Stop Wasting My Gradients: Practical SVRG

Reza Babanezhad[1], Mohamed Osama Ahmed[1], Alim Virani[2], Mark Schmidt[1]
Department of Computer Science
University of British Columbia
[1]{rezababa, moahmed, schmidtm}@cs.ubc.ca,[2]alim.virani@gmail.com

Jakub Konečný
School of Mathematics
University of Edinburgh
kubo.konecny@gmail.com

Scott Sallinen
Department of Electrical and Computer Engineering
University of British Columbia
scotts@ece.ubc.ca

Abstract

We present and analyze several strategies for improving the performance of stochastic variance-reduced gradient (SVRG) methods. We first show that the convergence rate of these methods can be preserved under a decreasing sequence of errors in the control variate, and use this to derive variants of SVRG that use growing-batch strategies to reduce the number of gradient calculations required in the early iterations. We further (i) show how to exploit support vectors to reduce the number of gradient computations in the later iterations, (ii) prove that the commonly–used regularized SVRG iteration is justified and improves the convergence rate, (iii) consider alternate mini-batch selection strategies, and (iv) consider the generalization error of the method.

1 Introduction

We consider the problem of optimizing the average of a finite but large sum of smooth functions,

$$\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x). \tag{1}$$

A huge proportion of the model-fitting procedures in machine learning can be mapped to this problem. This includes classic models like least squares and logistic regression but also includes more advanced methods like conditional random fields and deep neural network models. In the high-dimensional setting (large d), the traditional approaches for solving (1) are: *full gradient* (FG) methods which have linear convergence rates but need to evaluate the gradient f_i for all n examples on every iteration, and *stochastic gradient* (SG) methods which make rapid initial progress as they only use a single gradient on each iteration but ultimately have slower sublinear convergence rates.

Le Roux et al. [1] proposed the first general method, *stochastic average gradient* (SAG), that only considers one training example on each iteration but still achieves a linear convergence rate. Other methods have subsequently been shown to have this property [2, 3, 4], but these all require storing a previous evaluation of the gradient f_i' or the dual variables for each i. For many objectives this only requires $O(n)$ space, but for general problems this requires $O(np)$ space making them impractical.

Recently, several methods have been proposed with similar convergence rates to SAG but without the memory requirements [5, 6, 7, 8]. They are known as *mixed gradient*, *stochastic variance-reduced gradient* (SVRG), and *semi-stochastic gradient* methods (we will use SVRG). We give a canonical SVRG algorithm in the next section, but the salient features of these methods are that they evaluate two gradients on each iteration and occasionally must compute the gradient on all examples. SVRG

methods often dramatically outperform classic FG and SG methods, but these extra evaluations mean that SVRG is slower than SG methods in the important early iterations. They also mean that SVRG methods are typically slower than memory-based methods like SAG.

In this work we first show that SVRG is robust to inexact calculation of the full gradients it requires (§3), provided the accuracy increases over time. We use this to explore growing-batch strategies that require fewer gradient evaluations when far from the solution, and we propose a mixed SG/SVRG method that may improve performance in the early iterations (§4). We next explore using support vectors to reduce the number of gradients required when close to the solution (§5), give a justification for the regularized SVRG update that is commonly used in practice (§6), consider alternative mini-batch strategies (§7), and finally consider the generalization error of the method (§8).

2 Notation and SVRG Algorithm

SVRG assumes f is μ-strongly convex, each f_i is convex, and each gradient f_i' is Lipschitz-continuous with constant L. The method begins with an initial estimate x^0, sets $x_0 = x^0$ and then generates a sequence of iterates x_t using

$$x_t = x_{t-1} - \eta(f_{i_t}'(x_{t-1}) - f_{i_t}'(x^s) + \mu^s), \tag{2}$$

where η is the positive step size, we set $\mu^s = f'(x^s)$, and i_t is chosen uniformly from $\{1, 2, \dots, n\}$. After every m steps, we set $x^{s+1} = x_t$ for a random $t \in \{1, \dots, m\}$, and we reset $t = 0$ with $x_0 = x^{s+1}$.

