Fast Variance Reduction Method with Stochastic Batch Size

Xuanqing Liu 1  Cho-Jui Hsieh 1 2

Abstract

In this paper we study a family of variance reduction methods with randomized batch size—at each step, the algorithm first randomly chooses the batch size and then selects a batch of samples to conduct a variance-reduced stochastic update. We give the linear convergence rate for this framework for composite functions, and show that the optimal strategy to achieve the optimal convergence rate per data access is to always choose batch size of 1, which is equivalent to the SAGA algorithm. However, due to the presence of cache/disk IO effect in computer architecture, the number of data access cannot reflect the running time because of 1) random memory access is much slower than sequential access, 2) when data is too big to fit into memory, disk seeking takes even longer time. After taking these into account, choosing batch size of 1 is no longer optimal, so we propose a new algorithm called SAGA++ and show how to calculate the optimal average batch size theoretically. Our algorithm outperforms SAGA and other existing batched and stochastic solvers on real datasets. In addition, we also conduct a precise analysis to compare different update rules for variance reduction methods, showing that SAGA++ converges faster than SVRG in theory.

1. Introduction

In this paper, we consider the following finite-sum composite optimization problem:

\[ w^* = \arg \min_{w \in \mathbb{R}^p} \left\{ F(w) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(w) + g(w) \right\}, \]

Here we assume each \( f_i(w) \) is a \( \mu \)-strongly convex, \( L \)-smooth function, the regularization term \( g(w) \) is convex but not necessarily differentiable. In machine learning applications, \( n \) is the number of training samples, each \( f_i \) is the loss function such as logistic loss or \( \ell_2 \) loss, and \( g(w) \) is the regularization term which can be non-smooth (e.g., \( \ell_1 \) regularization). For large data, SGD is preferred over gradient descent and has been widely used in large-scale applications. However, since the variance of stochastic gradient will not go to zero even when \( w = w^* \) (the optimal solution), SGD has to gradually shrink the step size to guarantee convergence, at the cost of suboptimal rate. To speed up the convergence, there is a recent line of research on developing new algorithms with linear convergence rate using variance reduction techniques, the representative work includes SAG (Hofmann et al., 2015), SVRG (Johnson & Zhang, 2013), SAGA (Defazio et al., 2014), S2GD (Konecný & Richtárik, 2013) etc. Further, one can accelerate this framework via concepts similar to the Nesterov’s momentum method (Lin et al., 2015; Allen-Zhu, 2017).

The motivation of this paper is to study the effect of batch size in variance reduction methods. The effect of batch size in SGD (without variance reduction) has been studied in the literature such as (Li et al., 2014; Bengio, 2012; Keskar et al., 2016). Assuming a subset of \( b \) samples is chosen for SGD at each step, the theoretical analysis in (Dekel et al., 2012) suggests that the error is at the order of \( O(1/\sqrt{bT} + 1/T) \) after \( T \) iterations, and this bound is later improved to \( O(1/\sqrt{bT}) \) in (Li et al., 2014). When constraining on SVM-hinge loss, (Takáč et al., 2013) also shows an order of \( O\left(\frac{b}{b} + \frac{\delta_{2}}{b} \cdot \frac{1}{T}\right) \) iterations to get an \( \epsilon \)-suboptimal solution. Since each iteration will take the time proportional to \( b \), these bounds suggest that the acceleration of convergence exactly covers the overhead of each iteration. It is thus interesting to see whether the same conclusion also applies for variance reduction methods.

To answer this question, we study a family of variance reduction methods with randomized batch sizes. At each iteration, the algorithm first randomly selects the batch size and then chooses a batch of samples to conduct a variance reduced stochastic update. Our main findings and contributions can be listed as follows:

- We prove linear convergence rate for this family of stochastic batched variance reduction algorithms. Our
result covers composite minimization problems with non-smooth regularizations, and any distribution of batch sizes.

- Interestingly, with this unified analysis, we theoretically show that the convergence rate can be maximized if the algorithm always chooses batch size of 1. Therefore, increasing batch size does not help in terms of the number of data access.
- However, the number of data access does not precisely reflect the actual running time due to the memory hierarchy and cache/disk IO effect in computer architectures—accessing a continuous block of memory is faster than accessing disjoint ones, and disk seeking costs even more. After taking these into account, we propose the SAGA++ algorithm, and show how to calculate the optimal average batch size in practice. Our algorithm outperforms existing algorithms in terms of running time.
- In addition, we also develop a more precise analysis for comparing the convergence rates of variance reduction methods, and develop an algorithm to universally accelerate the stochastic methods for solving $\ell_1$-regularized problems by lazy updates. Which rediscover (Konečný et al., 2016) independently.

**Related Work**

We will discuss the related variance reduction methods in next section. Here we describe some other related work on stochastic optimization.

Stochastic optimization has become popular due to their vast and far reaching applications in large-scale machine learning, and this is also one of our main focus in this paper. Among them, stochastic gradient descent has been widely used, and its variants (Duchi et al., 2011; Kingma & Ba, 2014) are popular for training deep neural networks. There are also other examples, such as stochastic coordinate descent (Nesterov, 2012) and stochastic dual coordinate descent (Shalev-Shwartz & Zhang, 2013). At each iteration, SGD selects one sample to conduct the update, but its gradient often contains huge noise. To reduce the noise or variance, mini-batch SGD has been intensively studied in the literature, including (Li et al., 2014), and the recent work on big-batch SGD (De et al., 2016). Some theoretical results have been discussed in our introduction (Li et al., 2014; Dekel et al., 2012).

