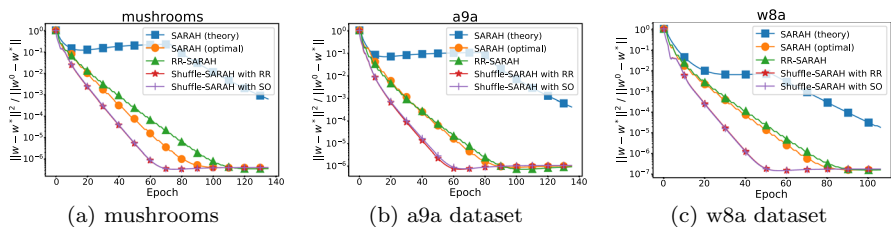
Fig. 4 Convergence of SARAH-type methods on various LiBSVM datasets. Convergence on the distance to the solution
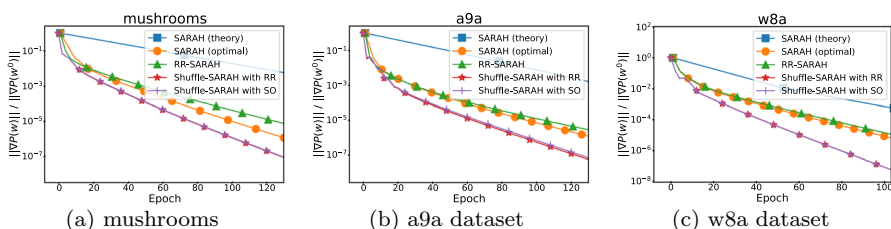
Fig. 5 Convergence of SARAH-type methods on various LIBSVM datasets. Convergence on the norm og the gradient

Appendix 2: `RR-SARAH`

This Algorithm is a modification of the original SARAH using Random Reshuffling. Unlike Algorithm 1, this algorithm uses the full gradient ∇P.

Algorithm 2 `RR-SARAH`

1 **Input:** $0 < \eta$ step-size
2 choose $w^- \in \mathbb{R}^d$
3 $w = w^-$
4 **for** $s = 0, 1, 2, \ldots$ **do**
5   define $w_s := w$
6     $v = \nabla P(w)$
7     $w^- = w$
8     $w = w - \eta v$
9   sample a permutation $\pi_s = (\pi_s^1, \ldots, \pi_s^n)$ of $[n]$
10   **for** $i = 1, 2, \ldots, n$ **do**
11        $v = v + \nabla f_{\pi_s^i}(w) - \nabla f_{\pi_s^i}(w^-)$
12        $w^- = w$
13        $w = w - \eta v$
14   **end**
15 **end**
16 **Return:** w

Theorem 2 *Suppose that Assumption 1 holds. Consider* `RR-SARAH` *(Algorithm 2) with the choice of η such that*

$$\eta \le \min \left[\frac{1}{8nL}; \frac{1}{8n^2\delta} \right]. \tag{6}$$

Then, we have

$$P(w_{s+1}) - P^* \le \left(1 - \frac{\eta\mu(n+1)}{2} \right) \left(P(w_s) - P^* \right).$$

Corollary 2 *Fix ϵ, and let us run* `RR-SARAH` *with η from (6). Then we can obtain an ϵ-approximate solution (in terms of $P(w) - P^* \le \epsilon$) after*

$$S = \mathcal{O}\left(\left[n \cdot \frac{L}{\mu} + n^2 \cdot \frac{\delta}{\mu} \right] \log \frac{1}{\epsilon} \right) \quad \text{calls of terms } f_i.$$

Appendix 3: Missing proofs for Sect. 3 and "Appendix 2"

Before we start to prove, let us note that δ-similarity from Assumption 1 gives $(\delta/2)$-smoothness of function $(f_i - P)$ for any $i \in [n]$. This implies δ-smoothness of function $(f_i - f_j)$ for any $i, j \in [n]$:

$$\|\nabla f_i(w_1) - \nabla f_j(w_1) - (\nabla f_i(w_2) - \nabla f_j(w_2))\|$$
$$\leq \|\nabla f_i(w_1) - \nabla P(w_1) - (\nabla f_i(w_2) - \nabla P(w_2))\|$$
$$+ \|\nabla P(w_1) - \nabla f_j(w_1) - (\nabla P(w_2) - \nabla f_j(w_2))\| \tag{7}$$
$$\leq 2 \cdot (\delta/2)\|w_1 - w_2\|^2 = \delta\|w_1 - w_2\|^2.$$

Next, we introduce additional notation for simplicity. If we consider Algorithm 1 in iteration $s \neq 0$, one can note that update rule is nothing more than

$$w_s = w_s^0 = w_{s-1}^{n+1},$$
$$v_s = v_s^0 = \frac{1}{n}\sum_{i=1}^{n} f_{\pi_{s-1}^i}(w_{s-1}^i), \tag{8}$$
$$w_s^1 = w_s^0 - \eta v_s^0,$$

$$v_s^i = v_s^{i-1} + f_{\pi_s^i}(w_s^i) - f_{\pi_s^i}(w_s^{i-1}), \tag{9}$$

$$w_s^{i+1} = w_s^i - \eta v_s^i. \tag{10}$$

These new notations will be used further in the proofs. For Algorithm 2, one can do exactly the same notations with $v_s = v_s^0 = \nabla P(w_s)$.