To analyze the convergence rate of SVRG, we will find it convenient to define the function

$$\rho(a, b) = \frac{1}{1 - 2\eta a}\left(\frac{1}{m\mu\eta} + 2b\eta\right).$$

as it appears repeatedly in our results. We will use $\rho(a)$ to indicate the value of $\rho(a, b)$ when $a = b$, and we will simply use ρ for the special case when $a = b = L$. Johnson & Zhang [6] show that if η and m are chosen such that $0 < \rho < 1$, the algorithm achieves a linear convergence rate of the form

$$\mathbb{E}[f(x^{s+1}) - f(x^*)] \leq \rho\mathbb{E}[f(x^s) - f(x^*)],$$

where x^* is the optimal solution. This convergence rate is very fast for appropriate η and m. While this result relies on constants we may not know in general, practical choices with good empirical performance include setting $m = n$, $\eta = 1/L$, and using $x^{s+1} = x_m$ rather than a random iterate.

Unfortunately, the SVRG algorithm requires $2m + n$ gradient evaluations for every m iterations of (2), since updating x_t requires two gradient evaluations and computing μ^s require n gradient evaluations. We can reduce this to $m + n$ if we store the gradients $f_i'(x^s)$, but this is not practical in most applications. Thus, SVRG requires many more gradient evaluations than classic SG iterations of memory-based methods like SAG.

3 SVRG with Error

We first give a result for the SVRG method where we assume that μ^s is equal to $f'(x^s)$ up to some error e^s. This is in the spirit of the analysis of [9], who analyze FG methods under similar assumptions. We assume that $\|x_t - x^*\| \leq Z$ for all t, which has been used in related work [10] and is reasonable because of the coercity implied by strong-convexity.

Proposition 1. *If $\mu^s = f'(x^s) + e^s$ and we set η and m so that $\rho < 1$, then the SVRG algorithm* (2) *with x^{s+1} chosen randomly from $\{x_1, x_2, \dots, x_m\}$ satisfies*

$$\mathbb{E}[f(x^{s+1}) - f(x^*)] \leq \rho\mathbb{E}[f(x^s) - f(x^*)] + \frac{Z\mathbb{E}\|e^s\| + \eta\mathbb{E}\|e^s\|^2}{1 - 2\eta L}.$$

We give the proof in Appendix A. This result implies that SVRG does not need a very accurate approximation of $f'(x^s)$ in the crucial early iterations since the first term in the bound will dominate. Further, this result implies that we can maintain the *exact* convergence rate of SVRG as long as the errors e^s decrease at an appropriate rate. For example, we obtain the same convergence rate provided that $\max\{\mathbb{E}\|e^s\|, \mathbb{E}\|e^s\|^2\} \leq \gamma\tilde{\rho}^s$ for any $\gamma \geq 0$ and some $\tilde{\rho} < \rho$. Further, we still obtain a linear convergence rate as long as $\|e^s\|$ converges to zero with a linear convergence rate.

Algorithm 1 Batching SVRG

Input: initial vector x^0, update frequency m, learning rate η.
for $s = 0, 1, 2, \ldots$ **do**
  Choose batch size $|\mathcal{B}^s|$
  $\mathcal{B}^s$ = $|\mathcal{B}^s|$ elements sampled without replacement from $\{1, 2, \ldots, n\}$.
  $\mu^s = \frac{1}{|\mathcal{B}^s|} \sum_{i \in \mathcal{B}^s} f_i^{'}(x^s)$
  x_0=x^s
  for $t = 1, 2, \ldots, m$ **do**
    Randomly pick $i_t \in 1, \ldots, n$
    $x_t = x_{t-1} - \eta(f_{i_t}^{'}(x_{t-1}) - f_{i_t}^{'}(x^s) + \mu^s)$ $(*)$
  end for
  option I: set $x^{s+1} = x_m$
  option II: set $x^{s+1} = x_t$ for random $t \in \{1, \ldots, m\}$
end for

3.1 Non-Uniform Sampling

Xiao & Zhang [11] show that non-uniform sampling (NUS) improves the performance of SVRG. They assume each f_i' is L_i-Lipschitz continuous, and sample $i_t = i$ with probability $L_i/n\bar{L}$ where $\bar{L} = \frac{1}{n}\sum_{i=1}^n L_i$. The iteration is then changed to

$$x_t = x_{t-1} - \eta\left(\frac{\bar{L}}{L_{i_t}}[f_{i_t}^{'}(x_{t-1}) - f_{i_t}^{'}(\tilde{x})] + \mu^s\right),$$

which maintains that the search direction is unbiased. In Appendix A, we show that if μ^s is computed with error for this algorithm and if we set η and m so that $0 < \rho(\bar{L}) < 1$, then we have a convergence rate of

$$\mathbb{E}[f(x^{s+1}) - f(x^*)] \leq \rho(\bar{L})\mathbb{E}[f(x^s) - f(x^*)] + \frac{Z\mathbb{E}\|e^s\| + \eta\mathbb{E}\|e^s\|^2}{1 - 2\eta\bar{L}},$$

which can be faster since the average $\bar{L}$ may be much smaller than the maximum value L.