Although some recent works have discussed about mini-batch variance reduction algorithms (Hofmann et al., 2015; Harikandeh et al., 2015), there is no clear conclusion on whether increasing the batch size helps the convergen speed. Ideally the convergence rate $1 - \rho$ should linearly depend on the batch size: $\rho \propto b$; if that is the case, simply by calculating the batched gradient in parallel we will see linear speed up. (Hofmann et al., 2015) suggests $\rho \approx b/n$ in big data regime and $\rho$ is independent of $b$ in ill-conditioned case, this can be regarded as an asymptotic situation of our result, which claims that $\rho$ is a increasing function of $b$, but a larger batch size is less useful when the Hessian is ill conditioned. However, with a more precise bound in terms of $b$, we are able to show that $b = 1$ is always optimal in terms of number of data access.

As to the sampling techniques, the random sampling of batch size is seen in (Richtárik & Takáč, 2016) where the authors considered about the partially separable functions and apply block coordinate descent by randomly generate a set of blocks with arbitrary size. Similar idea is later exploited in (Qu et al., 2015; Csiba & Richtárik, 2015). Our idea differs from these previous works in that we put computer architecture effects into account when deciding whether we should choose full gradient or stochastic gradient to update parameters.

### 2. Framework: Variance Reduction with Stochastic Batch Size

Our proposed framework is shown in Algorithm 1: at each iteration, the algorithm first randomly chooses the batch size, ranging from 1 to $n$, and then samples a batch accordingly. We use $B$ as a random set to denote the mini-batch chosen at each step, and its batch size, denoted as $|B|$, is a random variable. Denote $f'_i(x_t^i)$ as the previous gradient evaluated on sample $x_t$ and $w^t$ is the iterate at time $t$. The update rule is given by:

$$w^{t+1} = \text{Prox}_{\gamma g}(w^t - \gamma G(w^t)),$$

where $\gamma$ is the step size and $G(w^t)$ is the unbiased gradient estimator:

$$G(w^t) = \frac{1}{|B|} \sum_{i \in B} f'_i(x_t^i)^t - \frac{1}{|B|} \sum_{i \in B} f'_i(x_t^i) + \bar{u},$$

where $\bar{u} = \frac{1}{n} \sum_{i=1}^n f'_i(x_t^i)$ is stored and maintained in memory. Similar to SAGA, in general the algorithm needs to store all the vectors $f'_i(x_t^i)$ in memory, but for many commonly used cases it only needs to store a scalar for each sample index $i$. For example, in GLM problems where $f_i(w) = \ell_i(x_i^T w)$, since $f'_i(x_i^T w) = x_i \ell_i'(x_i^T w)$ we only need to store a scalar $\ell_i'(x_i^T w)$ for each $i \in \{1, 2, \ldots, n\}$.

Algorithm 1 is very general and can include most of the existing variance reduction methods because our algorithm does not put any restriction on the choice of batch size, which can be either random or fixed. We discuss the connections between this framework and others:

- When the batch size is $n$ with probability 1, Algorithm 1 will compute the full gradient at each iteration, which is equivalent to gradient descent.
We discuss our theoretical results and new insights in this section. First, we prove the linear convergence rate of Algorithm 1 in Section 3.1, and then in Section 3.2 we will take the cache/disk IO effect into consideration to derive the new algorithm SAGA++. We show the SAGA-style update used in this paper is more efficient than SVRG-style update in Section 3.3 and then discuss a new technique to conduct lazy update for l1 regularization in our Algorithm 1. We left the proof in appendix.

3.1. Convergence rate analysis

We assume the objective function is µ-strongly convex and L-Lipschitz smooth, and κ = L/µ is the condition number. We will use the following useful bounds in our analysis:

\[ f(y) \geq f(x) + f'(x)(y - x) + \mu/2\|y - x\|^2 \quad (4a) \]
\[ f(y) \leq f(x) + f'(x)(y - x) + L/2\|y - x\|^2. \quad (4b) \]

Hereafter we use \( \cdot \) to denote \ell_2 norm unless stated explicitly. To simplify notation, we define \( f_i^\delta(w) = f_i(w) - f_i(w^*) - f_i'(w^*)(w - w^*) \) and \( f_i^\delta(w) = 1/n \sum_{i=1}^n f_i^\delta(w) \) as the Bregman divergence between \( w \) and \( w^* \), and define \( \bar{H}_t = 1/n \sum_{i=1}^n f_i^\delta(\phi_i^t) \) to represent the averaged Bregman divergence between \( w^* \) and the snapshots \( \phi_i^t \).

To show the convergence of Algorithm 1, we first calculate the expected change of \( \bar{H}_t \) after each update:

**Lemma 1** For the update rule (2) and \( \bar{H}_t \) defined above, we have \( \mathbb{E}\bar{H}_{t+1} = \frac{n-|\Omega|}{n}\bar{H}_t + \frac{|\Omega|}{n} f_i^\delta(w^*). \)

Note that unlike (Hofmann et al., 2015) where \( |\Omega| \) is deterministic, here we generalize their result to allow random batch size. Similarly, the progress of \( \|w^t - w^*\| \) can be bounded by:

**Lemma 2** Define iteration progress by the distance to the optimal solution \( w^* \), then:

\[ \|w^t - w^*\|^2 - \mathbb{E}\|w^{t+1} - w^*\|^2 \geq \gamma \mu \|w^t - w^*\|^2 \]
\[ -((1 + \beta)\gamma^2 - \gamma/L)\mathbb{E}\|f_i(w^t) - f_i(w^*)\|^2 \]
\[ \gamma^2 \beta \|f_i(w^t) - f_i(w^*)\|^2 + \frac{2\gamma(1 - \mu)}{L} \mathbb{E}\|f_i^\delta(w^t) - f_i^\delta(w^*)\|^2 \]
\[ -2(1 + \beta^{-1})L\gamma^2 \bar{H}_t, \]

where \( \beta > 0 \) is an arbitrary constant.