Lemma 1 *Under Assumption* 1, *for Algorithms* 1 *and* 2 *with η from* (5) *the following holds*

$$P(w_{s+1}) \leq P(w_s) - \frac{\eta(n+1)}{2}\|\nabla P(w_s)\|^2 + \frac{\eta(n+1)}{2}\left\|\nabla P(w_s) - \frac{1}{n}\sum_{i=0}^{n} v_s^i\right\|^2.$$

Proof Using L-smoothness of function P (Assumption 1 (i)), we have

$$P(w_{s+1}) \leq P(w_s) + \langle \nabla P(w_s), w_{s+1} - w_s \rangle + \frac{L}{2}\|w_{s+1} - w_s\|^2$$
$$= P(w_s) - \eta(n+1)\left\langle \nabla P(w_s), \frac{1}{n+1}\sum_{i=0}^{n} v_s^i \right\rangle + \frac{\eta^2(n+1)^2 L}{2}\left\|\frac{1}{n+1}\sum_{i=0}^{n} v_s^i\right\|^2$$
$$= P(w_s) - \frac{\eta(n+1)}{2}\left(\|\nabla P(w_s)\|^2 + \left\|\frac{1}{n+1}\sum_{i=0}^{n} v_s^i\right\|^2 - \left\|\nabla P(w_s) - \frac{1}{n+1}\sum_{i=0}^{n} v_s^i\right\|^2\right)$$
$$+ \frac{\eta^2(n+1)^2 L}{2}\left\|\frac{1}{n+1}\sum_{i=0}^{n} v_s^i\right\|^2$$
$$= P(w_s) - \frac{\eta(n+1)}{2}\|\nabla P(w_s)\|^2 - \frac{\eta(n+1)}{2}(1 - \eta(n+1)L)\left\|\frac{1}{n+1}\sum_{i=0}^{n} v_s^i\right\|^2$$
$$+ \frac{\eta(n+1)}{2}\left\|\nabla P(w_s) - \frac{1}{n+1}\sum_{i=0}^{n} v_s^i\right\|^2.$$

With $\eta \le \frac{1}{8nL} \le \frac{1}{(n+1)L}$, we get

$$P(w_{s+1}) \le P(w_s) - \frac{\eta(n+1)}{2}\|\nabla P(w_s)\|^2 + \frac{\eta(n+1)}{2}\left\|\nabla P(w_s) - \frac{1}{n+1}\sum_{i=0}^{n} v_s^i\right\|^2.$$

Which completes the proof. □

Lemma 2 *Under Assumption* 1, *for Algorithms* 1 *and* 2 *the following holds*

$$\left\|\nabla P(w_s) - \frac{1}{n+1}\sum_{i=0}^{n} v_s^i\right\|^2 \le 2\|\nabla P(w_s) - v_s\|^2 + \left(\frac{4L^2}{n+1} + 4\delta^2 n\right)\sum_{i=1}^{n}\|w_s^i - w_s\|^2.$$

Proof To begin with, we prove that for any $k = n, \ldots, 0$, it holds

$$\sum_{i=k}^{n} v_s^i = \sum_{i=k+1}^{n}\left((n+1-i)\cdot\left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1})\right]\right) + (n-k+1)v_s^k. \quad (11)$$

One can prove it by mathematical induction. For $k = n$, we have $\sum_{i=n}^{n} v_s^i = v_s^n$. Suppose (11) holds true for k, let us prove for $k-1$:

$$\sum_{i=k-1}^{n} v_s^i = v_s^{k-1} + \sum_{i=k}^{n} v_s^i$$

$$= v_s^{k-1} + \sum_{i=k+1}^{n}\left((n+1-i)\cdot\left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1})\right]\right) + (n-k+1)v_s^k$$

$$= v_s^{k-1} + \sum_{i=k+1}^{n}\left((n+1-i)\cdot\left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1})\right]\right)$$

$$\quad + (n-k+1)\left[v_s^{k-1} + f_{\pi_s^k}(w_s^k) - f_{\pi_s^k}(w_s^{k-1})\right]$$

$$= \sum_{i=k}^{n}\left((n+1-i)\cdot\left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1})\right]\right) + (n-k)v_s^{k-1}.$$

Here we additionally used (9). This completes the proof of (11). In particular, (11) with $k = 0$ gives

$$\left\| \nabla P(w_s) - \frac{1}{n+1} \sum_{i=0}^{n} v_s^i \right\|^2$$

$$= \frac{1}{(n+1)^2} \left\| (n+1) \nabla P(w_s) - \sum_{i=0}^{n} v_s^i \right\|^2$$

$$= \frac{1}{(n+1)^2} \left\| (n+1) \nabla P(w_s) \right.$$

$$\left. - \sum_{i=1}^{n} \left((n+1-i) \cdot \left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1}) \right] \right) + (n+1) v_s^0 \right\|^2.$$