3.2 SVRG with Batching

There are many ways we could allow an error in the calculation of μ^s to speed up the algorithm. For example, if evaluating each f_i' involves solving an optimization problem, then we could solve this optimization problem inexactly. For example, if we are fitting a graphical model with an iterative approximate inference method, we can terminate the iterations early to save time.

When the f_i are simple but n is large, a natural way to approximate μ^s is with a subset (or 'batch') of training examples $\mathcal{B}^s$ (chosen *without* replacement),

$$\mu^s = \frac{1}{|\mathcal{B}^s|}\sum_{i \in \mathcal{B}^s} f_i'(x^s).$$

The batch size $|\mathcal{B}^s|$ controls the error in the approximation, and we can drive the error to zero by increasing it to n. Existing SVRG methods correspond to the special case where $|\mathcal{B}^s| = n$ for all s.

Algorithm 1 gives pseudo-code for an SVRG implementation that uses this sub-sampling strategy. If we assume that the sample variance of the norms of the gradients is bounded by S^2 for all x^s,

$$\frac{1}{n-1}\sum_{i=1}^n \left[\|f_i'(x^s)\|^2 - \|f'(x^s)\|^2\right] \leq S^2,$$

then we have that [12, Chapter 2]

$$\mathbb{E}\|e^s\|^2 \leq \frac{n - |\mathcal{B}^s|}{n|\mathcal{B}^s|}S^2.$$

So if we want $\mathbb{E}\|e^s\|^2 \leq \gamma\tilde{\rho}^{2s}$, where $\gamma \geq 0$ is a constant for some $\tilde{\rho} < 1$, we need

$$|\mathcal{B}^s| \geq \frac{nS^2}{S^2 + n\gamma\tilde{\rho}^{2s}}. \tag{3}$$

Algorithm 2 Mixed SVRG and SG Method

Replace (*) in Algorithm 1 with the following lines:
if $f_{i_t} \in \mathcal{B}^s$ **then**
$x_t = x_{t-1} - \eta(f'_{i_t}(x_{t-1}) - f'_{i_t}(x^s) + \mu^s)$
else
$x_t = x_{t-1} - \eta f'_{i_t}(x_{t-1})$
end if

If the batch size satisfies the above condition then

$$Z\mathbb{E}\|e^{s-1}\| + \eta\mathbb{E}\|e^{s-1}\|^2 \leq Z\sqrt{\gamma}\tilde{\rho}^s + \eta\gamma\tilde{\rho}^{2s} \leq 2\max\{Z\sqrt{\gamma}, \eta\gamma\tilde{\rho}\}\tilde{\rho}^s,$$

and *the convergence rate of SVRG is unchanged compared to using the full batch* on all iterations.

The condition (3) guarantees a linear convergence rate under any exponentially-increasing sequence of batch sizes, the strategy suggested by [13] for classic SG methods. However, a tedious calculation shows that (3) has an inflection point at $s = \log(S^2/\gamma n)/2\log(1/\tilde{\rho})$, corresponding to $|\mathcal{B}^s| = \frac{n}{2}$. This was previously observed empirically [14, Figure 3], and occurs because we are sampling without replacement. This transition means we don't need to increase the batch size exponentially.

4 Mixed SG and SVRG Method

An approximate μ^s can drastically reduce the computational cost of the SVRG algorithm, but does not affect the 2 in the $2m+n$ gradients required for m SVRG iterations. This factor of 2 is significant in the early iterations, since this is when stochastic methods make the most progress and when we typically see the largest reduction in the *test* error.

To reduce this factor, we can consider a *mixed* strategy: if i_t is in the batch $\mathcal{B}^s$ then perform an SVRG iteration, but if i_t is not in the current batch then use a classic SG iteration. We illustrate this modification in Algorithm 2. This modification allows the algorithm to take advantage of the rapid initial progress of SG, since it predominantly uses SG iterations when far from the solution. Below we give a convergence rate for this mixed strategy.