Combining Lemma 1 with 2, we can build a contraction on Lyapunov function as follows.

**Theorem 3** Define Lyapunov function as \( \mathcal{L}_t = c\bar{H}_t + \|w^t - w^*\|^2 \), \( c \) is a predefined constant, then we have \( \mathbb{E}\mathcal{L}_{t+1} \leq (1 - \rho)\mathcal{L}_t \) where \( \rho = \min\left(\frac{\mathbb{E}|\Omega|}{n} - \frac{2(1 + \beta^{-1})L\gamma^2}{\gamma\mu}, \gamma\mu\right) \) if the step size \( \gamma \) satisfies the following conditions:

**Upper bound**

\[ \gamma \leq \min \left( \frac{1}{1 + \beta}, \sqrt{\frac{c\mathbb{E}|\Omega|}{2(1 + \beta^{-1})nL}} \right). \]

**Lower bound**

\[ 2\mu\beta\gamma^2 + \frac{2(L - \mu)}{L} \gamma - \frac{c\mathbb{E}|\Omega|}{n} \geq 0. \]

Recall \( c, \beta \) are predefined constants.
However, the result above is too complex to interpret. To get a better understanding of how the averaged batch size \(E[B]\), step size \(\gamma\) and contraction factor \(\rho\) are related to each other, we simplify the result along different directions.

**Proposition 1** (Adaptive step size) In this case we want the step size \(\gamma\) to be independent on strong convexity \(\mu\). To this end, we set \(\beta = 2\), \(c = \frac{n}{\sqrt{4\log(1/\rho)}}\), so that \(\gamma = \frac{1}{4\sqrt{n}}\).

This result is the same with the adaptive step size of SAGA, which trade simplicity with tightness (the step size and convergence rate is independent on \(E[B]\) so we can not see the benefit of larger batch). To develop a more informative result, we resort to the following proposition:

**Proposition 2** (\(E[B]\)-dependent step size) If we set step size to \(\gamma = \frac{c}{\sqrt{n}} \left(\sqrt{1 + \frac{16\log (E[B])}{cn\mu}} - 1\right)\) and \(c = \frac{1}{\sqrt{4\log(1/\rho)}}\), \(\tau \in (0, 1)\) is a constant, our algorithm converges linearly with a contraction factor \(1 - \rho\), i.e. \(\|w^t - w^*\|^2 \leq (1 - \rho)^t \|w^0 - w^*\|^2 + cH_0\) and \(\rho = \gamma\mu\).

The selection of step size in Proposition 2 is optimal in terms of maximizing the convergence rate, as proven in appendix. Admittedly, after developing these results, the convergence rate and step size are loose after many inequalities, so these bounds should be regarded as the worst case situation. Even so, as a quick verification, we can show that our result matches the bounds of gradient descent and SAGA in the following extreme cases:

- **Gradient descent:** when setting \(P(|B| = n) = 1\), then \(\rho_{GD} = \frac{\tau}{\sqrt{n}} \left(\sqrt{1 + \frac{16\log (E[B])}{cn\mu}} - 1\right) \approx \frac{\sqrt{\tau}}{2\sqrt{n}}\) this gives the same order as the standard rate of gradient descent \(\frac{\mu}{2\sqrt{n}p}\).
- **SAGA:** \(P(|B| = 1) = 1\), for the ill-conditioned case where \(\kappa\) is comparable to \(n\), \(\rho = \frac{1}{2}\cdot\text{constant} \approx O\left(\frac{1}{n}\right)\).

While in the well-conditioned case, \(\kappa \ll n\), we have \(\rho = \frac{\tau}{\sqrt{\kappa}} \left(1 + \frac{16\log (E[B])}{\tau n\kappa} - 1\right) \approx O\left(\frac{1}{n}\right)\). These rates match the results in the original SAGA paper (Defazio et al., 2014).

Note that the step size, unlike SGD, is always bounded away from zero (we do not need to decrease the step size in each epoch). This can be seen from the fact that \(\gamma = \frac{c}{\sqrt{n}} \left(\sqrt{1 + \frac{16\log (E[B])}{cn\mu}} - 1\right) \geq 0\).

Next we try to find out the optimal averaged batch size \(E[B]\) in order to achieve the optimal convergence rate in terms of number of data access. Based on Proposition 2, to return an \(\epsilon\)-accurate solution we need \(O\left(\frac{\log \epsilon}{\log(1 - \rho)}\right)\) iterations, which implies \(O\left(\frac{\log \epsilon}{\log(1 - \rho)} \ E[B]\right)\) epochs if the averaged batch size is \(E[B]\). We then derive the following corollary to show that simply increasing the batch size will slow down the convergence rate per data access:

**Corollary 1** (Theoretically optimal batch size) Since the effective number of data access per iteration is proportional to the averaged batch size \(E[B]\), the optimal batch size should maximize the decrement of function value with fixed gradient calculation, which can be formulated as follows:

\[
E[B]^* = \arg \min_{E[B]} \frac{\log \epsilon}{\log(1 - \rho)} E[B] = \arg \min_{E[B]} \frac{E[B]}{\rho} \\
= \frac{\arg \min_{E[B]} E[B]}{\log(1 - \rho)} \left(1 + \frac{16\log (E[B])}{\tau n\kappa} - 1\right)^{-1}.
\]

By taking derivative it is easy to see that the function is monotone increasing with \(E[B]\) (we leave it in appendix). So theoretically \(E[B] = 1\) (which is SAGA method) is optimal.