Using $\|a + b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$, we get

$$\left\| \nabla P(w_s) - \frac{1}{n+1} \sum_{i=0}^{n} v_s^i \right\|^2$$

$$\leq 2\|\nabla P(w_s) - v_s^0\|^2 \tag{12}$$

$$+ \frac{2}{(n+1)^2} \left\| \sum_{i=1}^{n} \left((n+1-i) \cdot \left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1}) \right] \right) \right\|^2.$$

Again using mathematical induction, we prove for $k = n, \dots, 1$ the following estimate:

$$\left\| \sum_{i=1}^{n} \left((n+1-i) \cdot \left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1}) \right] \right) \right\|^2$$

$$\leq 2L^2(n+1) \sum_{i=k+1}^{n} \|w_s^i - w_s\|^2 + 2\delta^2(n+1) \sum_{i=k+1}^{n} (n+1-i)^2 \|w_s - w_s^{i-1}\|^2$$

$$+ \frac{n+1}{k+1} \left\| (n-k) \nabla f_{\pi_s^k}(w_s) + \nabla f_{\pi_s^k}(w_s^k) - (n-k+1) \nabla f_{\pi_s^k}(w_s^{k-1}) \right. \tag{13}$$

$$\left. + \sum_{i=1}^{k-1} \left((n+1-i) \cdot \left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1}) \right] \right) \right\|^2.$$

For $k = n$, the statement holds automatically. Suppose (13) holds true for k, let us prove for $k - 1$:

$$\left\| \sum_{i=1}^{n} \left((n+1-i) \cdot \left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1}) \right] \right) \right\|^2$$

$$\leq 2L^2(n+1) \sum_{i=k+1}^{n} \| w_s^i - w_s \|^2 + 2\delta^2(n+1) \sum_{i=k+1}^{n} (n-i+1)^2 \| w_s - w_s^{i-1} \|^2$$

$$+ \frac{n+1}{k+1} \left\| (n-k)\nabla f_{\pi_s^k}(w_s) + \nabla f_{\pi_s^k}(w_s^k) - (n-k+1)\nabla f_{\pi_s^k}(w_s^{k-1}) \right.$$

$$+ \sum_{i=1}^{k-1} \left((n+1-i) \cdot \left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1}) \right] \right) \Big\|^2$$

$$= 2L^2(n+1) \sum_{i=k+1}^{n} \| w_s^i - w_s \|^2 + 2\delta^2(n+1) \sum_{i=k+1}^{n} (n-i+1)^2 \| w_s - w_s^{i-1} \|^2$$

$$+ \frac{n+1}{k+1} \left\| \nabla f_{\pi_s^k}(w_s^k) - f_{\pi_s^k}(w_s) + (n-k+1)\nabla f_{\pi_s^k}(w_s) - (n-k+1)\nabla f_{\pi_s^k}(w_s^{k-1}) \right.$$

$$+ (n-k+2) \cdot \left[\nabla f_{\pi_s^{k-1}}(w_s^{k-1}) - \nabla f_{\pi_s^{k-1}}(w_s^{k-2}) \right]$$

$$+ \sum_{i=1}^{k-2} \left((n+1-i) \cdot \left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1}) \right] \right) \Big\|^2$$

$$= 2L^2(n+1) \sum_{i=k+1}^{n} \| w_s^i - w_s \|^2 + 2\delta^2(n+1) \sum_{i=k+1}^{n} (n-i+1)^2 \| w_s - w_s^{i-1} \|^2$$

$$+ \frac{n+1}{k+1} \left\| \nabla f_{\pi_s^k}(w_s^k) - f_{\pi_s^k}(w_s) \right.$$

$$+ (n-k+1) \cdot \left[\nabla f_{\pi_s^k}(w_s) - \nabla f_{\pi_s^{k-1}}(w_s) - \nabla f_{\pi_s^k}(w_s^{k-1}) + \nabla f_{\pi_s^{k-1}}(w_s^{k-1}) \right]$$

$$+ (n-k+1)\nabla f_{\pi_s^{k-1}}(w_s) + \nabla f_{\pi_s^{k-1}}(w_s^{k-1}) - (n-k+2)\nabla f_{\pi_s^{k-1}}(w_s^{k-2})$$

$$+ \sum_{i=1}^{k-2} \left((n+1-i) \cdot \left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1}) \right] \right) \Big\|^2 .$$

Using $\| a + b \|^2 \leq (1+c)\|a\|^2 + (1+1/c)\|b\|^2$ with $c = k$, we have

$$\left\| \sum_{i=1}^{n} \left((n+1-i) \cdot \left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1}) \right] \right) \right\|^2$$

$$\leq 2L^2(n+1) \sum_{i=k+1}^{n} \| w_s^i - w_s \|^2 + 2\delta^2(n+1) \sum_{i=k+1}^{n} (n-i+1)^2 \| w_s - w_s^{i-1} \|^2$$

$$+ (n+1) \left\| \nabla f_{\pi_s^k}(w_s^k) - f_{\pi_s^k}(w_s) \right.$$

$$+ (n-k+1) \cdot \left[\nabla f_{\pi_s^k}(w_s) - \nabla f_{\pi_s^{k-1}}(w_s) - \nabla f_{\pi_s^k}(w_s^{k-1}) + \nabla f_{\pi_s^{k-1}}(w_s^{k-1}) \right] \Big\|^2$$

$$+ \frac{n+1}{k} \left\| (n-k+1) \nabla f_{\pi_s^{k-1}}(w_s) + \nabla f_{\pi_s^{k-1}}(w_s^{k-1}) - (n-k+2) \nabla f_{\pi_s^{k-1}}(w_s^{k-2}) \right.$$

$$+ \sum_{i=1}^{k-2} \left((n+1-i) \cdot \left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1}) \right] \right) \Big\|^2.$$

