VR-SGD: A Simple Stochastic Variance Reduction Method for Machine Learning

Fanhua Shang, *Member, IEEE,* Kaiwen Zhou, Hongying Liu, James Cheng, Ivor W. Tsang, Lijun Zhang, *Member, IEEE,* Dacheng Tao, *Fellow, IEEE,* and Licheng Jiao, *Fellow, IEEE*

Abstract—In this paper, we propose a simple variant of the original SVRG, called variance reduced stochastic gradient descent (VR-SGD). Unlike the choices of snapshot and starting points in SVRG and its proximal variant, Prox-SVRG, the two vectors of VR-SGD are set to the average and last iterate of the previous epoch, respectively. The settings allow us to use much larger learning rates, and also make our convergence analysis more challenging. We also design two different update rules for smooth and non-smooth objective functions, respectively, which means that VR-SGD can tackle non-smooth and/or non-strongly convex problems directly without any reduction techniques. Moreover, we analyze the convergence properties of VR-SGD for strongly convex problems, which show that VR-SGD attains linear convergence. Different from most algorithms that have no convergence guarantees for non-strongly convex problems, we also provide the convergence guarantees of VR-SGD for this case, and empirically verify that VR-SGD with varying learning rates achieves similar performance to its momentum accelerated variant that has the optimal convergence rate $\mathcal{O}(1/T^2)$. Finally, we apply VR-SGD to solve various machine learning problems, such as convex and non-convex empirical risk minimization, and leading eigenvalue computation. Experimental results show that VR-SGD converges significantly faster than SVRG and Prox-SVRG, and usually outperforms state-of-the-art accelerated methods, e.g., Katyusha.

Index Terms—Stochastic optimization, stochastic gradient descent (SGD), variance reduction, empirical risk minimization, strongly convex and non-strongly convex, smooth and non-smooth

1 Introduction

In this paper, we focus on the following composite optimization problem:

$$\min_{x\in\mathbb{R}^d} F(x) \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} f_i(x) + g(x) \tag{1}$$

where $f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$, $f_i(x) : \mathbb{R}^d \to \mathbb{R}$, $i = 1, \ldots, n$ are the smooth functions, and $g(x)$ is a relatively simple (but possibly non-differentiable) convex function (referred to as a regularizer). The formulation (1) arises in many places in machine learning, signal processing, data science, statistics and operations research, such as *regularized empirical risk minimization* (ERM). For instance, one popular choice of the component function $f_i(\cdot)$ in binary classification problems is the logistic loss, i.e., $f_i(x) = \log(1 + \exp(-b_i a_i^T x))$, where $\{(a_1, b_1), \ldots, (a_n, b_n)\}$ is a collection of training examples, and $b_i \in \{\pm 1\}$. Some popular choices for the regularizer include the ℓ_2-norm regularizer (i.e., $g(x) = (\lambda/2)\|x\|^2$), the ℓ_1-norm regularizer (i.e., $g(x) = \lambda\|x\|_1$), and the elastic-net regularizer (i.e., $g(x) = (\lambda_1/2)\|x\|^2 + \lambda_2\|x\|_1$). Some other applications include deep neural networks [1], [2], [3], [4], [5], group Lasso [6], sparse learning and coding [7], [8], [9], [10], phase retrieval [11], matrix completion [12], [13], conditional random fields [14], generalized eigen-decomposition and canonical correlation analysis [15], and eigenvector computation [16], [17] such as principal component analysis (PCA) and singular value decomposition (SVD).

1.1 Stochastic Gradient Descent

We are especially interested in developing efficient algorithms to solve Problem (1) involving the sum of a large number of component functions. The standard and effective method for solving (1) is the (proximal) gradient descent (GD) method, including Nesterov's accelerated gradient descent (AGD) [18], [19] and accelerated proximal gradient (APG) [20], [21]. For the *smooth* problem (1), GD takes the following update rule: starting with x_0, and for any $k \geq 0$

$$x_{k+1} = x_k - \eta_k \left[\frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x_k) + \nabla g(x_k)\right] \tag{2}$$

where $\eta_k > 0$ is commonly referred to as the learning rate in machine learning or step-size in optimization. When $g(\cdot)$ is *non-smooth* (e.g., the ℓ_1-norm regularizer), we typically introduce the following proximal operator to replace (2),

$$x_{k+1} = \text{Prox}_{\eta_k}^{g}(y_k) := \arg\min_{x\in\mathbb{R}^d} \left\{\frac{1}{2\eta_k}\|x - y_k\|^2 + g(x)\right\} \tag{3}$$

- *F. Shang, H. Liu (Corresponding author) and L. Jiao are with the Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, School of Artificial Intelligence, Xidian University, China. E-mails: {fhshang, hyliu, lchjiao}@xidian.edu.cn.*
- *K. Zhou and J. Cheng are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. E-mails: {kwzhou, jcheng}@cse.cuhk.edu.hk.*
- *I.W. Tsang is with the Centre for Artificial Intelligence, University of Technology Sydney, Ultimo, NSW 2007, Australia. E-mail: Ivor.Tsang@uts.edu.au.*
- *L. Zhang is with the National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China. E-mail: zlj@nju.edu.cn.*
- *D. Tao is with the UBTECH Sydney Artificial Intelligence Centre and the School of Information Technologies in the Faculty of Engineering and Information Technologies, The University of Sydney, Darlington, NSW 2008, Australia. E-mail: dacheng.tao@sydney.edu.au.*

Manuscript received December 19, 2017.

where $y_k = x_k - (\eta_k/n)\sum_{i=1}^{n}\nabla f_i(x_k)$. GD has been proven to achieve linear convergence for *strongly convex* problems, and both AGD and APG attain the optimal convergence rate $\mathcal{O}(1/T^2)$ for *non-strongly convex* problems, where T denotes the number of iterations. However, the per-iteration cost of all the batch (or deterministic) methods is $O(nd)$, which is expensive for very large n.

Instead of evaluating the full gradient of $f(\cdot)$ at each iteration, an efficient alternative is the stochastic (or incremental) gradient descent (SGD) method [22]. SGD only evaluates the gradient of a single component function at each iteration, and has much *lower* per-iteration cost, $O(d)$. Thus, SGD has been successfully applied to many large-scale learning problems [23], [24], [25], especially training for deep learning models [2], [3], [26], and its update rule is

$$x_{k+1} = x_k - \eta_k[\nabla f_{i_k}(x_k) + \nabla g(x_k)] \tag{4}$$

where $\eta_k \propto 1/\sqrt{k}$, and the index i_k can be chosen uniformly at random from $\{1, 2, \ldots, n\}$. Although the expectation of the stochastic gradient estimator $\nabla f_{i_k}(x_k)$ is an *unbiased* estimation for $\nabla f(x_k)$, i.e., $\mathbb{E}[\nabla f_{i_k}(x_k)] = \nabla f(x_k)$, the variance of $\nabla f_{i_k}(x_k)$ may be large due to the variance of random sampling [1]. Thus, stochastic gradient estimators are also called "noisy gradients", and we need to gradually reduce its step size, which leads to slow convergence. In particular, even under the strongly convex (SC)condition, standard SGD attains a slower *sub-linear* convergence rate $\mathcal{O}(1/T)$ [27].

1.2 Accelerated SGD

Recently, many SGD methods with *variance reduction* have been proposed, such as stochastic average gradient (SAG) [28], stochastic variance reduced gradient (SVRG) [1], stochastic dual coordinate ascent (SDCA) [29], SAGA [30], stochastic primal-dual coordinate (SPDC) [31], and their proximal variants, such as Prox-SAG [32], Prox-SVRG [33] and Prox-SDCA [34]. These accelerated SGD methods can use a constant learning rate η instead of diminishing step sizes for SGD, and fall into the following three categories: *primal* methods such as SVRG and SAGA, *dual* methods such as SDCA, and *primal-dual* methods such as SPDC. In essence, many of the primal methods use the full gradient at the snapshot $\widetilde{x}$ or the average gradient to progressively reduce the variance of stochastic gradient estimators, as well as the dual and primal-dual methods, *which leads to a revolution in the area of first-order optimization* [35]. Thus, they are also known as the hybrid gradient descent method [36] or semi-stochastic gradient descent method [37]. In particular, under the strongly convex condition, most of the accelerated SGD methods enjoy a linear convergence rate (also known as a geometric or exponential rate) and the oracle complexity of $\mathcal{O}((n+L/\mu)\log(1/\epsilon))$ to obtain an ϵ-suboptimal solution, where each $f_i(\cdot)$ is L-smooth, and $F(\cdot)$ is μ-strongly convex. The complexity bound shows that they converge faster than accelerated deterministic methods, whose oracle complexity is $\mathcal{O}(n\sqrt{L/\mu}\log(1/\epsilon))$ [38], [39].

SVRG [1] and its proximal variant, Prox-SVRG [33], are particularly attractive because of their *low storage* requirement compared with other methods such as SAG, SAGA and SDCA, which require storage of all the gradients of component functions or dual variables. At the beginning of the s-th epoch in SVRG, the full gradient $\nabla f(\widetilde{x}^{s-1})$ is computed at the snapshot $\widetilde{x}^{s-1}$, which is updated periodically.

Definition 1. *The stochastic variance reduced gradient estimator is independently introduced in [1], [36] as follows:*

$$\widetilde{\nabla} f_{i_k^s}(x_k^s) = \nabla f_{i_k^s}(x_k^s) - \nabla f_{i_k^s}(\widetilde{x}^{s-1}) + \nabla f(\widetilde{x}^{s-1}), \tag{5}$$

where s is the epoch that iteration k belongs to.

It is not hard to verify that the variance of the SVRG estimator $\widetilde{\nabla} f_{i_k^s}(x_k^s)$ (i.e., $\mathbb{E}\|\widetilde{\nabla} f_{i_k^s}(x_k^s) - \nabla f(x_k^s)\|^2$) can be much smaller than that of the SGD estimator $\nabla f_{i_k}(x_k^s)$ (i.e., $\mathbb{E}\|\nabla f_{i_k}(x_k^s) - \nabla f(x_k^s)\|^2$). Theoretically, for *non-strongly convex* (Non-SC) problems, the variance reduced methods converge slower than the accelerated batch methods such as FISTA [21], i.e., $\mathcal{O}(1/T)$ vs. $\mathcal{O}(1/T^2)$.

More recently, many *acceleration* techniques were proposed to further speed up the stochastic variance reduced methods mentioned above. These techniques mainly include the Nesterov's acceleration techniques in [24], [38], [39], [40], [41], reducing the number of gradient calculations in early iterations [35], [42], [43], the projection-free property of the conditional gradient method (also known as the Frank-Wolfe algorithm [44]) as in [45], the stochastic sufficient decrease technique [46], and the momentum acceleration tricks in [35], [47], [48]. [39] proposed an accelerating Catalyst framework and achieved the oracle complexity of $\mathcal{O}((n+\sqrt{nL/\mu})\log(L/\mu)\log(1/\epsilon))$ for strongly convex problems. [47] and [49] proved that the accelerated methods can attain the oracle complexity of $\mathcal{O}(n\log(1/\epsilon)+\sqrt{nL/\epsilon})$ for non-strongly convex problems. The overall complexity matches the theoretical upper bound provided in [50]. Katyusha [47], point-SAGA [51] and MiG [49] achieve the best-known oracle complexity of $\mathcal{O}((n+\sqrt{nL/\mu})\log(1/\epsilon))$ for strongly convex problems, which is identical to the upper complexity bound in [50]. Hence, Katyusha and MiG are the *best-known* stochastic optimization method for both *SC* and *Non-SC* problems, as pointed out in [50]. However, selecting the best values for the parameters in the accelerated methods (e.g., the momentum parameter) is still an open problem. In particular, most of accelerated stochastic variance reduction methods, including Katyusha, require at least one auxiliary variable and one momentum parameter, which lead to complicated algorithm design and high per-iteration complexity, especially for very high-dimensional and sparse data.