### 3.2. SAGA++: Optimal batch sizes when taking cache/disk IO effect into consideration

According to Corollary 1, one should always choose \(|B| = 1\) in order to minimize the number of data accesses. However, small number of data access may not necessarily lead to short running time in practice—in modern computer architectures, “sequential accesses” of data stored in the memory is much faster than “random accesses”, because accessing the memory wildly can result in frequent cache miss or disk seeking. Therefore, calculating the full gradient \(f'(w)\) will take less time than calculating \(n\) random gradient components \(f'_i(w)\) (see Table 1 for measurement result). This leads to a new variance reduction method with non-deterministic batch size selection strategy (SAGA++) that combines full gradient access and SAGA (\(|B| = 1\)): at each step we choose \(|B| = n\) with probability \(p\) and \(|B| = 1\) with probability \(1 - p\). When \(|B| = n\), SAGA++ accesses the whole dataset and this can be relatively fast due to the sequential memory access pattern, while when \(|B| = 1\) it randomly accesses one sample. By changing \(p\) we can smoothly change the average batch size from 1 to \(n\). Next we show how to take the cache/disk IO effect into consideration and derive the “optimal” \(p\) in theory, while in the experimental part we show that the optimal average batch size can be large, depending on the problem and data.

To derive the optimal average batch size that yields minimal running time, we assume the computer needs \(T_{seq}\) time to sequentially access \(n\) samples, and \(T_{rand}\) to randomly access the same number of samples. Therefore, when \(|B| = 1\), each update costs within \(T_{rand}/n\) time units. We call the ratio \(T_{seq}/T_{rand}\) as the cache effect ratio.

**Corollary 2** (Optimal batch size with cache/disk IO effect) If \(T_{seq}/T_{rand} = \eta\), \(\eta < 1\), then the optimal average
Figure 1. a) solution of (8) when cache/disk IO effect coefficient \( \eta = 0.7 \), the optimal batch size \( \mathbb{E}B^* \) is the intersection of two lines (marked as blue and orange), in this plot the condition number ranges from 0.025\( n \) to 0.5\( n \). b) To see the relation of \( \mathbb{E}B^* \) and condition number \( \kappa \) more clearly, we solve (8) numerically, the optimal batch size drop rapidly when \( \kappa \) grows. At the same level of condition number, we should use a larger average batch when the cache/disk IO effect is strong (marked as blue and orange), in this plot the condition number ranges from 0.025\( n \) to 0.5\( n \). c,d) Experiment on \texttt{avazu} dataset(cache/disk IO effect ratio \( \eta = 0.46 \)), with respect to both data access(gradient calculation) and running time.

SAGA++ will keep updating them even when batch size = 1. Since we always keep the latest information, the convergence speed is always better than SVRG, and we leave the detailed discussion to Section 3.3.

3.3. One step analysis: comparing Algorithm 1 with SVRG-style update

By far, we have only discussed the convergence speed under SAGA-style framework. In this section, we analyze why Algorithm 1 has faster convergence rate compared to the SVRG-style updates. This explains why SAGA++ (a special case of SVRG) is faster than SVRG, since they have the same data access pattern and differs in update rules. Here SVRG-style means we do not update the control variate in (3) before a new gradient snapshot is calculated, while in Algorithm 1 we store and update each gradient memorization \( f_i'(\phi_i) \) as well as its summation once “fresher” gradient is available. Since the proposed framework in Algorithm 1 includes SAGA and SAGA++ as special cases, we call it “SAGA-style” update hereafter.

The main advantage of SVRG-style update is that it needs less memory, however, since many machine learning problems can be formulated as generalized linear model (GLM): \( f_i(w) = f(w^\top x_i) \), so the gradient \( f_i'(w) = f'(w^\top x_i)x_i \) is
purely determined by $w^\top x_i$, and SAGA-style update need only to store this scalar instead of the gradient vector for each sample. Therefore, for GLM problems the memory overhead of SAGA-style algorithm is simply an $O(n)$ vector.

In terms of convergence rate, the following theory indicates that SAGA-style updates can better control the variance of gradient. First of all, we extend (3) to a more general variance reduced gradient defined by $G(w^t) = f'_i(w^t) - g_i$, where $g_i = \alpha_i - \frac{1}{n} \sum_{j=1}^{n} \alpha_i$ can be any zero-mean control variate. The update rules for SVRG, SAGA, and SAGA++ can be written as $\alpha_i = f'_i(w^\tau)$, where:

**SVRG:** $\tau = kT$

**SAGA:** $0 \leq \tau \leq t$

**SAGA++:** $kT \leq \tau \leq t$,

where $f'_i(w^\tau) = f'_i(\phi^\tau_i)$ is stored in memory, $T$ is the number of inner iterations inside each outer iteration and suppose the program have just finished the $k$-th outer iteration. We only consider the $|E| = 1$ case since we want to focus on the control variate rather than batch size. For each $i$, by regarding $\tau$ as a random variable, we can calculate its probability distribution as follows:

$$
\tau_{SVRG} = kT, \quad \tau_{SAGA} = \begin{cases} 
0, & p = \left(1 - \frac{1}{n}\right)^t \\
1, & p = \frac{1}{n} \left(1 - \frac{1}{n}\right)^{t-1} \\
\vdots & \vdots \\
t, & p = \frac{1}{n}
\end{cases}
$$

$$
\tau_{SAGA++} = \begin{cases} 
kT, & p = \left(1 - \frac{1}{n}\right)^{t-kT} \\
kT + 1, & p = \frac{1}{n} \left(1 - \frac{1}{n}\right)^{t-kT-1} \\
\vdots & \vdots \\
t, & p = \frac{1}{n}
\end{cases}
$$