With $\| a + b \|^2 \leq 2\| a \|^2 + 2\| b \|^2$ Assumption 1 (δ-similarity (7) and L-smoothness), one can obtain

$$\left\| \sum_{i=1}^{n} \left((n+1-i) \cdot \left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1}) \right] \right) \right\|^2$$

$$\leq 2L^2(n+1) \sum_{i=k+1}^{n} \| w_s^i - w_s \|^2 + 2\delta^2(n+1) \sum_{i=k+1}^{n} (n-i+1)^2 \| w_s - w_s^{i-1} \|^2$$

$$+ 2(n+1) \| \nabla f_{\pi_s^k}(w_s^k) - f_{\pi_s^k}(w_s) \|^2$$

$$+ 2(n+1)(n-k+1)^2 \| \nabla f_{\pi_s^k}(w_s) - \nabla f_{\pi_s^{k-1}}(w_s) - \nabla f_{\pi_s^k}(w_s^{k-1}) + \nabla f_{\pi_s^{k-1}}(w_s^{k-1}) \|^2$$

$$+ \frac{n+1}{k} \left\| (n-k+1) \nabla f_{\pi_s^{k-1}}(w_s) + \nabla f_{\pi_s^{k-1}}(w_s^{k-1}) - (n-k+2) \nabla f_{\pi_s^{k-1}}(w_s^{k-2}) \right.$$

$$+ \sum_{i=1}^{k-2} \left((n+1-i) \cdot \left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1}) \right] \right) \Big\|^2$$

$$\leq 2L^2(n+1) \sum_{i=k}^{n} \| w_s^i - w_s \|^2 + 2\delta^2(n+1) \sum_{i=k}^{n} (n-i+1)^2 \| w_s - w_s^{i-1} \|^2$$

$$+ \frac{n+1}{k} \left\| (n-k+1) \nabla f_{\pi_s^{k-1}}(w_s) + \nabla f_{\pi_s^{k-1}}(w_s^{k-1}) - (n-k+2) \nabla f_{\pi_s^{k-1}}(w_s^{k-2}) \right.$$

$$+ \sum_{i=1}^{k-2} \left((n+1-i) \cdot \left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1}) \right] \right) \Big\|^2.$$

This completes the proof of (13). In particular, (13) with $k = 1$ gives

$$\left\| \sum_{i=1}^{n} \left((n+1-i) \cdot \left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1}) \right] \right) \right\|^2$$

$$\leq 2L^2(n+1) \sum_{i=2}^{n} \|w_s^i - w_s\|^2 + 2\delta^2(n+1) \sum_{i=2}^{n} (n-i+1)^2 \|w_s - w_s^{i-1}\|^2$$

$$+ \frac{n+1}{2} \|(n-1)\nabla f_{\pi_s^1}(w_s) + \nabla f_{\pi_s^1}(w_s^1) - n\nabla f_{\pi_s^1}(w_s^0)\|^2.$$

With (8) and L-smoothness of function $f_{\pi_s^1}$ (Assumption 1 (i)), we have

$$\left\| \sum_{i=1}^{n} \left((n+1-i) \cdot \left[\nabla f_{\pi_s^i}(w_s^i) - \nabla f_{\pi_s^i}(w_s^{i-1}) \right] \right) \right\|^2$$

$$\leq 2L^2(n+1) \sum_{i=2}^{n} \|w_s^i - w_s\|^2 + 2\delta^2(n+1) \sum_{i=2}^{n} (n-i+1)^2 \|w_s - w_s^{i-1}\|^2$$

$$+ \frac{n+1}{2} \|\nabla f_{\pi_s^1}(w_s^1) - \nabla f_{\pi_s^1}(w_s)\|^2 \tag{14}$$

$$\leq 2L^2(n+1) \sum_{i=1}^{n} \|w_s^i - w_s\|^2 + 2\delta^2(n+1) \sum_{i=2}^{n} (n-i+1)^2 \|w_s - w_s^{i-1}\|^2$$

$$\leq \left(2L^2(n+1) + 2\delta^2(n+1)n^2 \right) \sum_{i=1}^{n} \|w_s^i - w_s\|^2.$$