Proposition 2. *Let $\mu^s = f'(x^s) + e^s$ and we set η and m so that $0 < \rho(L, \alpha L) < 1$ with $\alpha = |\mathcal{B}^s|/n$. If we assume $\mathbb{E}\|f'_i(x)\|^2 \leq \sigma^2$ then Algorithm 2 has*

$$\mathbb{E}[f(x^{s+1}) - f(x^*)] \leq \rho(L, \alpha L)\mathbb{E}[f(x^s) - f(x^*)] + \frac{Z\mathbb{E}\|e^s\| + \eta\mathbb{E}\|e^s\|^2 + \frac{\eta\sigma^2}{2}(1-\alpha)}{1 - 2\eta L}$$

We give the proof in Appendix B. The extra term depending on the variance σ^2 is typically the bottleneck for SG methods. Classic SG methods require the step-size η to converge to zero because of this term. However, the mixed SG/SVRG method can keep the fast progress from using a constant η since the term depending on σ^2 converges to zero as α converges to one. Since $\alpha < 1$ implies that $\rho(L, \alpha L) < \rho$, this result implies that when $[f(x^s) - f(x^*)]$ is large compared to e^s and σ^2 that the mixed SG/SVRG method actually converges faster.

Sharing a single step size η between the SG and SVRG iterations in Proposition 2 is sub-optimal. For example, if x is close to x^* and $|\mathcal{B}^s| \approx n$, then the SG iteration might actually take us far away from the minimizer. Thus, we may want to use a decreasing sequence of step sizes for the SG iterations. In Appendix B, we show that using $\eta = O^*(\sqrt{(n - |\mathcal{B}|)/n|\mathcal{B}|})$ for the SG iterations can improve the dependence on the error e^s and variance σ^2.

5 Using Support Vectors

Using a batch $\mathcal{B}^s$ decreases the number of gradient evaluations required when SVRG is far from the solution, but its benefit diminishes over time. However, for certain objectives we can further

Algorithm 3 Heuristic for skipping evaluations of f_i at x

if $sk_i = 0$ **then**
  compute $f_i'(x)$.
  if $f_i'(x) = 0$ **then**
    $ps_i = ps_i + 1$. {Update the number of consecutive times $f_i'(x)$ was zero.}
    $sk_i = 2^{\max\{0, ps_i - 2\}}$. {Skip exponential number of future evaluations if it remains zero.}
  else
    $ps_i = 0$. {This could be a support vector, do not skip it next time.}
  end if
  return $f_i'(x)$.
else
  $sk_i = sk_i - 1$. {In this case, we skip the evaluation.}
  return 0.
end if

reduce the number of gradient evaluations by identifying *support vectors*. For example, consider minimizing the Huberized hinge loss (HSVM) with threshold ϵ [15],

$$\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f(b_i a_i^T x), \quad f(\tau) = \begin{cases} 0 & \text{if } \tau > 1 + \epsilon, \\ 1 - \tau & \text{if } \tau < 1 - \epsilon, \\ \frac{(1+\epsilon-\tau)^2}{4\epsilon} & \text{if } |1 - \tau| \leq \epsilon, \end{cases}$$

In terms of (1), we have $f_i(x) = f(b_i a_i^T x)$. The performance of this loss function is similar to logistic regression and the hinge loss, but it has the appealing properties of both: it is *differentiable* like logistic regression meaning we can apply methods like SVRG, but it has *support vectors* like the hinge loss meaning that many examples will have $f_i(x^*) = 0$ and $f_i'(x^*) = 0$. We can also construct Huberized variants of many non-smooth losses for regression and multi-class classification.

If we knew the support vectors where $f_i(x^*) > 0$, we could solve the problem faster by ignoring the non-support vectors. For example, if there are 100000 training examples but only 100 support vectors in the optimal solution, we could solve the problem 1000 times faster. While we typically don't know the support vectors, in this section we outline a heuristic that gives large practical improvements by trying to identify them as the algorithm runs.

Our heuristic has two components. The first component is maintaining the *list of non-support vectors at* x^s. Specifically, we maintain a list of examples i where $f_i'(x^s) = 0$. When SVRG picks an example i_t that is part of this list, we know that $f_{i_t}'(x^s) = 0$ and thus the iteration only needs one gradient evaluation. This modification is not a heuristic, in that it still applies the exact SVRG algorithm. However, at best it can only cut the runtime in half.

The heuristic part of our strategy is to skip $f_i'(x^s)$ or $f_i'(x_t)$ if our evaluation of f_i' has been zero more than two consecutive times (and skipping it an exponentially larger number of times each time it remains zero). Specifically, for each example i we maintain two variables, sk_i (for 'skip') and ps_i (for 'pass'). Whenever we need to evaluate f_i' for some x^s or x_t, we run Algorithm 3 which may skip the evaluation. This strategy can lead to huge computational savings in later iterations if there are few support vectors, since many iterations will require no gradient evaluations.