(10)

To see the difference of convergence rate between those update rules, we introduce the following lemmas:

**Lemma 4** If we use the distance $\|w^t - w^*\|^2$ as a metric to the sub-optimality, then we have:

$$
\mathbb{E}[\|w^{t+1} - w^*\|^2 | \mathcal{F}_t] \leq (1 - \gamma \mu) \|w^t - w^*\|^2 + (4L\gamma^2 - \frac{2\mu \gamma}{L} - 2\mu \gamma^2) f^2(w^t) + 2\gamma^2 \mathbb{E}[\|\alpha_i - f'_i(w^*)\|^2 | \mathcal{F}_i],
$$

(11)

where the expectation is taken over the choices of $i (\mathcal{F}_t$ is the $\sigma$-algebra at time $t$), $f^2(w) = f(w) - f(w^*) - f'(w^*)(w - w^*)$ is the Bregman divergence and we have $0 \leq f^2(w) \leq F(w) - F(w^*)$.

The first two terms in (11) is related to the distance between the current and optimal solution, only the last term involves the control variate $\alpha_i$ in different update rules, that is exactly what we are interested in, which can be further bounded by:

**Lemma 5** For an algorithm with $P(\tau = j) = p_j$, we can upper bound the gradient difference term:

$$
\mathbb{E}[\|\alpha_i - f'_i(w^*)\|^2 | \mathcal{F}_0] \leq 2\gamma \sum_{j=1}^{t} p_j F^{\text{sub}}(w^j).
$$

(12)

To see the change of $F^{\text{sub}}(w^t) = F(w^t) - F(w^*)$ with $t$, we claim that those variance reduction methods is expected to decrease in each step as long as $\gamma$ is small (but keep to some constant):

**Proposition 3** For strongly convex function $f(w)$, define the update rule $w^{t+1} = w^t - \gamma G(w^t)$ (we ignore the regularization term for simplicity). If we want the function value to be a super-martingale, i.e. $\mathbb{E}[f(w^{t+1}) | \mathcal{F}_t] \leq f(w^t)$, then for SGD we require $\gamma \rightarrow 0$. But for variance reduction methods, since the variance of $G(w^t)$ goes to zero as fast as $f(w^t) - f^*$ (see appendix for details), a constant step size is enough.

Finally we can compare the update rules listed in (3.3) by Proposition 3: Since the upper bound of distance improvement in (11) is determined by the variation of control variate $\mathbb{E}[\|\alpha_i - f'_i(w^*)\|^2 | \mathcal{F}_0]$ and it is further upper bounded by (12), this can be seen as a weighted sum of expected function suboptimal $\mathbb{E}[F^{\text{sub}} | \mathcal{F}_0]$ and further from Corollary 3 we know $F^{\text{sub}}$ is expected to decrease at each iteration, so we can conclude that more “weight” $p_j$ should be put on smaller $\mathbb{E}[F^{\text{sub}} | \mathcal{F}_0]$, in another word, a good update rule should keep all the stochastic gradient active, rather than stale for too long. Therefore, by (10) we can observe that the distribution of $p_j$ in SAGA++ is strictly better than both SAGA and SVRG, which indicates that SAGA++ has a faster convergence rate while keeping the same computational cost.

3.4. Lazy update for $\ell_1$ regularization

For sparse datasets and $f_i(w) = f(w^\top x_i)$, the stochastic gradient $f'_i(w^\top x_i)x_i$ has the same indices of zero elements as $x_i$. However, the $\tilde{u}$ vector in update rule (3) is a dense vector, which will lead to updating all the $d$ variables. To reduce the time complexity back to $O(\text{nnz}(x_i))$ per step, a “lazy update” technique was discussed in (Schmidt et al.) for $\ell_1$ regularization. The main idea is that instead of performing an immediate update to all the variables, we only update variables associate with nonzero elements in $x_i$. In the following, we derive the lazy update technique for $\ell_1$ regularization. As an illustration, Figure 2 shows a simple case where index $j$ has two consecutive zero elements in data vectors chosen at time $t = 1$ and $t = 2$, or $x_i[j] = x_i[j] = 0$. 
The fact that doing one step full gradient update is faster than full gradient calculation can be significantly accelerated, if we come back to synchronous updates and given that only paces between workers, these algorithms are suboptimal. So their asynchronous versions that alleviate this problem to sources are idle. Although many first order methods have than the available CPU cores, many of the computing re-

stochastic gradient upgrade when batch size is much smaller

optimization algorithms, in a more implicit way: when do-

read; similar idea can also be applied to a variety of parallel

formal proof here for the completeness of this paper.

Due to the space limit, we left the detailed formulation of

P

have:

prox

1

(3)

(2)

(1)

Sequence methodology and SAGA in all cases. (2) The lazy update (briefly dis-

at the random update stage, thus outperforms both SVRG

and SAGA in all cases. (2) The lazy update (briefly dis-

a lot; without such technique all the variance reduction

Our algorithm, SAGA++, has a sequential access stage like

often outperforms SAGA due to faster sequential access.

First, we compare all the algorithms on kddb dataset with
different regularization parameters. The results in Figure 3
shows that SAGA++ outperforms other algorithms for all
the three choices of parameters. Indeed, $\lambda = 10^{-6}$ (the
middle figure) is the best parameter in terms of prediction
accuracy, so our comparison covers both large and small $\lambda$s.