Substituting of (14) to (12) completes proof. $\qquad\square$

Lemma 3 *Under Assumption* 1, *for Algorithms* 1 *and* 2 *with* η *from* (5) *the following holds for* $i \in [n]$

$$\|v_s^i\|^2 \leq \|v_s^{i-1}\|^2.$$

Proof With (9) and (10), we have

$$\begin{aligned}
\|v_s^i\|^2 &= \|v_s^{i-1} + f_{\pi_s^i}(w_s^i) - f_{\pi_s^i}(w_s^{i-1})\|^2 \\
&= \|v_s^{i-1}\|^2 + \|f_{\pi_s^i}(w_s^i) - f_{\pi_s^i}(w_s^{i-1})\|^2 + 2\langle v_s^{i-1}, f_{\pi_s^i}(w_s^i) - f_{\pi_s^i}(w_s^{i-1}) \rangle \\
&= \|v_s^{i-1}\|^2 + \|f_{\pi_s^i}(w_s^i) - f_{\pi_s^i}(w_s^{i-1})\|^2 + \frac{2}{\eta}\langle w_s^{i-1} - w_s^i, f_{\pi_s^i}(w_s^i) - f_{\pi_s^i}(w_s^{i-1}) \rangle.
\end{aligned}$$

Assumption 1 (i) on convexity and L-smoothness of $f_{\pi_s^i}$ gives (see also Theorem 2.1.5 from [30])

$$\|v_s^i\|^2 = \|v_s^{i-1} + f_{\pi_s^i}(w_s^i) - f_{\pi_s^i}(w_s^{i-1})\|^2$$

$$= \|v_s^{i-1}\|^2 + \|f_{\pi_s^i}(w_s^i) - f_{\pi_s^i}(w_s^{i-1})\|^2 + 2\langle v_s^{i-1}, f_{\pi_s^i}(w_s^i) - f_{\pi_s^i}(w_s^{i-1})\rangle$$

$$= \|v_s^{i-1}\|^2 + \|f_{\pi_s^i}(w_s^i) - f_{\pi_s^i}(w_s^{i-1})\|^2 + \frac{2}{\eta}\langle w_s^{i-1} - w_s^i, f_{\pi_s^i}(w_s^i) - f_{\pi_s^i}(w_s^{i-1})\rangle$$

$$\leq \|v_s^{i-1}\|^2 + \left(1 - \frac{2}{L\eta}\right)\|f_{\pi_s^i}(w_s^i) - f_{\pi_s^i}(w_s^{i-1})\|^2.$$

Taking into account that $\eta \leq \frac{1}{8nL} \leq \frac{1}{2L}$, we finishes the proof. $\qquad\square$

Proof of Theorem 2 For $\mathtt{RR-SARAH}$ $v_s = \nabla P(w_s)$, then by Lemma 2, we get

$$\left\|\nabla P(w_s) - \frac{1}{n+1}\sum_{i=0}^{n} v_s^i\right\|^2 \leq \left(\frac{4L^2}{n+1} + 4\delta^2 n\right)\sum_{i=1}^{n}\|w_s^i - w_s\|^2.$$

Combining with Lemma 1, one can obtain

$$P(w_{s+1}) \leq P(w_s) - \frac{\eta(n+1)}{2}\|\nabla P(w_s)\|^2 + \frac{\eta(n+1)}{2}\left(\frac{4L^2}{n+1} + 4\delta^2 n\right)\sum_{i=1}^{n}\|w_s^i - w_s\|^2.$$

Next, we work with $\sum_{i=1}^{n}\|w_s^i - w_s\|^2$. By Lemma 3 and the update for w_s^i ((8) and (10)), we get

$$\sum_{i=1}^{n}\|w_s^i - w_s\|^2 = \eta^2\sum_{i=1}^{n}\left\|\sum_{k=0}^{i-1}v_s^k\right\|^2 \leq \eta^2\sum_{i=1}^{n}i\sum_{k=0}^{i-1}\left\|v_s^k\right\|^2 \leq \eta^2\sum_{i=1}^{n}i\sum_{k=0}^{i-1}\|v_s\|^2$$

$$\leq \eta^2\|v_s\|^2\sum_{i=1}^{n}i\sum_{k=0}^{i-1}1 \leq \eta^2 n^3\|v_s\|^2 = \eta^2 n^3\|\nabla P(w_s)\|^2. \tag{15}$$

Hence,

$$P(w_{s+1}) \leq P(w_s) - \frac{\eta(n+1)}{2}\|\nabla P(w_s)\|^2 + \frac{\eta(n+1)}{2}\left(\frac{4L^2}{n+1} + 4\delta^2 n\right)\cdot \eta^2 n^3\|\nabla P(w_s)\|^2$$

$$\leq P(w_s) - \frac{\eta(n+1)}{2}\left(1 - \left(\frac{4L^2}{n+1} + 4\delta^2 n\right)\cdot \eta^2 n^3\right)\|\nabla P(w_s)\|^2.$$

With $\gamma \leq \min\left\{\frac{1}{8nL}; \frac{1}{8n^2\delta}\right\}$, we get

$$P(w_{s+1}) - P^* \leq P(w_s) - P^* - \frac{\eta(n+1)}{4}\|\nabla P(w_s)\|^2.$$

Strong-convexity of P ends the proof:

$$P(w_{s+1}) - P^* \le \left(1 - \frac{\eta(n+1)\mu}{2}\right)\left(P(w_s) - P^*\right).$$