Identifying support vectors to speed up computation has long been an important part of SVM solvers, and is related to the classic shrinking heuristic [16]. While it has previously been explored in the context of dual coordinate ascent methods [17], this is the first work exploring it for linearly-convergent stochastic gradient methods.

6 Regularized SVRG

We are often interested in the special case where problem (1) has the decomposition

$$\min_{x \in \mathbb{R}^d} \; f(x) \equiv h(x) + \frac{1}{n} \sum_{i=1}^{n} g_i(x). \tag{4}$$

A common choice of h is a scaled 1-norm of the parameter vector, $h(x) = \lambda\|x\|_1$. This non-smooth regularizer encourages sparsity in the parameter vector, and can be addressed with the proximal-SVRG method of Xiao & Zhang [11]. Alternately, if we want an explicit Z we could set h to the indicator function for a 2-norm ball containing x^*. In Appendix C, we give a variant of Proposition 1 that allows errors in the proximal-SVRG method for non-smooth/constrained settings like this.

Another common choice is the ℓ_2-regularizer, $h(x) = \frac{\lambda}{2}\|x\|^2$. With this regularizer, the SVRG updates can be equivalently written in the form

$$x_{t+1} = x_t - \eta\left(h'(x_t) + g'_{i_t}(x_t) - g'_{i_t}(x^s) + \mu^s\right), \tag{5}$$

where $\mu^s = \frac{1}{n}\sum_{i=1}^n g_i(x^s)$. That is, they take an exact gradient step with respect to the regularizer and an SVRG step with respect to the g_i functions. When the g'_i are sparse, this form of the update allows us to implement the iteration without needing full-vector operations. A related update is used by Le Roux et al. to avoid full-vector operations in the SAG algorithm [1, §4]. In Appendix C, we prove the below convergence rate for this update.

Proposition 3. *Consider instances of problem* (1) *that can be written in the form* (4) *where h' is L_h-Lipschitz continuous and each g'_i is L_g-Lipschitz continuous, and assume that we set η and m so that $0 < \rho(L_m) < 1$ with $L_m = \max\{L_g, L_h\}$. Then the regularized SVRG iteration* (5) *has*

$$\mathbb{E}[f(x^{s+1}) - f(x^*)] \leq \rho(L_m)\mathbb{E}[f(x^s) - f(x^*)],$$

Since $L_m \leq L$, and strictly so in the case of ℓ_2-regularization, this result shows that for ℓ_2-regularized problems SVRG actually converges faster than the standard analysis would indicate (a similar result appears in Konečný et al. [18]). Further, this result gives a theoretical justification for using the update (5) for other h functions where it is not equivalent to the original SVRG method.

7 Mini-Batching Strategies

Konečný et al. [18] have also recently considered using batches of data within SVRG. They consider using 'mini-batches' in the inner iteration (the update of x_t) to decrease the variance of the method, but still use full passes through the data to compute μ^s. This prior work is thus complimentary to the current work (in practice, both strategies can be used to improve performance). In Appendix D we show that sampling the inner mini-batch proportional to L_i achieves a convergence rate of

$$\mathbb{E}\left[f(x^{s+1}) - f(x^*)\right] \leq \rho_M \mathbb{E}\left[f(x^s) - f(x^*)\right],$$

where M is the size of the mini-batch while

$$\rho_M = \frac{1}{M - 2\eta\bar{L}}\left(\frac{M}{m\mu\eta} + 2\bar{L}\eta\right),$$

and we assume $0 < \rho_M < 1$. This generalizes the standard rate of SVRG and improves on the result of Konečný et al. [18] in the smooth case. This rate can be faster than the rate of the standard SVRG method at the cost of a more expensive iteration, and may be clearly advantageous in settings where parallel computation allows us to compute several gradients simultaneously.

The regularized SVRG form (5) suggests an alternate mini-batch strategy for problem (1): consider a mini-batch that contains a 'fixed' set $\mathcal{B}_f$ and a 'random' set $\mathcal{B}_t$. Without loss of generality, assume that we sort the f_i based on their L_i values so that $L_1 \geq L_2 \geq \cdots \geq L_n$. For the fixed $\mathcal{B}_f$ we will *always* choose the M_f values with the largest L_i, $\mathcal{B}_f = \{f_1, f_2, \ldots, f_{M_f}\}$. In contrast, we choose the members of the random set $\mathcal{B}_t$ by sampling from $B_r = \{f_{M_f+1}, \ldots, f_n\}$ proportional to their Lipschitz constants, $p_i = \frac{L_i}{(M_r)\bar{L}_r}$ with $\bar{L}_r = (1/M_r)\sum_{i=M_f+1}^n L_i$. In Appendix D, we show the following convergence rate for this strategy:

Proposition 4. *Let $g(x) = (1/n)\sum_{i\notin[\mathcal{B}_f]} f_i(x)$ and $h(x) = (1/n)\sum_{i\in[\mathcal{B}_f]} f_i(x)$. If we replace the SVRG update with*

$$x_{t+1} = x_t - \eta\left(h'(x_t) + (1/M_r)\sum_{i\in\mathcal{B}_t}\frac{\bar{L}_r}{L_i}(f'_i(x_t) - f'_i(x^s)) + g'(x^s)\right),$$

then the convergence rate is

$$\mathbb{E}[f(x^{s+1}) - f(x^*)] \leq \rho(\kappa, \zeta)\mathbb{E}[F(x^s) - f(x^*)].$$

where $\zeta = \frac{(n-M_f)\bar{L}_r}{(M-M_f)n}$ and $\kappa = \max\{\frac{L_1}{n}, \zeta\}$.

If $L_1 \leq n\bar{L}/M$ and $M_f < \frac{(\alpha-1)nM}{\alpha n - M}$ with $\alpha = \frac{\bar{L}}{\bar{L}_r}$, then we get a faster convergence rate than SVRG with a mini-batch of size M. The scenario where this rate is slower than the existing mini-batch SVRG strategy is when $L_1 \leq n\bar{L}/M$. But we could relax this assumption by dividing each element of the fixed set $\mathcal{B}_f$ into two functions: βf_i and $(1-\beta)f_i$, where $\beta = 1/M$, then replacing each function f_i in $\mathcal{B}_f$ with βf_i and adding $(1-\beta)f_i$ to the random set B_r. This result may be relevant if we have access to a field-programmable gate array (FPGA) or graphical processing unit (GPU) that can compute the gradient for a fixed subset of the examples very efficiently. However, our experiments (Appendix F) indicate this strategy only gives marginal gains.

In Appendix F, we also consider constructing mini-batches by sampling proportional to $f_i(x^s)$ or $\|f_i'(x^s)\|$. These seemed to work as well as Lipschitz sampling on all but one of the datasets in our experiments, and this strategy is appealing because we have access to these values while we may not know the L_i values. However, these strategies diverged on one of the datasets.

8 Learning efficiency

In this section we compare the performance of SVRG as a large-scale learning algorithm compared to FG and SG methods. Following Bottou & Bousquet [19], we can formulate the generalization error $\mathcal{E}$ of a learning algorithm as the sum of three terms

$$\mathcal{E} = \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}} + \mathcal{E}_{\text{opt}}$$

where the approximation error $\mathcal{E}_{\text{app}}$ measures the effect of using a limited class of models, the estimation error $\mathcal{E}_{\text{est}}$ measures the effect of using a finite training set, and the optimization error $\mathcal{E}_{\text{opt}}$ measures the effect of inexactly solving problem (1). Bottou & Bousquet [19] study asymptotic performance of various algorithms for a fixed approximation error and under certain conditions on the distribution of the data depending on parameters α or ν. In Appendix E, we discuss how SVRG can be analyzed in their framework. The table below includes SVRG among their results.

Algorithm	Time to reach $\mathcal{E}_{\text{opt}} \leq \epsilon$	Time to reach $\mathcal{E} = O(\mathcal{E}_{app} + \epsilon)$	Previous with $\kappa \sim n$
FG	$\mathcal{O}\left(n\kappa d \log\left(\frac{1}{\epsilon}\right)\right)$	$\mathcal{O}\left(\frac{d^2\kappa}{\epsilon^{1/\alpha}} \log^2\left(\frac{1}{\epsilon}\right)\right)$	$\mathcal{O}\left(\frac{d^3}{\epsilon^{2/\alpha}} \log^3\left(\frac{1}{\epsilon}\right)\right)$
SG	$\mathcal{O}\left(\frac{d\nu\kappa^2}{\epsilon}\right)$	$\mathcal{O}\left(\frac{d\nu\kappa^2}{\epsilon}\right)$	$\mathcal{O}\left(\frac{d^3\nu}{\epsilon} \log^2\left(\frac{1}{\epsilon}\right)\right)$
SVRG	$\mathcal{O}\left((n+\kappa)d \log\left(\frac{1}{\epsilon}\right)\right)$	$\mathcal{O}\left(\frac{d^2}{\epsilon^{1/\alpha}} \log^2\left(\frac{1}{\epsilon}\right) + \kappa d \log\left(\frac{1}{\epsilon}\right)\right)$	$\mathcal{O}\left(\frac{d^2}{\epsilon^{1/\alpha}} \log^2\left(\frac{1}{\epsilon}\right)\right)$

In this table, the condition number is $\kappa = L/\mu$. In this setting, linearly-convergent stochastic gradient methods can obtain better bounds for ill-conditioned problems, with a better dependence on the dimension and without depending on the noise variance ν.