1.3 Our Contributions

From the above discussions, we can see that most of the accelerated stochastic variance reduction methods such as [35], [38], [43], [45], [46], [47], [52], [53] and applications such as [7], [9], [10], [13], [16], [17] are based on the SVRG method [1]. Thus, any key improvement on SVRG is very important for the research of stochastic optimization. In this paper, we propose a simple variant of the original SVRG [1], called *variance reduced stochastic gradient descent* (*VR-SGD*). The snapshot point and starting point of each epoch in VR-SGD are set to the *average* and *last iterate* of the previous epoch, respectively. This is different from the settings of SVRG and Prox-SVRG [33], where the two points of the former are set to be the last iterate, and those of the latter are

TABLE 1
Comparison of convergence rates of VR-SGD and its counterparts.

	SVRG [1]	Prox-SVRG [33]	VR-SGD
SC, smooth	linear rate	unknown	linear rate
SC, non-smooth	unknown	linear rate	linear rate
Non-SC, smooth	unknown	unknown	$\mathcal{O}(1/T)$
Non-SC, non-smooth	unknown	unknown	$\mathcal{O}(1/T)$

set to be the average of the previous epoch. This difference makes the convergence analysis of VR-SGD significantly more *challenging* than that of SVRG and Prox-SVRG. Our empirical results show that the performance of VR-SGD is significantly better than its counterparts, SVRG and Prox-SVRG. Impressively, VR-SGD with varying learning rates achieves better or at least comparable performance with accelerated methods, such as Catalyst [39] and Katyusha [47]. The main contributions of this paper are summarized below.

- The snapshot and starting points of VR-SGD are set to two different vectors, i.e., $\widetilde{x}^s = \frac{1}{m}\sum_{k=1}^{m} x_k^s$ (Option I) or $\widetilde{x}^s = \frac{1}{m-1}\sum_{k=1}^{m-1} x_k^s$ (Option II), and $x_0^{s+1} = x_m^s$. In particular, we find that the settings of VR-SGD allow us to take much *larger* learning rates than SVRG, e.g., $1/L$ vs. $1/(10L)$, and thus significantly speed up its convergence in practice. Moreover, VR-SGD has an advantage over SVRG in terms of *robustness* of learning rate selection.
- Unlike *proximal* stochastic gradient methods, e.g., Prox-SVRG and Katyusha, which have a unified update rule for the two cases of *smooth* and *non-smooth* objectives (see Section 2.2 for details), VR-SGD employs two different update rules for the two cases, respectively, as in (12) and (13) below. Empirical results show that gradient update rules as in (12) for *smooth* optimization problems are better choices than proximal update formulas as in (10).
- We provide the convergence guarantees of VR-SGD for solving smooth/non-smooth and *non-strongly convex* (or general convex) functions. In comparison, SVRG and Prox-SVRG do not have any convergence guarantees, as shown in Table 1.
- Moreover, we also present a momentum accelerated variant of VR-SGD, discuss their equivalent relationship, and empirically verify that they achieve similar performance to their variant that attains the optimal convergence rate $\mathcal{O}(1/T^2)$.
- Finally, we theoretically analyze the convergence properties of VR-SGD with Option I or Option II for smooth/non-smooth and *strongly convex* functions, which show that VR-SGD attains *linear* convergence.

2 Preliminary and Related Work

Throughout this paper, we use $\|\cdot\|$ to denote the ℓ_2-norm (also known as the standard Euclidean norm), and $\|\cdot\|_1$ is the ℓ_1-norm, i.e., $\|x\|_1 = \sum_{i=1}^{d}|x_i|$. $\nabla f(\cdot)$ denotes the full gradient of $f(\cdot)$ if it is differentiable, or $\partial f(\cdot)$ the subgradient if $f(\cdot)$ is only Lipschitz continuous. For each epoch $s \in [S]$ and inner iteration $k \in \{0, 1, \ldots, m-1\}$, $i_k^s \in [n]$ is the random chosen index. We mostly focus on the case of Problem (1)

Algorithm 1 SVRG (Option I) and Prox-SVRG (Option II)

Input: The number of epochs S, the number of iterations m per epoch, and the learning rate η.
Initialize: $\widetilde{x}^0$.

```
for s = 1, 2, ..., S do
    μ̃^s = (1/n) Σ_{i=1}^n ∇f_i(x̃^{s-1}),  x_0^s = x̃^{s-1};
    for k = 0, 1, ..., m − 1 do
        Pick i_k^s uniformly at random from [n];
        ∇̃f_{i_k^s}(x_k^s) = ∇f_{i_k^s}(x_k^s) − ∇f_{i_k^s}(x̃^{s-1}) + μ̃^s;
        Option I: x_{k+1}^s = x_k^s − η[∇̃f_{i_k^s}(x_k^s) + ∇g(x_k^s)],
                  or x_{k+1}^s = Prox_η^g(x_k^s − η∇̃f_{i_k^s}(x_k^s));
        Option II:
        x_{k+1}^s = arg min_{y∈R^d} {g(y) + y^T ∇̃f_{i_k^s}(x_k^s) + (1/2η)‖y − x_k^s‖^2};
    end for
    Option I: x̃^s = x_m^s;          // Last iterate for snapshot x̃
    Option II: x̃^s = (1/m) Σ_{k=1}^m x_k^s;   // Iterate averaging for x̃
end for
```

Output: $\widetilde{x}^S$

when each $f_i(\cdot)$ is L-smooth[1], and $F(\cdot)$ is μ-strongly convex. The two common assumptions are defined as follows.

2.1 Basic Assumptions

Assumption 1 (Smoothness). *Each $f_i(\cdot)$ is L-smooth, that is, there exists a constant $L > 0$ such that for all $x, y \in \mathbb{R}^d$,*

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|. \tag{6}$$

Assumption 2 (Strong Convexity). *$F(x)$ is μ-strongly convex, i.e., there exists a constant $\mu > 0$ such that for all $x, y \in \mathbb{R}^d$,*

$$F(y) \geq F(x) + \langle \nabla F(x), y - x\rangle + \frac{\mu}{2}\|x - y\|^2. \tag{7}$$

Note that when $g(\cdot)$ is *non-smooth*, the inequality in (7) needs to be revised by simply replacing the gradient $\nabla F(x)$ with an arbitrary sub-gradient of $F(\cdot)$ at x. In contrast, for a non-strongly convex or *general convex* function, the inequality in (7) can always be satisfied with $\mu = 0$.

2.2 Related Work

To speed up standard and proximal SGD, many stochastic variance reduced methods [28], [29], [30], [36] have been proposed for some special cases of Problem (1). In the case when each $f_i(x)$ is L-smooth, $f(x)$ is μ-strongly convex, and $g(x) \equiv 0$, Roux *et al.* [28] proposed a stochastic average gradient (SAG) method, which attains linear convergence. However, SAG, as well as other incremental aggregated gradient methods such as SAGA [30], needs to store all gradients, so that $O(nd)$ memory is required in general [42]. Similarly, SDCA [29] requires storage of all dual variables [1], which uses $O(n)$ memory. In contrast, SVRG proposed by Johnson and Zhang [1], as well as Prox-SVRG [33], has the similar convergence rate to SAG and SDCA, but without the memory requirements of all gradients and dual variables. In particular, the SVRG estimator in (5) may be the most

1. In fact, we can extend the theoretical results for the case, when the gradients of all component functions have the same Lipschitz constant L, to the more general case, when some component functions $f_i(\cdot)$ have different degrees of smoothness.

popular choice for *stochastic gradient estimators*. The update rule of SVRG for the case of Problem (1) when $g(\cdot)\equiv 0$ is

$$x_{k+1}^s = x_k^s - \eta\widetilde{\nabla} f_{i_k^s}(x_k^s). \tag{8}$$

When the smooth regularizer $g(\cdot)\neq 0$, the update rule in (8) becomes: $x_{k+1}^s = x_k^s - \eta[\widetilde{\nabla} f_{i_k^s}(x_k^s) + \nabla g(x_k^s)]$. Although the original SVRG in [1] only has convergence guarantees for the special case of Problem (1), when each $f_i(x)$ is L-smooth, $f(x)$ is μ-strongly convex, and $g(x)\equiv 0$, we can extend SVRG to the proximal setting by introducing the proximal operator in (3), as shown in Line 7 of Algorithm 1.

Based on the SVRG estimator in (5), some accelerated algorithms [38], [39], [47] have been proposed. The proximal update rules of Katyusha [47] are formulated as follows:

$$x_{k+1}^s = w_1 y_k^s + w_2\widetilde{x}^{s-1} + (1-w_1-w_2)z_k^s, \tag{9a}$$

$$y_{k+1}^s = \arg\min_{y\in\mathbb{R}^d}\left\{\frac{1}{2\eta}\|y-y_k^s\|^2 + y^T\widetilde{\nabla} f_{i_k^s}(x_{k+1}^s) + g(y)\right\}, \tag{9b}$$

$$z_{k+1}^s = \arg\min_{z\in\mathbb{R}^d}\left\{\frac{3L}{2}\|z-x_{k+1}^s\|^2 + z^T\widetilde{\nabla} f_{i_k^s}(x_{k+1}^s) + g(z)\right\} \tag{9c}$$

where $w_1, w_2\in[0,1]$ are two parameters. To eliminate the need for parameter tuning, η is set to $1/(3w_1L)$, and w_2 is fixed to 0.5 in [47]. In addition, [15], [16], [17] applied efficient stochastic solvers to compute leading eigenvectors of a symmetric matrix or generalized eigenvectors of two symmetric matrices. The first such method is VR-PCA proposed by Shamir [16], and the convergence properties of VR-PCA for such a non-convex problem are also provided. Garber *et al.* [17] analyzed the convergence rate of SVRG when $f(\cdot)$ is a convex function that is a sum of non-convex component functions. Moreover, [4], [5] and [54] proved that SVRG and SAGA with minor modifications can converge asymptotically to a stationary point of non-convex functions. Some parallel and distributed variants [55], [56], [57] of accelerated SGD methods have also been proposed.

An important class of stochastic methods is the *proximal stochastic gradient* (Prox-SG) method, such as Prox-SVRG [33], SAGA [30], and Katyusha [47]. Different from standard variance reduction SGD methods such as SVRG, the Prox-SG method has a unified update rule for both smooth and non-smooth cases of $g(\cdot)$. For instance, the update rule of Prox-SVRG [33] is formulated as follows:

$$x_{k+1}^s = \arg\min_{y\in\mathbb{R}^d}\left\{g(y) + y^T\widetilde{\nabla} f_{i_k^s}(x_k^s) + \frac{1}{2\eta}\|y-x_k^s\|^2\right\}. \tag{10}$$

For the sake of completeness, the details of Prox-SVRG [33] are shown in Algorithm 1 with Option II. When $g(\cdot)$ is the widely used ℓ_2-norm regularizer, i.e., $g(\cdot) = (\lambda_1/2)\|\cdot\|^2$, the proximal update formula in (10) becomes

$$x_{k+1}^s = \frac{1}{1+\lambda_1\eta}\left[x_k^s - \eta\widetilde{\nabla} f_{i_k^s}(x_k^s)\right]. \tag{11}$$

3 Variance Reduced SGD

In this section, we propose an efficient variance reduced stochastic gradient descent (VR-SGD) algorithm, as shown in Algorithm 2. Different from the choices of the snapshot and starting points in SVRG [1] and Prox-SVRG [33], the two vectors of each epoch in VR-SGD are set to the average and last iterate of the previous epoch, respectively. Moreover, unlike existing proximal stochastic methods, we design two different update rules for smooth and non-smooth objective functions, respectively.

3.1 Snapshot and Starting Points

Like SVRG, VR-SGD is also divided into S epochs, and each epoch consists of m stochastic gradient steps, where m is usually chosen to be $\Theta(n)$, as suggested in [1], [33], [47]. Within each epoch, we need to compute the full gradient $\nabla f(\widetilde{x}^s)$ at the snapshot $\widetilde{x}^s$ and use it to define the variance reduced stochastic gradient estimator $\widetilde{\nabla} f_{i_k^s}(x_k^s)$ in (5). Unlike SVRG, whose snapshot is set to the last iterate of the previous epoch, the *snapshot* $\widetilde{x}^s$ of VR-SGD is set to the *average* of the previous epoch, e.g., $\widetilde{x}^s = \frac{1}{m}\sum_{k=1}^m x_k^s$ in Option I of Algorithm 2, which leads to better robustness to gradient noise[2], as also suggested in [35], [46], [61]. In fact, the choice of Option II in Algorithm 2, i.e., $\widetilde{x}^s = \frac{1}{m-1}\sum_{k=1}^{m-1} x_k^s$, also works well in practice, as shown in Fig. 2 in the Supplementary Material. Therefore, we provide the convergence guarantees for our algorithm with either Option I or Option II in the next section. In particular, we find that one of the effects of the choice in Option I or Option II of Algorithm 2 is to allow taking much larger learning rates or step sizes than SVRG in practice, e.g., $1/L$ for VR-SGD vs. $1/(10L)$ for SVRG (see Fig. 1). Actually, a larger learning rate enjoyed by VR-SGD means that the variance of its stochastic gradient estimator goes asymptotically to zero faster.