$\square$

Proof of Theorem 1 For `Shuffled-SARAH` $v_s = \frac{1}{n}\sum_{i=1}^n f_{\pi_{s-1}^i}(w_{s-1}^i)$, then

$$\left\|\nabla P(w_s) - \frac{1}{n}\sum_{i=1}^n v_s^i\right\|^2$$

$$\le \left(\frac{4L^2}{n+1} + 4\delta^2 n\right)\sum_{i=1}^n \|w_s^i - w_s\|^2 + 2\left\|\frac{1}{n}\sum_{i=1}^n \left[f_{\pi_{s-1}^i}(w_s) - f_{\pi_{s-1}^i}(w_{s-1}^i)\right]\right\|^2$$

$$\le \left(\frac{4L^2}{n+1} + 4\delta^2 n\right)\sum_{i=1}^n \|w_s^i - w_s\|^2 + \frac{2L^2}{n}\sum_{i=1}^n \left\|w_{s-1}^i - w_s\right\|^2.$$

(16)

With $\sum_{i=1}^n \|w_s^i - w_s\|^2$ we can work in the same way as in proof of Theorem 2. In remains to deal with $\sum_{i=1}^n \left\|w_{s-1}^i - w_s\right\|^2$. Using Lemma 3 and the update for w_s^i ((8) and (10)), we get

$$\sum_{i=1}^n \|w_{s-1}^i - w_s\|^2 = \eta^2 \sum_{i=1}^n \left\|\sum_{k=1}^{n+1-i} v_{s-1}^{n+1-k}\right\|^2 \le \eta^2 \sum_{i=1}^n (n+1-i)\sum_{k=1}^{n+1-i}\left\|v_{s-1}^{n+1-k}\right\|^2$$

$$\le \eta^2 \sum_{i=1}^n (n+1-i)\sum_{k=1}^{n+1-i}\left\|v_{s-1}\right\|^2$$

$$\le \eta^2 \left\|v_{s-1}\right\|^2 \sum_{i=1}^n (n+1-i)\sum_{k=1}^{n+1-i} 1$$

$$\le \eta^2 n^3 \left\|v_{s-1}\right\|^2.$$

(17)

Combining the results of Lemma 1 with (16), (15) and (17), one can obtain

$$P(w_{s+1}) \le P(w_s) - \frac{\eta(n+1)}{2} \|\nabla P(w_s)\|^2$$
$$+ \frac{\eta(n+1)}{2} \left[\left(\frac{4L^2}{n+1} + 4\delta^2 n \right) \cdot \eta^2 n^3 \|v_s\|^2 + \frac{2L^2}{n} \cdot \eta^2 n^3 \|v_{s-1}\|^2 \right]$$
$$= P(w_s) - \frac{\eta(n+1)}{4} \|\nabla P(w_s)\|^2$$
$$+ \frac{\eta(n+1)}{2} \left[\left(\frac{4L^2}{n+1} + 4\delta^2 n \right) \cdot \eta^2 n^3 \|v_s\|^2 + \frac{2L^2}{n} \cdot \eta^2 n^3 \|v_{s-1}\|^2 \right]$$
$$- \frac{\eta(n+1)}{4} \|\nabla P(w_s)\|^2$$
$$\le P(w_s) - \frac{\eta(n+1)}{4} \|\nabla P(w_s)\|^2$$
$$+ \frac{\eta(n+1)}{2} \left[\left(\frac{4L^2}{n+1} + 4\delta^2 n \right) \cdot \eta^2 n^3 \|v_s\|^2 + \frac{2L^2}{n} \cdot \eta^2 n^3 \|v_{s-1}\|^2 \right]$$
$$- \frac{\eta(n+1)}{8} \|v_s\|^2 + \frac{\eta(n+1)}{4} \|v_s - \nabla P(w_s)\|^2$$
$$\le P(w_s) - \frac{\eta(n+1)}{4} \|\nabla P(w_s)\|^2$$
$$+ \frac{\eta(n+1)}{2} \left[\left(\frac{4L^2}{n+1} + 4\delta^2 n \right) \cdot \eta^2 n^3 \|v_s\|^2 + \frac{2L^2}{n} \cdot \eta^2 n^3 \|v_{s-1}\|^2 \right]$$
$$- \frac{\eta(n+1)}{8} \|v_s\|^2 + \frac{\eta(n+1)}{4} \cdot \frac{2L^2}{n} \cdot \eta^2 n^3 \|v_{s-1}\|^2.$$

The last step is deduced the same way as (17). Small rearrangement gives

$$P(w_{s+1}) - P^* \le P(w_s) - P^* - \frac{\eta(n+1)}{4} \|\nabla P(w_s)\|^2$$
$$- \frac{\eta(n+1)}{8} \left(1 - \left(\frac{16L^2}{n+1} + 16\delta^2 n \right) \cdot \eta^2 n^3 \right) \|v_s\|^2$$
$$+ \eta(n+1) \cdot \frac{2L^2}{n} \cdot \eta^2 n^3 \|v_{s-1}\|^2.$$

With the choice of $\eta \le \min \left\{ \frac{1}{8nL}; \frac{1}{8n^2\delta} \right\}$, we have

$$P(w_{s+1}) - P^* + \frac{\eta(n+1)}{16} \|v_s\|^2$$
$$\le P(w_s) - P^* - \frac{\eta(n+1)}{4} \|\nabla P(w_s)\|^2 + \frac{\eta(n+1)}{16} \cdot \frac{32L^2}{n} \cdot \eta^2 n^3 \|v_{s-1}\|^2.$$