9 Experimental Results

In this section, we present experimental results that evaluate our proposed variations on the SVRG method. We focus on logistic regression classification: given a set of training data $(a_1, b_1) \dots (a_n, b_n)$ where $a_i \in \mathbb{R}^d$ and $b_i \in \{-1, +1\}$, the goal is to find the $x \in \mathbb{R}^d$ solving

$$\underset{x \in \mathbb{R}^d}{\text{argmin}} \ \frac{\lambda}{2}\|x\|^2 + \frac{1}{n}\sum_{i=1}^{n} \log(1 + \exp(-b_i a_i^T x)),$$

We consider the datasets used by [1], whose properties are listed in the supplementary material. As in their work we add a bias variable, normalize dense features, and set the regularization parameter λ to $1/n$. We used a step-size of $\alpha = 1/L$ and we used $m = |\mathcal{B}^s|$ which gave good performance across methods and datasets. In our first experiment, we compared three variants of SVRG: the original strategy that uses all n examples to form μ^s (*Full*), a growing batch strategy that sets $|\mathcal{B}^s| = 2^s$ (*Grow*), and the mixed SG/SVRG described by Algorithm 2 under this same choice (*Mixed*). While a variety of practical batching methods have been proposed in the literature [13, 20, 21], we did not find that any of these strategies consistently outperformed the doubling used by the simple *Grow*

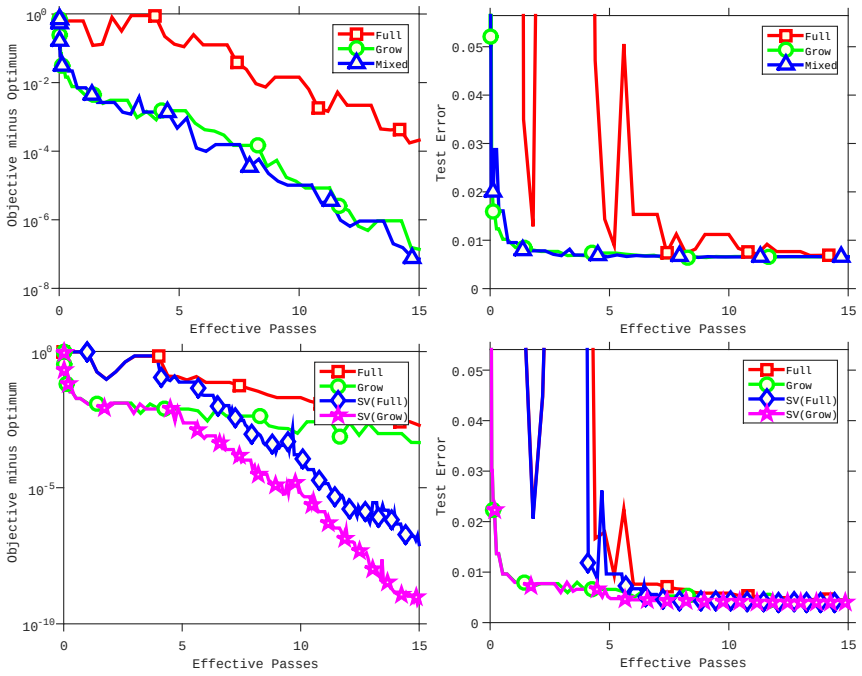

Figure 1: Comparison of training objective (left) and test error (right) on the *spam* dataset for the logistic regression (top) and the HSVM (bottom) losses under different batch strategies for choosing μ^s (*Full*, *Grow*, and *Mixed*) and whether we attempt to identify support vectors (*SV*).

strategy. Our second experiment focused on the ℓ_2-regularized HSVM on the same datasets, and we compared the original SVRG algorithm with variants that try to identify the support vectors (*SV*).

We plot the experimental results for one run of the algorithms on one dataset in Figure 1, while Appendix F reports results on the other 8 datasets over 10 different runs. In our results, the growing batch strategy (*Grow*) always had better test error performance than using the full batch, while for large datasets it also performed substantially better in terms of the training objective. In contrast, the *Mixed* strategy sometimes helped performance and sometimes hurt performance. Utilizing support vectors often improved the training objective, often by large margins, but its effect on the test objective was smaller.