Unlike Prox-SVRG [33] whose starting point is initialized to the average of the previous epoch, the starting point of VR-SGD is set to the last iterate of the previous epoch. That is, in VR-SGD, the last iterate of the previous epoch becomes the new starting point, while the two points of Prox-SVRG are completely different, thereby leading to relatively slow convergence in general. Both the starting and snapshot points of SVRG [1] are set to the last iterate of the previous epoch[3], while the two points of Prox-SVRG [33] are set to the average of the previous epoch (also suggested in [1]). By setting the starting and snapshot points in VR-SGD to the two different vectors mentioned above, the convergence analysis of VR-SGD becomes significantly more challenging than that of SVRG and Prox-SVRG, as shown in Section 4.

3.2 The VR-SGD Algorithm

In this part, we propose an efficient VR-SGD algorithm to solve Problem (1), as outlined in **Algorithm 2** for the case of smooth objective functions. It is well known that the original SVRG [1] only works for the case of smooth minimization problems. However, in many machine learning applications,

2. It should be emphasized that the noise introduced by random sampling is inevitable, and generally slows down the convergence speed in this sense. However, SGD and its variants are probably the mostly used optimization algorithms for deep learning [58]. In particular, [59] has shown that by adding gradient noise at each step, noisy gradient descent can escape the saddle points efficiently and converge to a local minimum of the non-convex minimization problem, e.g., the application of deep neural networks in [60].

3. Note that the theoretical convergence of the original SVRG [1] relies on its Option II, i.e., both $\widetilde{x}^s$ and x_0^{s+1} are set to x_k^s, where k is randomly chosen from $\{1,2,\ldots,m\}$. However, the empirical results in [1] suggest that Option I is a better choice than its Option II, and the convergence guarantee of SVRG with Option I for strongly convex objective functions is provided in [62].

Algorithm 2 VR-SGD for solving smooth problems

Input: The number of epochs S, and the number of iterations m per epoch.

Initialize: $x_0^1 = \widetilde{x}^0$, and $\{\eta_s\}$.

1: **for** $s = 1, 2, \ldots, S$ **do**
2: $\widetilde{\mu}^s = \frac{1}{n}\sum_{i=1}^{n}\nabla f_i(\widetilde{x}^{s-1})$; *// Compute the full gradient*
3: **for** $k = 0, 1, \ldots, m-1$ **do**
4: Pick i_k^s uniformly at random from $[n]$;
5: $\widetilde{\nabla} f_{i_k^s}(x_k^s) = \nabla f_{i_k^s}(x_k^s) - \nabla f_{i_k^s}(\widetilde{x}^{s-1}) + \widetilde{\mu}^s$;
6: $x_{k+1}^s = x_k^s - \eta_s[\widetilde{\nabla} f_{i_k^s}(x_k^s) + \nabla g(x_k^s)]$;
7: **end for**
8: Option I: $\widetilde{x}^s = \frac{1}{m}\sum_{k=1}^{m} x_k^s$; *// Iterate averaging for $\widetilde{x}$*
9: Option II: $\widetilde{x}^s = \frac{1}{m-1}\sum_{k=1}^{m-1} x_k^s$; *// Iterate averaging for $\widetilde{x}$*
10: $x_0^{s+1} = x_m^s$; *// Initiate x_0^{s+1} for the next epoch*
11: **end for**

Output: $\widehat{x}^S = \widetilde{x}^S$, if $F(\widetilde{x}^S) \leq F(\frac{1}{S}\sum_{s=1}^{S}\widetilde{x}^s)$, and $\widehat{x}^S = \frac{1}{S}\sum_{s=1}^{S}\widetilde{x}^s$ otherwise.

e.g., elastic net regularized logistic regression, the strongly convex objective function $F(x)$ is non-smooth. To solve this class of problems, the proximal variant of SVRG, Prox-SVRG [33], was subsequently proposed. Unlike the original SVRG, VR-SGD can not only solve *smooth* objective functions, but also directly tackle *non-smooth* ones. That is, when the regularizer $g(x)$ is smooth (e.g., the ℓ_2-norm regularizer), the key update rule of VR-SGD is

$$x_{k+1}^s = x_k^s - \eta_s[\widetilde{\nabla} f_{i_k^s}(x_k^s) + \nabla g(x_k^s)]. \tag{12}$$

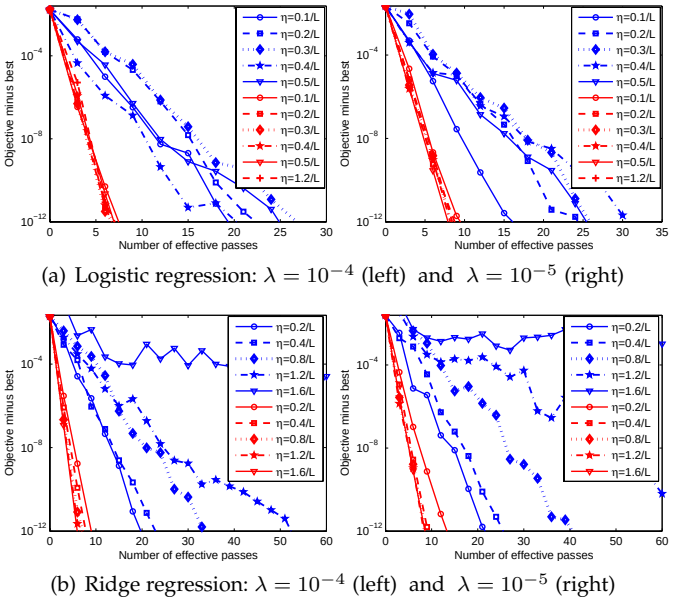
When $g(x)$ is non-smooth (e.g., the ℓ_1-norm regularizer), the key update rule of VR-SGD in **Algorithm** 2 becomes

$$x_{k+1}^s = \mathrm{Prox}_{\eta_s}^{g}\left(x_k^s - \eta_s\widetilde{\nabla} f_{i_k^s}(x_k^s)\right). \tag{13}$$

Unlike the proximal stochastic methods such as Prox-SVRG [33], all of which have a unified update rule as in (10) for both the smooth and non-smooth cases of $g(\cdot)$, VR-SGD has two different update rules for the two cases, as in (12) and (13). Fig. 1 demonstrates that VR-SGD has a significant advantage over SVRG in terms of robustness of learning rate selection. That is, VR-SGD yields good performance within the range of the learning rate from $0.2/L$ to $1.2/L$, whereas the performance of SVRG is very *sensitive* to the selection of learning rates. Thus, VR-SGD is convenient to be applied in various real-world problems of machine learning. In fact, VR-SGD can use much larger learning rates than SVRG for ridge regression problems in practice, e.g., $8/(5L)$ for VR-SGD vs. $1/(5L)$ for SVRG, as shown in Fig. 1(b).

3.3 VR-SGD for Non-Strongly Convex Objectives

Although many stochastic variance reduced methods have been proposed, most of them, including SVRG and Prox-SVRG, only have convergence guarantees for the case of Problem (1), when $F(x)$ is strongly convex. However, $F(x)$ may be non-strongly convex in many machine learning applications, such as Lasso and ℓ_1-norm regularized logistic regression. As suggested in [47], [63], this class of problems can be transformed into strongly convex ones by adding a proximal term $(\tau/2)\|x - x_0^s\|^2$, which can be efficiently solved by Algorithm 2. However, the reduction technique



(a) Logistic regression: $\lambda = 10^{-4}$ (left) and $\lambda = 10^{-5}$ (right)

(b) Ridge regression: $\lambda = 10^{-4}$ (left) and $\lambda = 10^{-5}$ (right)

Fig. 1. Comparison of SVRG [1] and VR-SGD with different learning rates for solving ℓ_2-norm regularized logistic regression and ridge regression on Covtype. Note that the *blue* lines stand for the results of SVRG, while the *red* lines correspond to the results of VR-SGD (best viewed in colors).

may degrade the performance of the involved algorithms both in theory and in practice [43]. Thus, we use VR-SGD to directly solve *non-strongly convex* problems.

The learning rate η_s of Algorithm 2 can be fixed to a constant. Inspired by existing accelerated stochastic algorithms [35], [47], the learning rate in Algorithm 2 can be gradually increased in early iterations for both strongly convex and non-strongly convex problems, which leads to faster convergence (see Fig. 3 in the Supplementary Material). Different from SGD and Katyusha [47], where the learning rate of the former requires to be gradually decayed and that of the latter needs to be gradually increased, the update rule of η_s in **Algorithm** 2 is defined as follows: η_0 is an initial learning rate, and for any $s \geq 1$,

$$\eta_s = \eta_0 / \max\{\alpha,\, 2/(s+1)\} \tag{14}$$

where $0 < \alpha \leq 1$ is a given constant, e.g., $\alpha = 0.2$.

3.4 Extensions of VR-SGD

It has been shown in [37], [38] that *mini-batching* can effectively decrease the variance of stochastic gradient estimates. Therefore, we first extend the proposed VR-SGD method to the mini-batch setting, as well as its convergence results below. Here, we denote by b the mini-batch size and I_k^s the selected random index set $I_k \subset [n]$ for each outer-iteration $s \in [S]$ and inner-iteration $k \in \{0, 1, \ldots, m-1\}$.

Definition 2. *The stochastic variance reduced gradient estimator in the mini-batch setting is defined as*

$$\widetilde{\nabla} f_{I_k^s}(x_k^s) = \frac{1}{b}\sum_{i \in I_k^s}\left[\nabla f_i(x_k^s) - \nabla f_i(\widetilde{x}^{s-1})\right] + \nabla f(\widetilde{x}^{s-1}) \tag{15}$$

where $I_k^s \subset [n]$ is a mini-batch of size b.

If some component functions are non-smooth, we can use the proximal operator oracle [63] or the Nesterov's

smoothing [64] and homotopy smoothing [65] techniques to smoothen them, and thereby obtain their smoothed approximations. In addition, we can directly extend our VR-SGD method to the non-smooth setting as in [35] (e.g., Algorithm 3 in [35]) without using any smoothing techniques.

Considering that each component function $f_i(x)$ may have different degrees of smoothness, picking the random index i_k^s from a non-uniform distribution is a much better choice than commonly used uniform random sampling [66], [67], as well as without-replacement sampling vs. with-replacement sampling [68]. This can be done using the same techniques in [33], [47], i.e., the sampling probabilities for all $f_i(x)$ are proportional to their Lipschitz constants, i.e., $p_i = L_i / \sum_{j=1}^{n} L_j$. VR-SGD can also be combined with other accelerated techniques used for SVRG. For instance, the epoch length of VR-SGD can be automatically determined by the techniques in [43], [69], instead of a fixed epoch length. We can reduce the number of gradient calculations in early iterations as in [42], [43], which leads to faster convergence in general (see Section 5.5 for details). Moreover, we can introduce the Nesterov's acceleration techniques in [24], [38], [39], [40], [41] and momentum acceleration tricks in [35], [47], [70] to further improve the performance of VR-SGD.

4 Algorithm Analysis

In this section, we provide the convergence guarantees of VR-SGD for solving both smooth and non-smooth general convex problems, and extend the results to the mini-batch setting. We also study the convergence properties of VR-SGD for solving both smooth and non-smooth strongly convex objective functions. Moreover, we discuss the equivalent relationship between VR-SGD and its momentum accelerated variant, as well as some of its extensions.

4.1 Convergence Properties: Non-strongly Convex

In this part, we analyze the convergence properties of VR-SGD for solving more general non-strongly convex problems. Considering that the proposed algorithm (i.e., Algorithm 2) has two different update rules for smooth and non-smooth cases, we give the convergence guarantees of VR-SGD for the two cases as follows.