Next, we compare the running time of all the algorithms on
three datasets in Figure 4. The results show that SAGA++
is faster than all the competitors on these three datasets.
We conclude our experimental results by the following ob-
servations: (1) Although SAGA has faster convergence in
terms of “number of data access” (see Figure 1-c), SVRG
often outperforms SAGA due to faster sequential access.
Our algorithm, SAGA++, has a sequential access stage like
SVRG, while uses the most up-to-date gradient information
at the random update stage, thus outperforms both SVRG
and SAGA in all cases. (2) The lazy update (briefly dis-
cussed in Section 3.4) accelerates SAGA/SVRG/SAGA++
a lot; without such technique all the variance reduction
methods will be much slower than LIBLINEAR. However,
with such technique they can outperform the LIBLINEAR
implentation of proximal Newton methods.

5. Conclusions and Discussions

We study the unified framework for variance reduction meth-
ods with stochastic batch size and prove the linear convergence rate in strongly convex finite sum functions with
continuous regularizer. We show that choosing batch size
always equals to 1 (equivalent to SAGA) leads to the best
rate in terms of number of data accesses; however, it is
Fast Variance Reduction Method with Stochastic Batch Size

Table 1. Dataset statistics. Time for sequential accessing $n$ samples is measured by one computation of full gradient $f'(w)$, while time for random accesses is measured by computing $n$ random gradient components $f'_i(w)$.

| Dataset | Size(GB) | #Sample   | #Feature | nnz ratio | Time to access whole data(sec) |
|---------|----------|-----------|----------|-----------|--------------------------------|
|         |          |           |          |           | Sequential | Random            |
| kddb    | 5.13     | 19,264,097| 29,890,095| 9.84e-7  | 3.91       | 11.43             |
| avazu   | 5.04     | 25,832,830| 999,962  | 1.50e-5  | 4.14       | 9.08              |
| criteo  | 26.74    | 45,840,617| 999,999  | 3.90e-5  | 14.07      | 30.51             |

†Download from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Figure 3. Running time comparison on kddb dataset with different regularization parameters ($\lambda$). Result shows that our SAGA++ algorithm is faster than competitors under different regularization parameters.

Figure 4. Running time comparison among different data ($\lambda = 1.0 \times 10^{-7}$ for all data). Meta-information can be found in Table 1.

not optimal in terms of running time, so we develop a new SAGA++ algorithm. We demonstrate that SAGA++ outperforms SAGA and SVRG in terms of running time, both in theory and in practice.

One reason that SAGA++ outperforms other VR methods is due to the cache/IO effect, although we only shows in-memory optimization, one can imagine that when data is too large to fit in memory and an out-of-core solver is needed, the overhead of IO will be even more significant. In this setting we would expect an even more greater advantage over SVRG/SAGA. Another important reason is that SAGA++ update its control variate more frequently, making the stochastic gradient a lower variance estimator, so intuitively its more close to gradient descent.

Acknowledgements

The authors acknowledge the support of NSF via IIS-1719097 and the computing resources provided by Google cloud and Nvidia.

References

Allen-Zhu, Z. Katyusha: The first direct acceleration of stochastic gradient methods. In Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, pp. 1200–1205. ACM, 2017.

Bengio, Y. Practical recommendations for gradient-based training of deep architectures. In Neural networks: Tricks of the trade, pp. 437–478. Springer, 2012.
Csiba, D. and Richtárik, P. Primal method for erm with flexible mini-batching schemes and non-convex losses. *arXiv preprint arXiv:1506.02227*, 2015.

De, S., Yadav, A., Jacobs, D., and Goldstein, T. Big batch sgd: Automated inference using adaptive batch sizes. *arXiv preprint arXiv:1610.05792*, 2016.

Defazio, A. *New Optimization Methods for Machine Learning*. PhD thesis, PhD thesis, Australian National University, 2014.

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

Dekel, O., Gilad-Bachrach, R., Shamir, O., and Xiao, L. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(Jan):165–202, 2012.

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

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Harikandeh, R., Ahmed, M. O., Virani, A., Schmidt, M., Konečný, J., and Sallinen, S. Stopwasting my gradients: Practical svrg. In *Advances in Neural Information Processing Systems*, pp. 2251–2259, 2015.

Hofmann, T., Lucchi, A., Lacoste-Julien, S., and McWilliams, B. Variance reduced stochastic gradient descent with neighbors. In *Advances in Neural Information Processing Systems*, pp. 2305–2313, 2015.

Hsieh, C.-J., Yu, H.-F., and Dhillon, I. Passcode: Parallel asynchronous stochastic dual co-ordinate descent. In *International Conference on Machine Learning*, pp. 2370–2379, 2015.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pp. 315–323, 2013.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Konecný, J. and Richtárik, P. Semi-stochastic gradient descent methods. *arXiv preprint arXiv:1312.1666*, 2(1):3, 2013.

Konečný, J., Liu, J., Richtárik, P., and Takáč, M. Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):242–255, 2016.

Leblond, R., Pedregosa, F., and Lacoste-Julien, S. Asaga: asynchronous parallel saga. *arXiv preprint arXiv:1606.04809*, 2016.

Li, M., Zhang, T., Chen, Y., and Smola, A. J. Efficient minibatch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 661–670. ACM, 2014.

Lin, H., Mairal, J., and Harchaoui, Z. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, pp. 3384–3392, 2015.

Nesterov, Y. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Qu, Z., Richtárik, P., and Zhang, T. Quartz: Randomized dual coordinate ascent with arbitrary sampling. In *Advances in neural information processing systems*, pp. 865–873, 2015.