Again using that $\eta \le \frac{1}{8nL}$, we obtain $32L^2\eta^2 n^2 \le \left(1 - \frac{\eta(n+1)\mu}{2} \right)$ and

$$P(w_{s+1}) - P^* + \frac{\eta(n+1)}{16}\|v_s\|^2$$
$$\leq P(w_s) - P^* - \frac{\eta(n+1)}{4}\|\nabla P(w_s)\|^2 + \left(1 - \frac{\eta(n+1)\mu}{2}\right) \cdot \frac{\eta(n+1)}{16}\|v_{s-1}\|^2.$$

Strong-convexity of P ends the proof. $\qquad\square$

References

1. Ahn, K., Yun, C., Sra, S.: SGD with shuffling: optimal rates without component convexity and large epoch requirements. Adv. Neural Inf. Process. Syst. **33**, 17526–17535 (2020)
2. Allen-Zhu, Z.: Katyusha: the first direct acceleration of stochastic gradient methods. In: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, pp. 1200–1205 (2017)
3. Allen-Zhu, Z., Yuan, Y.: Improved SVRG for non-strongly-convex or sum-of-non-convex objectives. In: International Conference on Machine Learning, pp. 1080–1089. PMLR (2016)
4. Bengio, Y.: Practical recommendations for gradient-based training of deep architectures. In: Neural Networks: Tricks of the Trade, 2nd ed., pp. 437–478. Springer (2012)
5. Bottou, L.: Curiously fast convergence of some stochastic gradient descent algorithms. In: Proceedings of the Symposium on Learning and Data Science, Paris, vol. 8, pp. 2624–2633. Citeseer (2009)
6. Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. SIAM Rev. **60**(2), 223–311 (2018)
7. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. TIST **2**(3), 1–27 (2011)
8. Cohen, M., Diakonikolas, J., Orecchia, L.: On acceleration with noise-corrupted gradients. In: International Conference on Machine Learning, pp. 1019–1028. PMLR (2018)
9. Cutkosky, A., Orabona, F.: Momentum-based variance reduction in non-convex sgd. arXiv preprint arXiv:1905.10018 (2019)
10. 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)
11. Fang, C., Li, C.J., Lin, Z., Zhang, T.: Spider: near-optimal non-convex optimization via stochastic path integrated differential estimator. arXiv preprint arXiv:1807.01695 (2018)
12. Ghadimi, S., Lan, G., Zhang, H.: Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. Math. Program. **155**(1–2), 267–305 (2016)
13. Gurbuzbalaban, M., Ozdaglar, A., Parrilo, P.A.: On the convergence rate of incremental aggregated gradient algorithms. SIAM J. Optim. **27**(2), 1035–1048 (2017)
14. Hendrikx, H., Xiao, L., Bubeck, S., Bach, F., Massoulie, L.: Statistically preconditioned accelerated gradient method for distributed optimization. In: International Conference on Machine Learning, pp. 4203–4227. PMLR (2020)
15. Hu, W., Li, C.J., Lian, X., Liu, J., Yuan, H.: Efficient smooth non-convex stochastic compositional optimization via stochastic recursive gradient descent (2019)
16. Huang, X., Yuan, K., Mao, X., Yin, W.: An improved analysis and rates for variance reduction under without-replacement sampling orders. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems, vol. 34, pp. 3232–3243. Curran Associates Inc, Red Hook (2021)