4.1.1 Smooth Objective Functions

We first provide the convergence guarantee of our algorithm for solving Problem (1) when $F(x)$ is *smooth*. In order to simplify analysis, we denote $F(x)$ by $f(x)$, that is, $f_i(x) := f_i(x) + g(x)$ for all $i = 1, 2, \ldots, n$, and then $g(x) \equiv 0$.

Lemma 1 (Variance bound). *Let x^* be the optimal solution of Problem (1). Suppose Assumption 1 holds. Then the following inequality holds*

$$\mathbb{E}\Big[\|\widetilde{\nabla} f_{i_k^s}(x_k^s) - \nabla f(x_k^s)\|^2\Big] \leq 4L[f(x_k^s) - f(x^*) + f(\widetilde{x}^{s-1}) - f(x^*)].$$

The proofs of this lemma, the lemmas and theorems below are all included in the Supplementary Material. Lemma 1 provides the upper bound on the expected variance of the variance reduced gradient estimator in (5), i.e., the SVRG estimator. For Algorithm 2 with Option II and a fixed learning rate η, we have the following result.

Theorem 1 (Smooth objectives). *Suppose Assumption 1 holds. Then the following inequality holds*

$$\mathbb{E}\Big[f(\widehat{x}^S)\Big] - f(x^*) \leq \frac{2(m+1)}{[\gamma - 4m + 2]S}[f(\widetilde{x}^0) - f(x^*)] + \frac{\beta(\beta-1)L}{2[\gamma - 4m + 2]S}\|\widetilde{x}^0 - x^*\|^2$$

where $\gamma = (\beta - 1)(m - 1)$, and $\beta = 1/(L\eta)$.

From Theorem 1 and its proof, one can see that our convergence analysis is very different from that of existing stochastic methods, such as SVRG [1], Prox-SVRG [33], and SVRG++ [43]. Similarly, the convergence of Algorithm 2 with Option I and a fixed learning rate can be guaranteed, as stated in Theorem 6 in the Supplementary Material. All the results show that VR-SGD attains a convergence rate of $\mathcal{O}(1/T)$ for non-strongly convex functions.

4.1.2 Non-Smooth Objective Functions

We also provide the convergence guarantee of Algorithm 2 with Option I and (13) for solving Problem (1) when $F(x)$ is *non-smooth* and non-strongly convex, as shown below.

Theorem 2 (Non-smooth objectives). *Suppose Assumption 1 holds. Then the following inequality holds*

$$\mathbb{E}\Big[F(\widehat{x}^S)\Big] - F(x^*) \leq \frac{2(m+1)}{(\beta-5)mS}[F(\widetilde{x}^0) - F(x^*)] + \frac{\beta(\beta-1)L}{2(\beta-5)mS}\|\widetilde{x}^0 - x^*\|^2.$$

Similarly, the convergence of Algorithm 2 with Option II and a fixed learning rate can be guaranteed, as stated in Corollary 4 in the Supplementary Material.

4.1.3 Mini-Batch Settings

The upper bound on the variance of $\widetilde{\nabla} f_{i_k^s}(x_k^s)$ is extended to the *mini-batch setting* as follows.

Corollary 1 (Variance bound of mini-batch). *If each $f_i(\cdot)$ is convex and L-smooth, then the following inequality holds*

$$\mathbb{E}\Big[\|\widetilde{\nabla} f_{I_k^s}(x_k^s) - \nabla f(x_k^s)\|^2\Big] \leq 4L\delta(b)[F(x_k^s) - F(x^*) + F(\widetilde{x}^{s-1}) - F(x^*)]$$

where $\delta(b) = (n-b)/[(n-1)b]$.

This corollary is essentially identical to Theorem 4 in [37], and hence its proof is omitted. It is not hard to verify that $0 \leq \delta(b) \leq 1$. Based on the variance upper bound, we further analyze the convergence properties of VR-SGD in the mini-batch setting, as shown below.

Theorem 3 (Mini-batch). *If each $f_i(\cdot)$ is convex and L-smooth, then the following inequality holds*

$$\mathbb{E}\Big[F(\widehat{x}^S)\Big] - F(x^*) \leq \frac{2\delta(b)(m+1)}{\zeta mS}\mathbb{E}\big[F(\widetilde{x}^0) - F(x^*)\big] + \frac{\beta(\beta-1)L}{2\zeta mS}\mathbb{E}\big[\|x^* - \widetilde{x}^0\|^2\big]$$

where $\zeta = \beta - 1 - 4\delta(b)$.

From Theorem 3, one can see that when $b = n$ (i.e., the batch setting), $\delta(n) = 0$, and the first term on the right-hand side of the above inequality diminishes. That is, VR-SGD degenerates to a batch method. When $b = 1$, we have $\delta(1) = 1$, and thus Theorem 3 degenerates to Theorem 2.

4.2 Convergence Properties: Strongly Convex

We also analyze the convergence properties of VR-SGD for solving *strongly convex* problems. We first give the following convergence result for Algorithm 2 with Option II.

Theorem 4 (Strongly convex). *Suppose Assumptions 1, 2 and 3 in the Supplementary Material hold, and m is sufficiently large so that*

$$\rho := \frac{2L\eta(m+c)}{(m-1)(1-3L\eta)} + \frac{c(1-L\eta)}{\mu\eta(m-1)(1-3L\eta)} < 1$$

where c is a constant. Then Algorithm 2 with Option II has the following geometric convergence in expectation:

$$\mathbb{E}\left[F(\widehat{x}^S) - F(x^*)\right] \leq \rho^S\left[F(\widetilde{x}^0) - F(x^*)\right].$$

We also provide the linear convergence guarantees for Algorithm 2 with Option I for solving non-smooth and strongly convex functions, as stated in Theorem 7 in the Supplementary Material. Similarly, the linear convergence of Algorithm 2 can be guaranteed for the smooth strongly-convex case. All the theoretical results show that VR-SGD attains a *linear* convergence rate and at most the oracle complexity of $\mathcal{O}((n+L/\mu)\log(1/\epsilon))$ for both *smooth* and *non-smooth* strongly convex functions. In contrast, the convergence of SVRG [1] is only guaranteed for smooth and strongly convex problems.

Although the learning rate in Theorem 4 needs to be less than $1/(3L)$, we can use much larger learning rates in practice, e.g., $\eta = 1/L$. However, it can be easily verified that the learning rate of SVRG should be less than $1/(4L)$ in theory, and adopting a larger learning rate for SVRG is not always helpful in practice, which means that VR-SGD can use much larger learning rates than SVRG both in theory and in practice. In other words, although they have the same theoretical convergence rate, VR-SGD converges significantly faster than SVRG in practice, as shown by our experiments. Note that similar to the convergence analysis in [4], [5], [54], the convergence of VR-SGD for some non-convex problems can also be guaranteed.

4.3 Equivalent to Its Momentum Accelerated Variant

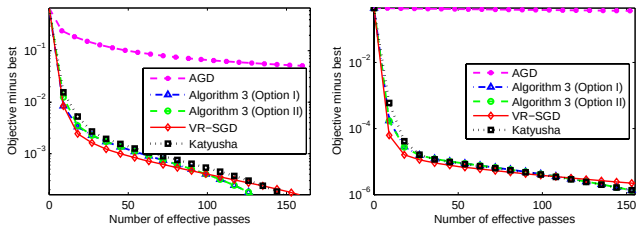
Inspired by the success of the momentum technique in our previous work [6], [49], [70], we present a momentum accelerated variant of Algorithm 2, as shown in Algorithm 3. Unlike existing momentum techniques, e.g., [18], [21], [24], [38], [39], [47], we use the convex combination of the snapshot $\widetilde{x}^{s-1}$ and latest iterate v_k^s for acceleration, i.e., $\widetilde{x}^{s-1} + w_s(v_{k+1}^s - \widetilde{x}^{s-1}) = w_s v_k^s + (1-w_s)\widetilde{x}^{s-1}$. It is not hard to verify that Algorithm 2 with Option I is equivalent to its variant (i.e., Algorithm 3 with Option I), when $w_s = \max\{\alpha, 2/(s+1)\}$ and α is sufficiently small (see the Supplementary Material for their equivalent analysis). We emphasize that the only difference between Options I and II in Algorithm 3 is the initialization of x_0^s and v_0^s.

Theorem 5. *Suppose Assumption 1 holds. Then the following inequality holds:*

$$\mathbb{E}[F(\widehat{x}^S) - F(x^*)] \leq \frac{4(1-w_1)}{w_1^2(S+1)^2}[F(\widetilde{x}^0) - F(x^*)] + \frac{2}{m\eta_0(S+1)^2}\|x^* - \widetilde{x}^0\|^2.$$

Algorithm 3 The momentum accelerated algorithm

Input: S and m.
Initialize: $x_0^1 = v_0^1 = \widetilde{x}^0$, $\{w_s\}$, $\alpha > 0$, and η_0.
for $s = 1, 2, \ldots, S$ **do**
  $\widetilde{\mu}^s = \frac{1}{n}\sum_{i=1}^n \nabla f_i(\widetilde{x}^{s-1})$, $\eta_s = \eta_0/\max\{\alpha, 2/(s+1)\}$;
  Option I: $v_0^s = x_0^s$, or Option II: $x_0^s = w_s v_0^s + (1-w_s)\widetilde{x}^{s-1}$;
  for $k = 0, 1, \ldots, m-1$ **do**
    Pick i_k^s uniformly at random from $[n]$;
    $\widetilde{\nabla} f_{i_k^s}(x_k^s) = \nabla f_{i_k^s}(x_k^s) - \nabla f_{i_k^s}(\widetilde{x}^{s-1}) + \widetilde{\mu}^s$;
    $v_{k+1}^s = v_k^s - \eta_s[\widetilde{\nabla} f_{i_k^s}(x_k^s) + \nabla g(x_k^s)]$;
    $x_{k+1}^s = \widetilde{x}^{s-1} + w_s(v_{k+1}^s - \widetilde{x}^{s-1})$;
  end for
  $\widetilde{x}^s = \frac{1}{m}\sum_{k=1}^m x_k^s$;
  Option I: $x_0^{s+1} = x_m^s$, or Option II: $v_0^{s+1} = v_m^s$;
end for
Output: $\widehat{x}^S = \widetilde{x}^S$, if $F(\widetilde{x}^S) \leq F(\frac{1}{S}\sum_{s=1}^S \widetilde{x}^s)$, and $\widehat{x}^S = \frac{1}{S}\sum_{s=1}^S \widetilde{x}^s$ otherwise.



(a) Small dataset: MNIST (b) Large dataset: Epsilon

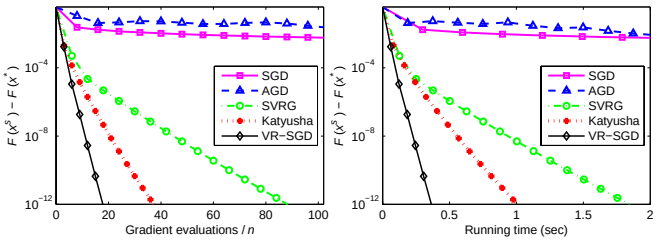
Fig. 2. Comparison of AGD [19], Katyusha [47], Algorithm 3 with Option I and II, and VR-SGD for solving logistic regression with $\lambda = 0$.

Choosing $m = \Theta(n)$, Algorithm 3 with Option II achieves an ϵ-suboptimal solution (i.e., $\mathbb{E}[F(\widehat{x}^S)] - F(x^*) \leq \varepsilon$) using at most $\mathcal{O}(n\sqrt{[F(\widetilde{x}^0) - F(x^*)]/\varepsilon} + \sqrt{nL/\varepsilon}\|\widetilde{x}^0 - x^*\|)$ iterations.