10 Discussion

As SVRG is the only memory-free method among the new stochastic linearly-convergent methods, it represents the natural method to use for a huge variety of machine learning problems. In this work we show that the convergence rate of the SVRG algorithm can be preserved even under an inexact approximation to the full gradient. We also showed that using mini-batches to approximate μ^s gives a natural way to do this, explored the use of support vectors to further reduce the number of gradient evaluations, gave an analysis of the regularized SVRG update, and considered several new mini-batch strategies. Our theoretical and experimental results indicate that many of these simple modifications should be considered in any practical implementation of SVRG.

Acknowledgements

We would like to thank the reviewers for their helpful comments. This research was supported by the Natural Sciences and Engineering Research Council of Canada (RGPIN 312176-2010, RGPIN 311661-08, RGPIN-06068-2015). Jakub Konečný is supported by a Google European Doctoral Fellowship.

References

[1] N. Le Roux, M. Schmidt, and F. Bach, "A stochastic gradient method with an exponential convergence rate for strongly-convex optimization with finite training sets," *Advances in neural information processing systems (NIPS)*, 2012.

[2] S. Shalev-Schwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization," *Journal of Machine Learning Research*, vol. 14, pp. 567–599, 2013.

[3] J. Mairal, "Optimization with first-order surrogate functions," *International Conference on Machine Learning (ICML)*, 2013.

[4] A. Defazio, F. Bach, and S. Lacoste-Julien, "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives," *Advances in neural information processing systems (NIPS)*, 2014.

[5] M. Mahdavi, L. Zhang, and R. Jin, "Mixed optimization for smooth functions," *Advances in neural information processing systems (NIPS)*, 2013.

[6] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," *Advances in neural information processing systems (NIPS)*, 2013.

[7] L. Zhang, M. Mahdavi, and R. Jin, "Linear convergence with condition number independent access of full gradients," *Advances in neural information processing systems (NIPS)*, 2013.

[8] J. Konečný and P. Richtárik, "Semi-stochastic gradient descent methods," *arXiv preprint*, 2013.

[9] M. Schmidt, N. Le Roux, and F. Bach, "Convergence rates of inexact proximal-gradient methods for convex optimization," *Advances in neural information processing systems (NIPS)*, 2011.

[10] C. Hu, J. Kwok, and W. Pan, "Accelerated gradient methods for stochastic optimization and online learning," *Advances in neural information processing systems (NIPS)*, 2009.

[11] L. Xiao and T. Zhang, "A proximal stochastic gradient method with progressive variance reduction," *SIAM Journal on Optimization*, vol. 24, no. 2, pp. 2057–2075, 2014.

[12] S. Lohr, *Sampling: design and analysis*. Cengage Learning, 2009.

[13] M. P. Friedlander and M. Schmidt, "Hybrid deterministic-stochastic methods for data fitting," *SIAM Journal of Scientific Computing*, vol. 34, no. 3, pp. A1351–A1379, 2012.

[14] A. Aravkin, M. P. Friedlander, F. J. Herrmann, and T. Van Leeuwen, "Robust inversion, dimensionality reduction, and randomized sampling," *Mathematical Programming*, vol. 134, no. 1, pp. 101–125, 2012.

[15] S. Rosset and J. Zhu, "Piecewise linear regularized solution paths," *The Annals of Statistics*, vol. 35, no. 3, pp. 1012–1030, 2007.

[16] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods - Support Vector Learning* (B. Schölkopf, C. Burges, and A. Smola, eds.), ch. 11, pp. 169–184, Cambridge, MA: MIT Press, 1999.

[17] N. Usunier, A. Bordes, and L. Bottou, "Guarantees for approximate incremental svms," *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[18] J. Konečný, J. Liu, P. Richtárik, and M. Takáč, "ms2gd: Mini-batch semi-stochastic gradient descent in the proximal setting," *arXiv preprint*, 2014.

[19] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," *Advances in neural information processing systems (NIPS)*, 2007.

[20] R. H. Byrd, G. M. Chin, J. Nocedal, and Y. Wu, "Sample size selection in optimization methods for machine learning," *Mathematical programming*, vol. 134, no. 1, pp. 127–155, 2012.

[21] K. van den Doel and U. Ascher, "Adaptive and stochastic algorithms for EIT and DC resistivity problems with piecewise constant solutions and many measurements," *SIAM J. Scient. Comput*, vol. 34, 2012.