Recht, B., Re, C., Wright, S., and Niu, F. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pp. 693–701, 2011.

Reddi, S. J., Hefny, A., Sra, S., Poczos, B., and Smola, A. J. On variance reduction in stochastic gradient descent and its asynchronous variants. In *Advances in Neural Information Processing Systems*, pp. 2647–2655, 2015.

Richtárik, P. and Takáč, M. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156(1-2):433–484, 2016.

Schmidt, M., Le Roux, N., and Bach, F. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, pp. 1–30.

Shalev-Shwartz, S. and Zhang, T. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.
Takác, M., Bijral, A. S., Richtárik, P., and Srebro, N. Minibatch primal and dual methods for svms. In *ICML (3)*, pp. 1022–1030, 2013.

Xiao, L. and Zhang, T. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.
A. Appendix

A.1. Proof of Lemma 1

It is straightforward to see:

\[
\mathbb{E}\tilde{H}_{t+1} = \mathbb{E}\left[ \frac{1}{n} \sum_{i=1}^{n} f_i^t(\phi_i^{t+1}) \right] = \mathbb{E}\left[ \frac{1}{n} \left( \sum_{i \in B} f_i^t(w^t) + \sum_{i \not\in B} f_i^t(\phi_i^t) \right) \right]
\]

\[
= \mathbb{E}\left[ \frac{1}{n} \left( \sum_{i \in B} f_i^t(w^t) + \sum_{i \not\in B} f_i^t(\phi_i^t) \right) | \mathcal{B} = \mathcal{B} \right]
\]

\[
= \frac{1}{n} \left( \mathbb{E}[\mathcal{B}] \sum_{i=1}^{n} f_i^t(w^t) + \frac{n - \mathbb{E}[\mathcal{B}]}{n} \sum_{i=1}^{n} f_i^t(\phi_i^t) \right)
\]

\[
= \frac{\mathbb{E}[\mathcal{B}]}{n} f^t(w^t) + \frac{n - \mathbb{E}[\mathcal{B}]}{n} \tilde{H}_t
\]

The second line of equality comes from the rule of total expectation, where the inner expectation is taken over the index set \( \mathcal{B} \), and the outer expectation is taken over the set cardinality \( |\mathcal{B}| \).

A.2. Proof of Lemma 2

The proof technique is similar to SAGA, as well as a useful inequality (Lemma 4 in (Defazio et al., 2014)):

\[
f(x) \geq f(y) + \langle f'(y), x - y \rangle + \frac{1}{2(L - \mu)} \| f'(x) - f'(y) \|^2
\]

\[
+ \frac{\mu L}{2(L - \mu)} \| x - y \|^2 - \frac{\mu}{L - \mu} \langle f'(x) - f'(y), x - y \rangle.
\]  

(A14)

First of all, by the update rule (2):

\[
\| w^{t+1} - w^* \|^2 = \| \text{Prox}_{\gamma_g}(w^t - \gamma G(w^t)) - \text{Prox}_{\gamma_g}(w^* - \gamma f'(w^*)) \|^2
\]

\[
\leq \| w^t - \gamma G(w^t) - w^* + \gamma f'(w^*) \|^2
\]

\[
= \| w^t - w^* \|^2 - 2\gamma \langle w^t - w^*, G(w^t) - f'(w^*) \rangle + \gamma^2 \| G(w^t) - f'(w^*) \|^2.
\]  

(A15)

The inequality follows from non-expansiveness of proximal operator, notice that our stochastic gradient \( G(w^t) \) is unbiased, take the expectation to the second term and apply (A14) to each \( f_i \) and the average over all \( i \) will go to:

\[
-\mathbb{E}[\langle w^t - w^*, G(w^t) - f'(w^*) \rangle] = -\langle w^t - w^*, f'(w^t) - f'(w^*) \rangle
\]

\[
\leq \langle w^t - w^*, f'(w^*) \rangle + \frac{L - \mu}{L} \langle f'(w^t) - f(w^t) \rangle - \frac{\mu}{2} \| w^t - w^* \|^2
\]

\[
- \frac{1}{2Ln} \sum_{i=1}^{n} \| f'_i(w^*) - f'_i(w^t) \|^2 - \frac{\mu}{L} \langle f'(w^*), w^t - w^* \rangle.
\]  

(A16)
Next we bound the last term in (A15): 

$$
\mathbb{E} \left\| f'(w^*) - G(w^*) \right\|^2 = \mathbb{E} \left\| f'(w^*) - \frac{1}{|B|} \sum_{i \in B} f_i'(w) + \frac{1}{|B|} \sum_{i \in B} f_i'(\phi_i) - \frac{1}{n} \sum_{i = 1}^{n} f_i'(\phi_i) \right\|^2 \\
= \mathbb{E} \left\| \frac{1}{n} \sum_{i = 1}^{n} f_i'(w^*) - \frac{1}{|B|} \sum_{i \in B} f_i'(w^*) - f'(w^*) + \frac{1}{|B|} \sum_{i \in B} f_i'(w^*) \right\|^2 \\
- \mathbb{E} \left\| \frac{1}{n} \sum_{i = 1}^{n} f_i'(\phi_i) - \frac{1}{|B|} \sum_{i \in B} f_i'(\phi_i) - f'(w^*) + \frac{1}{|B|} \sum_{i \in B} f_i'(w^*) \right\|^2 \\
+ \left\| f'(w^*) - \frac{1}{n} \sum_{i = 1}^{n} f_i'(w^*) \right\|^2 \quad (A17) 
$$