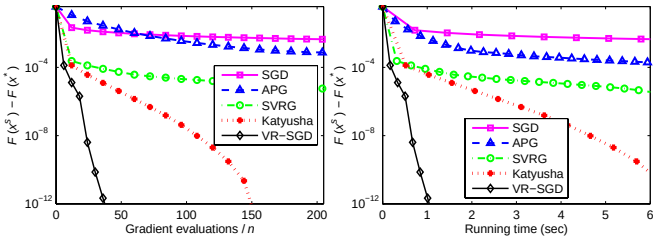
This theorem shows that the oracle complexity of Algorithm 3 with Option II is consistent with that of Katyusha [47], and is better than that of accelerated deterministic methods (e.g., AGD [19]), (i.e., $\mathcal{O}(n\sqrt{L/\varepsilon})$), which are also verified by the experimental results in Fig. 2. Our algorithm also achieves the optimal convergence rate $\mathcal{O}(1/T^2)$ for non-strongly convex functions as in [47], [48]. Fig. 2 shows that Katyusha and Algorithm 3 with Option II have similar performance as Algorithms 2 and 3 with Option I ($\eta_0 = 3/(5L)$) in terms of number of effective passes. Clearly, Algorithm 3 and Katyusha have higher complexity per iteration than Algorithm 2. Thus, we only report the results of VR-SGD (i.e., Algorithm 2) in Section 5.

4.4 Complexity Analysis

From Algorithm 2, we can see that the per-iteration cost of VR-SGD is dominated by the computation of $\nabla f_{i_k^s}(x_k^s)$, $\nabla f_{i_k^s}(\widetilde{x}^{s-1})$, and $\nabla g(x_k^s)$ or the proximal update in (13). Thus, the complexity is $O(d)$, which is as low as that of SVRG [1] and Prox-SVRG [33]. In fact, for some ERM problems, we can save the intermediate gradients $\nabla f_i(\widetilde{x}^{s-1})$ in the computation of $\widetilde{\mu}^s$, which generally requires $O(n)$ additional storage. As a result, each epoch only requires $(n+m)$ component gradient evaluations. In addition, for extremely sparse data, we can introduce the lazy update tricks in [37], [71], [72] to our algorithm, and perform the update steps in (12) and (13) only for the non-zero dimensions of each sample,



(a) ℓ_2-norm regularized logistic regression: $\lambda = 10^{-5}$



(b) ℓ_1-norm regularized logistic regression: $\lambda = 10^{-5}$

Fig. 3. Comparison of deterministic and stochastic methods on Adult.

rather than all dimensions. In other words, the per-iteration complexity of VR-SGD can be improved from $O(d)$ to $O(d')$, where $d' \leq d$ is the sparsity of feature vectors. Moreover, VR-SGD has a much lower per-iteration complexity than existing accelerated stochastic variance reduction methods such as Katyusha [47], which have more updating steps for additional variables, as shown in (9a)-(9c).

5 Experimental Results

In this section, we evaluate the performance of VR-SGD for solving a number of convex and non-convex ERM problems (such as logistic regression, Lasso and ridge regression), and compare its performance with several state-of-the-art stochastic variance reduced methods (including SVRG [1], Prox-SVRG [33], SAGA [30]) and accelerated methods, such as Catalyst [39] and Katyusha [47]. Moreover, we apply VR-SGD to solve other machine learning problems, such as ERM with non-convex loss and leading eigenvalue computation.

5.1 Experimental Setup

We used several publicly available data sets in the experiments: Adult (also called a9a), Covtype, Epsilon, MNIST, and RCV1, all of which can be downloaded from the LIBSVM Data website[4]. *It should be noted that each sample of these date sets was normalized so that they have unit length as in [33], [35], which leads to the same upper bound on the Lipschitz constants L_i, i.e., $L = L_i$ for all $i = 1, \ldots, n$.* As suggested in [1], [33], [47], the epoch length is set to $m = 2n$ for the stochastic variance reduced methods, SVRG [1], Prox-SVRG [33], Catalyst [39], and Katyusha [47], as well as VR-SGD. Then the only parameter we have to tune by hand is the learning rate, η. More specifically, we select learning rates from $\{10^j, 2.5 \times 10^j, 5 \times 10^j, 7.5 \times 10^j, 10^{j+1}\}$, where $j \in \{-2, -1, 0\}$. Since Katyusha has a much higher per-iteration complexity than SVRG and VR-SGD, we compare their performance in terms of both the number of effective passes and running time (seconds), where computing a single full gradient or evaluating n component gradients is considered as one effective pass over the data. For fair comparison, we implemented SVRG, Prox-SVRG, SAGA, Catalyst, Katyusha, and VR-SGD in C++ with a Matlab interface, as well as their sparse versions with lazy update tricks, and performed all the experiments on a PC with an Intel i5-4570 CPU and 16GB RAM. The source code of all the methods is available at https://github.com/jnhujnhu/VR-SGD.

4. https://www.csie.ntu.edu.tw/~cjlin/libsvm/

5.2 Deterministic Methods vs. Stochastic Methods

In this subsection, we compare the performance of stochastic methods (including SGD, SVRG, Katyusha, and VR-SGD) with that of deterministic methods such as AGD [18], [19] and APG [21] for solving strongly and non-strongly convex problems. Note that the important momentum parameter w of AGD is $w = (\sqrt{L} - \sqrt{\mu})/(\sqrt{L} + \sqrt{\mu})$ as in [73], while that of APG is defined as follows: $w_k = (\alpha_k - 1)/\alpha_{k+1}$ for all $k \geq 1$ [21], where $\alpha_{k+1} = (1 + \sqrt{1 + 4\alpha_k^2})/2$, and $\alpha_1 = 1$.

Fig. 3 shows the the objective gap (i.e., $F(x^s) - F(x^*)$) of those deterministic and stochastic methods for solving ℓ_2-norm and ℓ_1-norm regularized logistic regression problems (see the Supplementary Material for more results). It can be seen that the accelerated deterministic methods and SGD have similar convergence speed, and APG usually performs slightly better than SGD for non-strongly convex problems. The variance reduction methods (e.g., SVRG, Katyusha and VR-SGD) significantly outperform the accelerated deterministic methods and SGD for both strongly and non-strongly convex cases, suggesting the importance of variance reduction techniques. Although accelerated deterministic methods have a faster theoretical speed than SVRG for general convex problems, as discussed in Section 1.2, APG converges much slower in practice. VR-SGD consistently outperforms the other methods (including Katyusha) in all the settings, which verifies the effectiveness of VR-SGD.

5.3 Different Choices for Snapshot and Starting Points

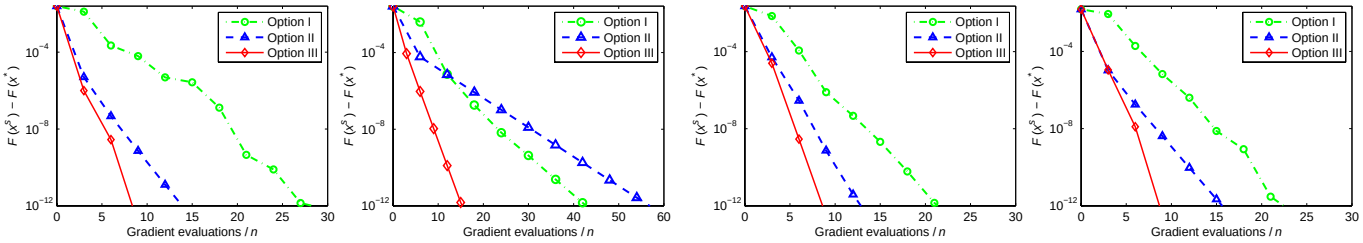
In the practical implementation of SVRG [1], both the snapshot $\widetilde{x}^s$ and starting point x_0^{s+1} in each epoch are set to the last iterate x_m^s of the previous epoch (i.e., Option I in Algorithm 1), while the two vectors in [33] are set to the average point of the previous epoch, $\frac{1}{m}\sum_{k=1}^{m} x_k^s$ (i.e., Option II in Algorithm 1). In contrast, $\widetilde{x}^s$ and x_0^{s+1} in our algorithm are set to $\frac{1}{m}\sum_{k=1}^{m} x_k^s$ and x_m^s (denoted by Option III, i.e., Option I[5] in Algorithm 2), respectively.

We compare the performance of the algorithms with the three settings (i.e., the Options I, II and III listed in Table 2) for solving ridge regression and Lasso problems, as shown in Fig. 4 (see the Supplementary Material for more results). Except for the three different settings for snapshot and starting points, we use the update rules in (12) and (13) for ridge regression and Lasso problems, respectively. We can see that the algorithm with Option III (i.e., Algorithm 2 with Option I) consistently converges much faster than the algorithms with Options I and II for both strongly convex and non-strongly convex cases. This indicates that the setting of Option III suggested in this paper is a better choice than Options I and II for stochastic optimization.

5. As Options I and II in Algorithm 2 achieve very similar performance, we only report the results of our algorithm with Option I.

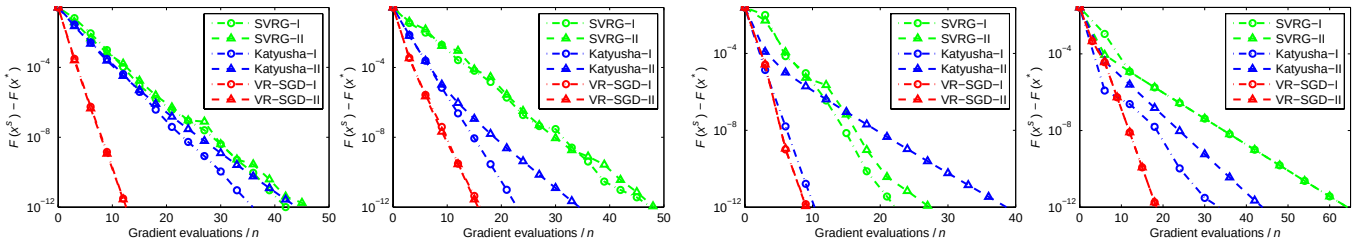
TABLE 2
The three choices of snapshot and starting points for stochastic variance reduction optimization.

Option I	Option II	Option III
$\widetilde{x}^s = x_m^s$ and $x_0^{s+1} = x_m^s$	$\widetilde{x}^s = \frac{1}{m}\sum_{k=1}^m x_k^s$ and $x_0^{s+1} = \frac{1}{m}\sum_{k=1}^m x_k^s$	$\widetilde{x}^s = \frac{1}{m}\sum_{k=1}^m x_k^s$ and $x_0^{s+1} = x_m^s$



(a) Ridge regression: $\lambda = 10^{-5}$ (left) and $\lambda = 10^{-6}$ (right) (b) Lasso: $\lambda = 10^{-4}$ (left) and $\lambda = 10^{-5}$ (right)

Fig. 4. Comparison of the algorithms with Options I, II, and III for solving ridge regression and Lasso on Covtype. In each plot, the vertical axis shows the objective value minus the minimum, and the horizontal axis denotes the number of effective passes.



(a) Adult: $\lambda = 10^{-3}$ (left) and $\lambda = 10^{-4}$ (right) (b) Covtype: $\lambda = 10^{-5}$ (left) and $\lambda = 10^{-6}$ (right)

Fig. 5. Comparison of SVRG [1], Katyusha [47], VR-SGD and their proximal versions for solving ridge regression problems. In each plot, the vertical axis shows the objective value minus the minimum, and the horizontal axis is the number of effective passes over data.

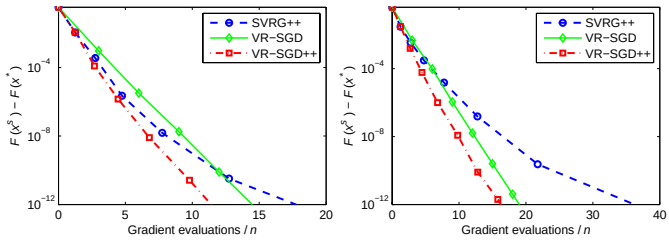
5.4 Common SG Updates vs. Prox-SG Updates

In this subsection, we compare the original Katyusha algorithm in [47] with the slightly modified Katyusha algorithm (denoted by Katyusha-I). In Katyusha-I, only the following two update rules are used to replace the original proximal stochastic gradient update rules in (9b) and (9c).

$$\begin{aligned} y_{k+1}^s &= y_k^s - \eta[\widetilde{\nabla} f_{i_k}(x_{k+1}^s) + \nabla g(x_{k+1}^s)], \\ z_{k+1}^s &= x_{k+1}^s - [\widetilde{\nabla} f_{i_k}(x_{k+1}^s) + \nabla g(x_{k+1}^s)]/(3L). \end{aligned} \tag{16}$$

Similarly, we also implement the proximal versions[6] for the original SVRG (called SVRG-I) and VR-SGD (denoted by VR-SGD-I) methods, and denote their proximal variants by SVRG-II and VR-SGD-II, respectively. In addition, the original Katyusha is denoted by Katyusha-II.