17. Jain, P., Nagaraj, D., Netrapalli, P.: SGD without replacement: sharper rates for general smooth convex functions. arXiv preprint arXiv:1903.01463 (2019)
18. Johnson, R., Zhang, T.: Accelerating stochastic gradient descent using predictive variance reduction. Adv. Neural Inf. Process. Syst. **26**, 315–323 (2013)
19. Khaled, A., Richtárik, P.: Better theory for SGD in the nonconvex world. arXiv preprint arXiv:2002.03329 (2020)
20. Koloskova, A., Doikov, N., Stich, S.U., Jaggi, M.: Shuffle SGD is always better than SGD: improved analysis of SGD with arbitrary data orders. arXiv preprint arXiv:2305.19259 (2023)
21. Li, B., Ma, M., Giannakis, G.B.: On the convergence of Sarah and beyond. In: International Conference on Artificial Intelligence and Statistics, pp. 223–233. PMLR (2020)
22. Li, Z., Bao, H., Zhang, X., Richtárik, P.: Page: a simple and optimal probabilistic gradient estimator for nonconvex optimization. In: International Conference on Machine Learning, pp. 6286–6295. PMLR (2021)
23. Li, Z., Richtárik, P.: Zerosarah: efficient nonconvex finite-sum optimization with zero full gradient computation. arXiv preprint arXiv:2103.01447 (2021)
24. Liu, D., Nguyen, L.M., Tran-Dinh, Q.: An optimal hybrid variance-reduced algorithm for stochastic composite nonconvex optimization. arXiv preprint arXiv:2008.09055 (2020)
25. Mairal, J.: Incremental majorization-minimization optimization with application to large-scale machine learning. SIAM J. Optim. **25**(2), 829–855 (2015)
26. Malinovsky, G., Sailanbayev, A., Richtárik, P.: Random reshuffling with variance reduction: new analysis and better rates. arXiv preprint arXiv:2104.09342 (2021)
27. Mishchenko, K., Khaled Ragab Bayoumi, A., Richtárik, P.: Random reshuffling: simple analysis with vast improvements. Adv. Neural Inf. Process. Syst. **33** (2020)
28. Mokhtari, A., Gurbuzbalaban, M., Ribeiro, A.: Surpassing gradient descent provably: a cyclic incremental method with linear convergence rate. SIAM J. Optim. **28**(2), 1420–1447 (2018)
29. Moulines, E., Bach, F.: Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (eds.) Advances in Neural Information Processing Systems, vol. 24. Curran Associates Inc, Red Hook (2011)
30. Nesterov, Y.: Introductory Lectures on Convex Optimization: A Basic Course, vol. 87. Springer, New York (2003)
31. Nguyen, L.M., Liu, J., Scheinberg, K., Takáč, M.: SARAH: a novel method for machine learning problems using stochastic recursive gradient. In: International Conference on Machine Learning, pp. 2613–2621. PMLR (2017)
32. Nguyen, L.M., Liu, J., Scheinberg, K., Takáč, M.: Stochastic recursive gradient algorithm for nonconvex optimization. arXiv preprint arXiv:1705.07261 (2017)
33. Nguyen, L.M., Nguyen, P.H., Richtárik, P., Scheinberg, K., Takác, M., van Dijk, M.: New convergence aspects of stochastic gradient algorithms. J. Mach. Learn. Res. **20**, 176–1 (2019)
34. Nguyen, L.M., Scheinberg, K., Takáč, M.: Inexact SARAH algorithm for stochastic optimization. Optim. Methods Softw. **36**(1), 237–258 (2021)
35. Nguyen, L.M., Tran-Dinh, Q., Phan, D.T., Nguyen, P.H., Van Dijk, M.: A unified convergence analysis for shuffling-type gradient methods. J. Mach. Learn. Res. **22**(1), 9397–9440 (2021)
36. Park, Y., Ryu, E.K.: Linear convergence of cyclic saga. Optim. Lett. **14**(6), 1583–1598 (2020)
37. Polyak, B.T.: Introduction to optimization
38. Qian, X., Qu, Z., Richtárik, P.: Saga with arbitrary sampling. In: International Conference on Machine Learning, pp. 5190–5199. PMLR (2019)
39. Recht, B., Ré, C.: Parallel stochastic gradient algorithms for large-scale matrix completion. Math. Program. Comput. **5**(2), 201–226 (2013)
40. Robbins, H., Monro, S.: A stochastic approximation method. Ann. Math. Stat. 400–407 (1951)
41. Schmidt, M., Le Roux, N., Bach, F.: Minimizing finite sums with the stochastic average gradient. Math. Program. **162**(1–2), 83–112 (2017)
42. Shalev-Shwartz, S., Ben-David, S.: Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, Cambridge (2014)
43. Shalev-Shwartz, S., Singer, Y., Srebro, N., Cotter, A.: Pegasos: primal estimated sub-gradient solver for SVM. Math. Program. **127**(1), 3–30 (2011)
44. Stich, S.U.: Unified optimal analysis of the (stochastic) gradient method. arXiv preprint arXiv:1907.04232 (2019)
45. Sun, R.Y.: Optimization for deep learning: an overview. J. Oper. Res. Soc. China **8**(2), 249–294 (2020)

46. Sun, T., Sun, Y., Li, D., Liao, Q.: General proximal incremental aggregated gradient algorithms: Better and novel results under general scheme. Adv. Neural Inf. Process. Syst. **32** (2019)
47. Takác, M., Bijral, A., Richtárik, P., Srebro, N.: Mini-batch primal and dual methods for SVMs. In: International Conference on Machine Learning, pp. 1022–1030. PMLR (2013)
48. Tropp, J.A.: User-friendly tail bounds for sums of random matrices. Found. Comput. Math. **12**, 389–434 (2012)
49. Vanli, N.D., Gurbuzbalaban, M., Ozdaglar, A.: A stronger convergence result on the proximal incremental aggregated gradient method. arXiv preprint arXiv:1611.08022 (2016)
50. Vaswani, S., Bach, F., Schmidt, M.: Fast and faster convergence of SGD for over-parameterized models and an accelerated perceptron. In: The 22nd International Conference on Artificial Intelligence and Statistics, pp. 1195–1204. PMLR (2019)
51. Yang, Z., Chen, Z., Wang, C.: Accelerating mini-batch SARAH by step size rules. Inf. Sci. **558**, 157–173 (2021)
52. Ying, B., Yuan, K., Sayed, A.H.: Variance-reduced stochastic learning under random reshuffling. IEEE Trans. Signal Process. **68**, 1390–1408 (2020)
53. Ying, B., Yuan, K., Vlaski, S., Sayed, A.H.: Stochastic learning under random reshuffling with constant step-sizes. IEEE Trans. Signal Process. **67**(2), 474–489 (2018)