Fig. 5 shows the performance of Katyusha-I and Katyusha-II for solving ridge regression on the two popular data sets: Adult and Covtype. We also report the results of SVRG, VR-SGD, and their proximal variants. It is clear that Katyusha-I usually performs better than Katyusha-II (i.e., the original Katyusha [47]), and converges significantly faster in the case when the regularization parameter is 10^{-4} or 10^{-6}. This seems to be the main reason why Katyusha has inferior performance when the regularization parameter is relatively large, as shown in Section 5.6.1. In contrast, VR-SGD and its proximal variant have similar performance,



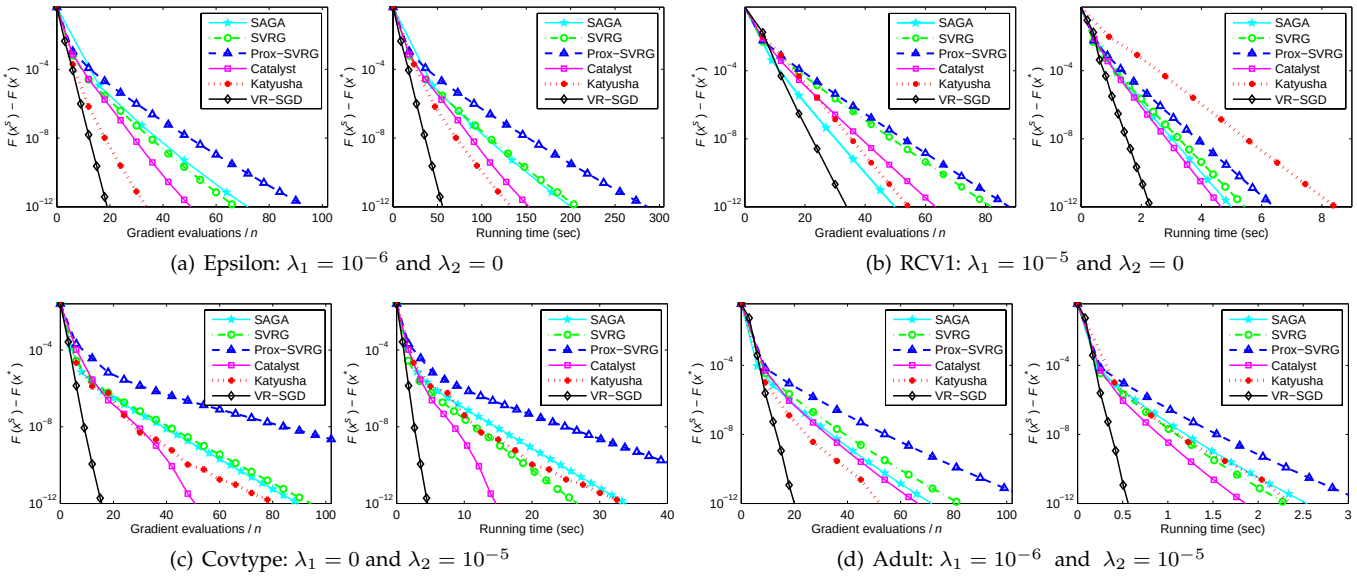
(a) $\lambda = 10^{-5}$ (b) $\lambda = 10^{-6}$

Fig. 6. Comparison of SVRG++ [43], VR-SGD and VR-SGD++ for solving logistic regression problems on Epsilon.

and the former slightly outperforms the latter in most cases (similar results are also observed for SVRG vs. its proximal variant). This suggests that stochastic gradient update rules as in (12) and (16) are better choices than proximal update rules as in (10), (9b) and (9c) for smooth objective functions. We also believe that our new insight can help us to design accelerated stochastic optimization methods.

Both Katyusha-I and Katyusha-II usually outperform SVRG and its proximal variant, especially when the regularization parameter is relatively small, e.g., $\lambda = 10^{-6}$. Moreover, it can be seen that both VR-SGD and its proximal variant achieve much better performance than the other methods in most cases, and are also comparable to Katyusha-I and Katyusha-II in the remaining cases. This further verifies that VR-SGD is suitable for various large-scale machine learning.

6. Here, the proximal variant of SVRG is different from Prox-SVRG [33], and their main difference is the choices of both the snapshot point and starting point. That is, the two vectors of the former are set to the last iterate x_m^s, while those of Prox-SVRG are set to the average point of the previous epoch, i.e., $\frac{1}{m}\sum_{k=1}^m x_k^s$.



(a) Epsilon: $\lambda_1 = 10^{-6}$ and $\lambda_2 = 0$

(b) RCV1: $\lambda_1 = 10^{-5}$ and $\lambda_2 = 0$

(c) Covtype: $\lambda_1 = 0$ and $\lambda_2 = 10^{-5}$

(d) Adult: $\lambda_1 = 10^{-6}$ and $\lambda_2 = 10^{-5}$

Fig. 7. Comparison of SAGA [30], SVRG [1], Prox-SVRG [33], Catalyst [39], Katyusha [47], and VR-SGD for solving ℓ_2-norm (the first row), ℓ_1-norm (c), and elastic net (d) regularized logistic regression problems. In each plot, the vertical axis shows the objective value minus the minimum, and the horizontal axis is the number of effective passes (left) or running time (right).

5.5 Growing Epoch Size Strategy in Early Iterations

In this subsection, we present a general growing epoch size strategy in early iterations (i.e., If $m_s < 2n$, $m_{s+1} = \lfloor \rho m_s \rfloor$ with the factor $\rho > 1$. Otherwise, $m_{s+1} = m_s$). Different from the doubling-epoch technique used in SVRG++ [43] (i.e., $m_{s+1} = 2m_s$), we gradually increase the epoch size in only the early iterations. Similar to the convergence analysis in Section 4, VR-SGD with the growing epoch size strategy (called VR-SGD++) can be guaranteed to converge. As suggested in [43], we set $m_1 = \lfloor n/4 \rfloor$ for both SVRG++ and VR-SGD++, and $\rho = 1.75$ for VR-SGD++. Note that they use the same initial learning rate. We compare their performance for solving ℓ_2-norm regularized logistic regression, as shown in Fig. 6 (see the Supplementary Material for more results). All the results show that VR-SGD++ converges faster than VR-SGD, which means that reducing the number of gradient calculations in early iterations can lead to faster convergence as discussed in [42]. Moreover, both VR-SGD++ and VR-SGD significantly outperform SVRG++, especially when the regularization parameter is relatively small, e.g., $\lambda = 10^{-6}$.

5.6 Real-world Applications

In this subsection, we apply VR-SGD to solve a number of machine learning problems, e.g., logistic regression, non-convex ERM and eigenvalue computation.

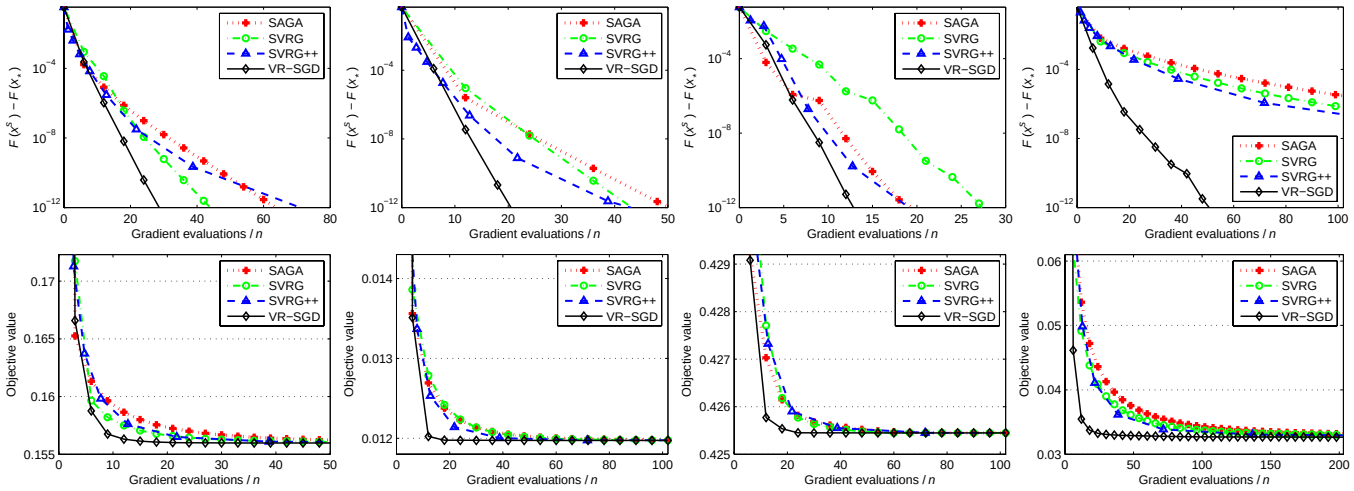
5.6.1 *Convex Logistic Regression*

In this part, we focus on the following generalized logistic regression problem for binary classification,

$$\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-b_i a_i^T x)) + \frac{\lambda_1}{2} \|x\|^2 + \lambda_2 \|x\|_1 \quad (17)$$

where $\{(a_i, b_i)\}$ is a set of training examples, and $\lambda_1, \lambda_2 \geq 0$ are the regularization parameters. Note that when $\lambda_2 > 0$, $f_i(x) = \log(1 + \exp(-b_i a_i^T x)) + (\lambda_1/2)\|x\|^2$. The formulation (17) includes the ℓ_2-norm (i.e., $\lambda_2 = 0$), ℓ_1-norm (i.e., $\lambda_1 = 0$), and elastic net (i.e., $\lambda_1 \neq 0$ and $\lambda_2 \neq 0$) regularized logistic regression problems. Fig. 7 shows how the objective gap decreases for the ℓ_2-norm, ℓ_1-norm, and elastic net regularized logistic regression problems, respectively (see the Supplementary Material for more results). From all the results, we make the following observations.

- When the problems are well-conditioned (e.g., $\lambda_1 = 10^{-4}$ or $\lambda_2 = 10^{-4}$), Prox-SVRG usually converges faster than SVRG for both strongly convex (e.g., ℓ_2-norm regularized logistic regression) and non-strongly convex (e.g., ℓ_1-norm regularized logistic regression) cases. On the contrary, SVRG often outperforms Prox-SVRG, when the problems are ill-conditioned, e.g., $\lambda_1 = 10^{-6}$ or $\lambda_2 = 10^{-6}$ (see Figs. 11 and 12 in the Supplementary Material). The main reason is that they have different initialization settings, i.e., $\widetilde{x}^s = x_m^s$ and $x_0^{s+1} = x_m^s$ for SVRG vs. $\widetilde{x}^s = \frac{1}{m}\sum_{k=1}^{m} x_k^s$ and $x_0^{s+1} = \frac{1}{m}\sum_{k=1}^{m} x_k^s$ for Prox-SVRG.
- Katyusha converges much faster than SAGA, SVRG, Prox-SVRG, and Catalyst in the cases when the problems are ill-conditioned, e.g., $\lambda_1 = 10^{-6}$, whereas it often achieves similar or inferior performance when the problems are well-conditioned, e.g., $\lambda_1 = 10^{-4}$ (see Figs. 11, 12 and 13 in the Supplementary Material). Note that we implemented the original algorithm with Option I in [47] for Katyusha. Obviously, the above observation matches the convergence properties of Katyusha provided in [47], that is, only if $m\mu/L \leq 3/4$, Katyusha attains the oracle complexity of $\mathcal{O}((n + \sqrt{nL/\mu}) \log(1/\epsilon))$ for strongly convex problems.
- VR-SGD converges significantly faster than SAGA, SVRG, Prox-SVRG and Catalyst, especially when the problems are ill-conditioned, e.g., $\lambda_1 = 10^{-6}$ or $\lambda_2 = 10^{-6}$ (see Figs. 11 and 12 in the Supplementary Material). The main reason is that VR-SGD can use much larger learning rates than them (e.g., $1/L$ for



(a) Adult (b) MNIST (c) Covtype (d) RCV1

Fig. 8. Comparison of SAGA [54], SVRG [4], SVRG++ [43], and VR-SGD for solving non-convex ERM problems with sigmoid loss: $\lambda = 10^{-5}$ (top) and $\lambda = 10^{-6}$ (bottom). Note that x_* denotes the best solution obtained by running all those methods for a large number of iterations and multiple random initializations.

VR-SGD vs. $1/(10L)$ for SVRG), which leads to faster convergence. This further verifies that the settings of both snapshot and starting points in our algorithm (i.e., Algorithm 2) are better choices than Options I and II in Algorithm 1.

- In particular, VR-SGD consistently outperforms the best-known stochastic method, Katyusha, in terms of the number of passes through the data, especially when the problems are well-conditioned, e.g., 10^{-4} and 10^{-5} (see Figs. 11 and 12 in the Supplementary Material). Since VR-SGD has a much lower per-iteration complexity than Katyusha, VR-SGD has more obvious advantage over Katyusha in terms of running time, especially in the case of sparse data (e.g., RCV1), as shown in Fig. 7(b). From the algorithms of Katyusha proposed in [47], we can see that the learning rate of Katyusha is at least set to $1/(3L)$. Similarly, the learning rate used in VR-SGD is comparable to that of Katyusha, which may be the main reason why the performance of VR-SGD is much better than that of Katyusha. This also implies that the algorithms (including VR-SGD) that enjoy larger learning rates can converge faster in general.

5.6.2 ERM with Non-Convex Loss

In this part, we apply VR-SGD to solve the following regularized ERM problem with non-convex sigmoid loss:

$$\min_{x\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} f_i(x) + \frac{\lambda}{2}\|x\|^2 \tag{18}$$

where $f_i(x) = 1/[1+\exp(b_i a_i^T x)]$. Some work [4], [74] has shown that the sigmoid function usually generalizes better than some other loss functions (such as squared loss, logistic loss and hinge loss) in terms of test accuracy especially when there are outliers. Here, we consider binary classification on the four data sets: Adult, MNIST, Covtype and RCV1. Note that we only consider classifying the first class in MNIST.

We compare the performance (including training objective value and function suboptimality, i.e., $F(x^s) - F(x_*)$) of VR-SGD with that of SAGA [54], SVRG [4], and SVRG++ [43], as shown in Fig. 8 (more results are provided in the Supplementary Material), where x_* denotes the best solution obtained by running all those methods for a large number of iterations and multiple random initializations. Note that both SAGA and SVRG are two variants of the original SAGA [30] and SVRG [1]. The results show that our VR-SGD method has faster convergence than the other methods, and its objective value is much lower. This implies that VR-SGD can yield much better solutions than the other methods including SVRG++. Furthermore, we can see that VR-SGD has much greater advantage over the other methods in the cases when the smaller λ is, which means that the objective function becomes more "non-convex".

Moreover, we report the classification testing accuracies of all those methods on the test sets of Adult and MNIST in Fig. 9, as the number of effective passes over datasets increases. Note that the regularization parameter is set to $\lambda = 10^{-4}$. It can be seen that our VR-SGD method obtains higher test accuracies than the other methods with much shorter running time, suggesting faster convergence.

5.6.3 Eigenvalue Computation

Finally, we apply VR-SGD to solve the following non-convex leading eigenvalue computation problem:

$$\min_{x\in\mathbb{R}^d: x^T x=1} -x^T\left(\frac{1}{n}\sum_{i=1}^{n} a_i a_i^T\right)x. \tag{19}$$

We plot the performance of the classical Power iteration method, VR-PCA [16], and VR-SGD on Epsilon and RCV1 in Fig. 10, where the relative error is defined as in [16], i.e., $\log_{10}(1 - \|A^T x\|^2/(\max_{u:u^T u=1}\|A^T u\|^2))$, and $A \in \mathbb{R}^{d\times n}$ is the data matrix. Note that the epoch length is set to $m = n$ for VR-PCA and VR-SGD, as suggested in [16], and both of them use a constant learning rate. The results show that the stochastic variance reduced methods, VR-PCA and VR-SGD, significantly outperform the traditional method, Power. Moreover, our VR-SGD method often converges much faster than VR-PCA.

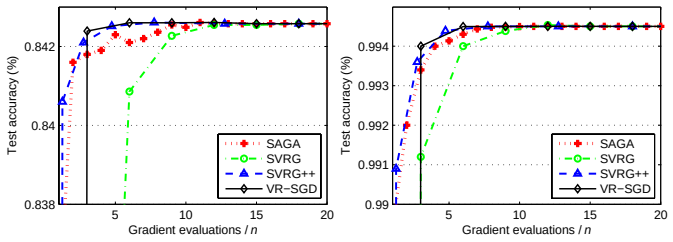


Fig. 9. Testing accuracy comparison on Adult (left) and MNIST (right).

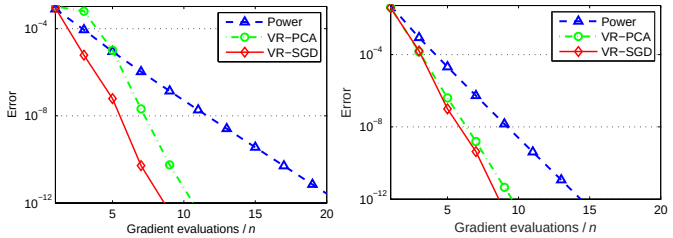


Fig. 10. Relative error comparison of leading eigenvalue computation on RCV1 (left) and Epsilon (right).

6 Conclusions

We proposed a simple variant of the original SVRG [1], called variance reduced stochastic gradient descent (VR-SGD). Unlike the choices of snapshot and starting points in SVRG and Prox-SVRG [33], the two points of each epoch in VR-SGD are set to the average and last iterate of the previous epoch, respectively. This setting allows us to use much larger learning rates than SVRG, e.g., $1/L$ for VR-SGD vs. $1/(10L)$ for SVRG, and also makes VR-SGD much more robust to learning rate selection. Different from existing proximal stochastic methods such as Prox-SVRG and Katyusha [47], we designed two different update rules for smooth and non-smooth problems, respectively, which makes VR-SGD suitable for non-smooth and/or non-strongly convex problems without using any reduction techniques as in [63]. Our empirical results also showed that for smooth problems stochastic gradient update rules as in (12) are better choices than proximal update formulas as in (10).

On the practical side, the choices of the snapshot and starting points make VR-SGD significantly faster than its counterparts, SVRG and Prox-SVRG. On the theoretical side, the setting also makes our convergence analysis more challenging. We analyzed the convergence properties of VR-SGD for strongly convex objective functions, which show that VR-SGD attains a linear convergence rate. Moreover, we provided the convergence guarantees of VR-SGD for non-strongly convex functions, and our experimental results showed that VR-SGD achieves similar performance to its momentum accelerated variant that has the optimal convergence rate $\mathcal{O}(1/T^2)$. In contrast, SVRG and Prox-SVRG cannot directly solve non-strongly convex functions [43]. Various experimental results show that VR-SGD significantly outperforms state-of-the-art variance reduction methods such as SAGA [54], SVRG [1] and Prox-SVRG [33], and also achieves better or at least comparable performance with recently-proposed acceleration methods, e.g., Catalyst [39] and Katyusha [47]. Since VR-SGD has a much lower per-iteration complexity than accelerated methods (e.g., Katyusha), it has more obvious advantage over them in terms of running time, especially for high-dimensional sparse data. This further verifies that VR-SGD is suitable for various large-scale machine learning. Furthermore, as the update rules of VR-SGD are much simpler than existing accelerated stochastic variance reduction methods such as Katyusha, it is more friendly to asynchronous parallel and distributed implementation similar to [55], [57], [75].

Acknowledgments

We thank the reviewers for their valuable comments. This work was supported in part by Project supported the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (No. 61621005), the Major Research Plan of the National Natural Science Foundation of China (Nos. 91438201 and 91438103), Project supported the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (No. 61621005), the National Natural Science Foundation of China (Nos. 61876220, 61876221, 61836009, U1701267, 61871310, 61573267, 61502369 and 61473215), the Program for Cheung Kong Scholars and Innovative Research Team in University (No. IRT_15R53), the Fund for Foreign Scholars in University Research and Teaching Programs (the 111 Project) (No. B07048), the Science Foundation of Xidian University (No. 10251180018), and Grants (CUHK 14206715 & 14222816) from the Hong Kong RGC.

References

[1] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 315–323.
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[3] S. Zhang, A. Choromanska, and Y. LeCun, "Deep learning with elastic averaging SGD," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 685–693.
[4] Z. Allen-Zhu and E. Hazan, "Variance reduction for faster non-convex optimization," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 699–707.
[5] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola, "Stochastic variance reduction for nonconvex optimization," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 314–323.
[6] Y. Liu, F. Shang, and J. Cheng, "Accelerated variance reduced stochastic ADMM," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 2287–2293.
[7] X. Li, T. Zhao, R. Arora, H. Liu, and J. Haupt, "Nonconvex sparse learning via stochastic optimization with progressive variance reduction," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 917–925.
[8] W. Zhang, L. Zhang, Z. Jin, R. Jin, D. Cai, X. Li, R. Liang, and X. He, "Sparse learning with stochastic composite optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1223–1236, Jun. 2017.
[9] C. Qu, Y. Li, and H. Xu, "Linear convergence of SVRG in statistical estimation," *arXiv:1611.01957v3*, 2017.
[10] C. Paquette, H. Lin, D. Drusvyatskiy, J. Mairal, and Z. Harchaoui, "Catalyst for gradient-based nonconvex optimization," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2018, pp. 613–622.
[11] J. Duchi and F. Ruan, "Stochastic methods for composite optimization problems," *arXiv:1703.08570*, 2017.
[12] B. Recht and C. Ré, "Parallel stochastic gradient algorithms for large-scale matrix completion," *Math. Prog. Comp.*, vol. 5, pp. 201–226, 2013.
[13] L. Wang, X. Zhang, and Q. Gu, "A unified variance reduction-based framework for nonconvex low-rank matrix recovery," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3712–3721.
[14] M. Schmidt, R. Babanezhad, M. Ahmed, A. Defazio, A. Clifton, and A. Sarkar, "Non-uniform stochastic average gradient method for training conditional random fields," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2015, pp. 819–828.

[15] Z. Allen-Zhu and Y. Li, "Doubly accelerated methods for faster CCA and generalized eigendecomposition," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 98–106.

[16] O. Shamir, "A stochastic PCA and SVD algorithm with an exponential convergence rate," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 144–152.

[17] D. Garber, E. Hazan, C. Jin, S. M. Kakade, C. Musco, P. Netrapalli, and A. Sidford, "Faster eigenvector computation via shift-and-invert preconditioning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2626–2634.

[18] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," *Soviet Math. Doklady*, vol. 27, pp. 372–376, 1983.

[19] ——, *Introductory Lectures on Convex Optimization: A Basic Course*. Boston: Kluwer Academic Publ., 2004.

[20] P. Tseng, "On accelerated proximal gradient methods for convex-concave optimization," *Technical report, University of Washington*, 2008.

[21] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[22] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, 1951.

[23] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2004, pp. 919–926.

[24] C. Hu, J. T. Kwok, and W. Pan, "Accelerated gradient methods for stochastic optimization and online learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 781–789.

[25] S. Bubeck, "Convex optimization: Algorithms and complexity," *Found. Trends Mach. Learn.*, vol. 8, pp. 231–358, 2015.

[26] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, pp. 354–359, 2017.

[27] A. Rakhlin, O. Shamir, and K. Sridharan, "Making gradient descent optimal for strongly convex stochastic optimization," in *Proc. Int. Conf. Mach. Learn.*, 2012, pp. 449–456.

[28] N. L. Roux, M. Schmidt, and F. Bach, "A stochastic gradient method with an exponential convergence rate for finite training sets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 2672–2680.

[29] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization," *J. Mach. Learn. Res.*, vol. 14, pp. 567–599, 2013.

[30] A. Defazio, F. Bach, and S. Lacoste-Julien, "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1646–1654.

[31] Y. Zhang and L. Xiao, "Stochastic primal-dual coordinate method for regularized empirical risk minimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 353–361.

[32] M. Schmidt, N. L. Roux, and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Math. Program.*, vol. 162, pp. 83–112, 2017.

[33] L. Xiao and T. Zhang, "A proximal stochastic gradient method with progressive variance reduction," *SIAM J. Optim.*, vol. 24, no. 4, pp. 2057–2075, 2014.

[34] S. Shalev-Shwartz and T. Zhang, "Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization," *Math. Program.*, vol. 155, pp. 105–145, 2016.

[35] F. Shang, Y. Liu, J. Cheng, and J. Zhuo, "Fast stochastic variance reduced gradient method with momentum acceleration for machine learning," *arXiv:1703.07948*, 2017.

[36] L. Zhang, M. Mahdavi, and R. Jin, "Linear convergence with condition number independent access of full gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 980–988.

[37] J. Konečný, J. Liu, P. Richtárik, , and M. Takáč, "Mini-batch semi-stochastic gradient descent in the proximal setting," *IEEE J. Sel. Top. Sign. Proces.*, vol. 10, no. 2, pp. 242–255, 2016.

[38] A. Nitanda, "Stochastic proximal gradient descent with acceleration techniques," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1574–1582.

[39] H. Lin, J. Mairal, and Z. Harchaoui, "A universal catalyst for first-order optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3366–3374.

[40] R. Frostig, R. Ge, S. M. Kakade, and A. Sidford, "Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2540–2548.

[41] G. Lan and Y. Zhou, "An optimal randomized incremental gradient method," *Math. Program.*, 2017.

[42] R. Babanezhad, M. O. Ahmed, A. Virani, M. Schmidt, J. Konecny, and S. Sallinen, "Stop wasting my gradients: Practical SVRG," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2242–2250.

[43] Z. Allen-Zhu and Y. Yuan, "Improved SVRG for non-strongly-convex or sum-of-non-convex objectives," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1080–1089.

[44] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Res. Logist. Quart.*, vol. 3, pp. 95–110, 1956.

[45] E. Hazan and H. Luo, "Variance-reduced and projection-free stochastic optimization," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1263–1271.

[46] F. Shang, Y. Liu, J. Cheng, K. W. Ng, and Y. Yoshida, "Guaranteed sufficient decrease for stochastic variance reduced gradient optimization," in *Proc. 21st Int. Conf. Artif. Intell. Statist.*, 2018, pp. 1027–1036.

[47] Z. Allen-Zhu, "Katyusha: The first direct acceleration of stochastic gradient methods," *J. Mach. Learn. Res.*, vol. 18, no. 221, pp. 1–51, 2018.

[48] L. Hien, C. Lu, H. Xu, and J. Feng, "Accelerated stochastic mirror descent algorithms for composite non-strongly convex optimization," *arXiv:1605.06892v4*, 2017.

[49] K. Zhou, F. Shang, and J. Cheng, "A simple stochastic variance reduced algorithm with fast convergence rates," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5975–5984.

[50] B. Woodworth and N. Srebro, "Tight complexity bounds for optimizing composite objectives," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3639–3647.

[51] A. Defazio, "A simple practical accelerated method for finite sums," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 676–684.

[52] L. Liu, J. Liu, and D. Tao, "Variance reduced methods for non-convex composition optimization," *arXiv:1711.04416v1*, 2017.

[53] L. Lei, C. Ju, J. Chen, and M. I. Jordan, "Nonconvex finite-sum optimization via SCSG methods," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2345–2355.

[54] S. J. Reddi, S. Sra, B. Póczos, and A. Smola, "Fast incremental method for smooth nonconvex optimization," in *Proc. Conf. Decision and Contorl*, 2016, pp. 1971–1977.

[55] S. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola, "On variance reduction in stochastic gradient descent and its asynchronous variants," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2629–2637.

[56] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017.

[57] J. D. Lee, Q. Lin, T. Ma, and T. Yang, "Distributed stochastic variance reduced gradient methods by sampling extra data with replacement," *J. Mach. Learn. Res.*, vol. 18, pp. 1–43, 2017.

[58] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[59] R. Ge, F. Huang, C. Jin, and Y. Yuan, "Escaping from saddle points – online stochastic gradient for tensor decomposition," in *Proc. Conf. Learn. Theory*, 2015, pp. 797–842.

[60] A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens, "Adding gradient noise improves learning for very deep networks," *arXiv:1511.06807*, 2015.

[61] N. Flammarion and F. Bach, "From averaging to acceleration, there is only a step-size," in *Proc. Conf. Learn. Theory*, 2015, pp. 658–695.

[62] C. Tan, S. Ma, Y. Dai, and Y. Qian, "Barzilai-Borwein step size for stochastic gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 685–693.

[63] Z. Allen-Zhu and E. Hazan, "Optimal black-box reductions between optimization objectives," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1606–1614.

[64] Y. Nesterov, "Smooth minimization of non-smooth functions," *Math. Program.*, vol. 103, pp. 127–152, 2005.

[65] Y. Xu, Y. Yan, Q. Lin, and T. Yang, "Homotopy smoothing for non-smooth problems with lower complexity than $O(1/\epsilon)$," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1208–1216.

[66] P. Zhao and T. Zhang, "Stochastic optimization with importance sampling for regularized loss minimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1–9.
[67] D. Needell, N. Srebro, and R. Ward, "Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm," *Math. Program.*, vol. 155, pp. 549–573, 2016.
[68] O. Shamir, "Without-replacement sampling for stochastic gradient methods," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 46–54.
[69] J. Konečný and P. Richtárik, "Semi-stochastic gradient descent methods," *Optim. Method Softw.*, vol. 32, no. 5, pp. 993–1005, 2017.
[70] F. Shang, Y. Liu, L. Jiao, K. Zhou, J. Cheng, Y. Ren, and Y. Jin, "ASVRG: Accelerated proximal SVRG," in *Proc. Mach. Learn. Res.*, 2018, pp. 1–16.
[71] B. Carpenter, "Lazy sparse stochastic gradient descent for regularized multinomial logistic regression," *Tech. Rep.*, 2008.
[72] J. Langford, L. Li, and T. Zhang, "Sparse online learning via truncated gradient," *J. Mach. Learn. Res.*, vol. 10, pp. 777–801, 2009.
[73] W. Su, S. Boyd, and E. J. Candes, "A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights," *J. Mach. Learn. Res.*, vol. 17, pp. 1–43, 2016.
[74] S. Shalev-Shwartz, O. Shamir, and K. Sridharan, "Learning kernel-based halfspaces with the 0-1 loss," *SIAM J. Comput.*, vol. 40, no. 6, pp. 1623–1646, 2011.
[75] H. Mania, X. Pan, D. Papailiopoulos, B. Recht, K. Ramchandran, and M. I. Jordan, "Perturbed iterate analysis for asynchronous stochastic optimization," *SIAM J. Optim.*, vol. 27, no. 4, pp. 2202–2229, 2017.

Fanhua Shang (M'14) received the Ph.D. degree in Circuits and Systems from Xidian University, Xi'an, China, in 2012.

He is currently a professor with the School of Artificial Intelligence, Xidian University, China. Prior to joining Xidian University, he was a Research Associate with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. From 2013 to 2015, he was a Post-Doctoral Research Fellow with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. From 2012 to 2013, he was a Post-Doctoral Research Associate with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA. His current research interests include machine learning, data mining, pattern recognition, and computer vision.

Zhou Kaiwen received the BS degree in Computer Science and Technology from Fudan University in 2017. He is currently working toward his Master degree in the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. His current research interests include data mining, stochastic optimization for machine learning, large scale machine learning, etc.

Hongying Liu (M'10) received her B.E. and M.S. degrees in Computer Science and Technology from XiAn University of Technology, China, in 2006 and 2009, respectively, and Ph.D. in Engineering from Waseda University, Japan in 2012. Currently, she is a faculty member at the School of Artificial Intelligence, and also with the Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, Xidian University, China. In addition, she is a member of IEEE. Her major research interests include image processing, intelligent signal processing, machine learning, etc.

James Cheng is an Assistant Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. His current research interests include distributed computing systems, large-scale network analysis, temporal networks, and big data.

Ivor W. Tsang is an ARC future fellow and a professor of artificial intelligence with the University of Technology Sydney. He is also the research director of the UTS Priority Research Centre for Artificial Intelligence. His research focuses on transfer learning, feature selection, big data analytics for data with trillions of dimensions, and their applications to computer vision and pattern recognition. He has more than 140 research papers published in top-tier journal and conference papers, including the Journal of Machine Learning Research, the Madras Law Journal, the IEEE Transactions on Pattern Analysis and Machine Intelligence, the IEEE Transactions on Neural Networks and Learning Systems, NIPS, ICML, etc. In 2009, he was conferred the 2008 Natural Science Award (Class II) by Ministry of Education, China, which recognized his contributions to kernel methods. In 2013, he received his prestigious Australian Research Council Future Fellowship for his research regarding Machine Learning on Big Data. In addition, he had received the prestigious IEEE Transactions on Neural Networks Outstanding 2004 Paper Award in 2007, the 2014 IEEE Transactions on Multimedia Prize Paper Award, and a number of best paper awards and honors from reputable international conferences, including the Best Student Paper Award at CVPR 2010, and the Best Paper Award at ICTAI 2011. He was also awarded the ECCV 2012 Outstanding Reviewer Award.

Lijun Zhang received the BS and PhD degrees in software engineering and computer science from Zhejiang University, China, in 2007 and 2012, respectively. He is currently an associate professor of the Department of Computer Science and Technology, Nanjing University, China. Prior to joining Nanjing University, he was a postdoctoral researcher at the Department of Computer Science and Engineering, Michigan State University. His research interests include machine learning, optimization, information retrieval, and data mining. He is a member of the IEEE.

Dacheng Tao (F'15) is a professor of computer science and ARC future fellow in the School of Information Technologies and the Faculty of Engineering and Information Technologies, and the inaugural director of the UBTECH Sydney Artificial Intelligence Centre, The University of Sydney. He mainly applies statistics and mathematics to artificial intelligence and data science. His research interests spread across computer vision, data science, image processing, machine learning, and video surveillance. His research results have expounded in one monograph and more than 500 publications at prestigious journals and prominent conferences, such as the IEEE Transactions on Pattern Analysis and Machine Intelligence, the IEEE Transactions on Neural Networks and Learning Systems, the IEEE Transactions on Image Processing, the Journal of Machine Learning Research, the International Journal of Computer Vision, NIPS, CIKM, ICML, CVPR, ICCV, ECCV, AISTATS, ICDM, and ACM SIGKDD, with several Best Paper Awards, such as the Best Theory/Algorithm Paper Runner up Award in IEEE ICDM07, the Best Student Paper Award in IEEE ICDM13, the 2014 ICDM 10-year Highest-Impact Paper Award, and the 2017 IEEE Signal Processing Society Best Paper Award. He received the 2015 Australian Scopus-Eureka Prize, the 2015 ACS Gold Disruptor Award, and the 2015 UTS Vice-Chancellors Medal for Exceptional Research. He is a fellow of the IEEE, the OSA, the IAPR, and the SPIE.

Licheng Jiao (F'18) received the B.S. degree from Shanghai Jiaotong University, Shanghai, China, in 1982, and the M.S. and Ph.D. degrees from Xian Jiaotong University, Xian, China, in 1984 and 1990, respectively.

He was a Post-Doctoral Fellow with the National Key Laboratory for Radar Signal Processing, Xidian University, Xian, from 1990 to 1991, where he has been a Professor with the School of Electronic Engineering, since 1992, and currently the Director of the Key Laboratory of Intelligent Perception and Image Understanding, Ministry of Education of China. He has charged of about 40 important scientific research projects, and published over 20 monographs and a hundred papers in international journals and conferences. His current research interests include image processing, natural computation, machine learning, and intelligent information processing.

Dr. Jiao is the Chairman of Awards and Recognition Committee, the Vice Board Chairperson of the Chinese Association of Artificial Intelligence, a Councilor of the Chinese Institute of Electronics, a Committee Member of the Chinese Committee of Neural Networks, and an expert of the Academic Degrees Committee of the State Council. He is a fellow of the IEEE.