In equation (A7) we use the property that $\mathbb{E}|X^2| = \mathbb{E}[X - \mathbb{E}[X]]^2 + \mathbb{E}[X]^2$, now use the inequality $\left\| X + Y \right\|^2 \leq (1 + \beta)\left\| X \right\|^2 + (1 + \beta^{-1})\left\| Y \right\|^2, \beta > 0$ to the first term:

$$
\mathbb{E} \left\| f'(w^*) - G(w^*) \right\|^2 \leq (1 + \beta)\mathbb{E} \left\| \frac{1}{n} \sum_{i = 1}^{n} f_i'(w) - \frac{1}{|B|} \sum_{i \in B} f_i'(w^*) - f'(w^*) + \frac{1}{|B|} \sum_{i \in B} f_i'(w^*) \right\|^2 \\
+ (1 + \beta^{-1})\mathbb{E} \left\| \frac{1}{n} \sum_{i = 1}^{n} f_i'(\phi_i) - \frac{1}{|B|} \sum_{i \in B} f_i'(\phi_i) - f'(w^*) + \frac{1}{|B|} \sum_{i \in B} f_i'(w^*) \right\|^2 \\
+ \beta \cdot \left\| f'(w^*) - \frac{1}{n} \sum_{i = 1}^{n} f_i'(w^*) \right\|^2. \quad (A18) 
$$

Next we bound the first and second terms again by variance decomposition, for simplicity we only take the first term as example:

$$
\mathbb{E} \left\| \frac{1}{n} \sum_{i = 1}^{n} f_i'(w) - \frac{1}{|B|} \sum_{i \in B} f_i'(w^*) - f'(w^*) + \frac{1}{|B|} \sum_{i \in B} f_i'(w^*) \right\|^2 \\
= \mathbb{E} \left\| \frac{1}{|B|} \sum_{i \in B} \left( f_i'(w^*) - f_i'(w^*) \right) \right\|^2 - \mathbb{E} \left\| \frac{1}{n} \sum_{i = 1}^{n} \left( f_i'(w^*) - f_i'(w^*) \right) \right\|^2 \\
\overset{(1)}{=} \mathbb{E} \left\| \frac{1}{|B|} \sum_{i \in B} \left\| f_i'(w^*) - f_i'(w^*) \right\|^2 \right\| - \mathbb{E} \left\| \frac{1}{n} \sum_{i = 1}^{n} \left( f_i'(w^*) - f_i'(w^*) \right) \right\|^2 \\
\overset{(2)}{=} \frac{1}{n} \left\| f_i'(w^*) - f_i'(w^*) \right\|^2 - \frac{1}{n} \left\| f_i'(w^*) - f_i'(w^*) \right\|^2 \\
\leq \frac{1}{n} \left\| f_i'(w^*) - f_i'(w^*) \right\|^2, 
$$
\(\rho\) is by RMS-AM inequality, and in \(\leq\) we drop the negative term. Similarly,
\[
\mathbb{E}\left[ \frac{1}{n} \sum_{i=1}^{n} f_i'(\phi_i^t) - \frac{1}{|B|} \sum_{i \in B} f_i'(\phi_i^t) - f'(w^*) + \frac{1}{|B|} \sum_{i \in B} f_i'(w^*) \right] \leq \frac{1}{n} \sum_{i=1}^{n} \|f_i'(w^*) - f_i'(\phi_i^t)\|^2.
\]

Plug (A19) into (A18) we get:
\[
\mathbb{E}\|f'(w^*) - G(w^*)\|^2 \leq \frac{(1 + \beta)n}{n} \sum_{i=1}^{n} \|f_i'(w^*) - f_i'(w^*)\|^2 + \frac{(1 + \beta^{-1})n}{n} \sum_{i=1}^{n} \|f_i'(w^*) - f_i'(\phi_i^t)\|^2
- \beta\|f'(w^*) - f'(w^*)\|^2.
\]

Combining (A15),(A16),(A20) becomes (5) immediately:
\[
\|w^t - w^*\|^2 - \mathbb{E}\|w^{t+1} - w^*\|^2 \geq \gamma \mu \|w^t - w^*\|^2 - (2\gamma^2 - \gamma/L)\mathbb{E}\|f_i'(w^t) - f_i'(w^*)\|^2
+ \gamma^2\|f'(w^t) - f'(w^*)\|^2 + \frac{2\gamma(L - \mu)}{L} f_0^t(w^*) - 4\gamma^2 L \mathbb{E}\|w^t - w^*\|^2.
\]

A.3. Proof of Theorem 3

It follows directly from Lemma 1 and 2:
\[
L_t - \mathbb{E}L_{t+1} = \frac{1}{H_t} (\|w^t - w^*\|^2 - \mathbb{E}\|w^{t+1} - w^*\|^2)
\geq \frac{c}{n} \left( E|B| - \frac{2(1 + \beta^{-1})\gamma^2 L}{n} \right) H_t + \gamma \mu \|w^t - w^*\|^2 + \frac{2\mu \gamma^2 + \frac{2\gamma(L - \mu)}{L} - \frac{c}{n} \mathbb{E}|B|}{L} f_0^t(w^*)
\]
\[
\geq \frac{c}{n} \left( E|B| - \frac{2(1 + \beta^{-1})\gamma^2 L}{n} \right) H_t + \gamma \mu \|w^t - w^*\|^2
\geq \rho \mathcal{L}_t,
\]
where \(\rho = \min(\frac{|B|}{n} - \frac{2(1 + \beta^{-1})\gamma^2 L}{n}, \gamma \mu)\), the last inequality \(\geq\) comes with following condition:
\[
2\mu \gamma^2 + \frac{2\gamma(L - \mu)}{L} - \frac{c}{n} \geq 0 \quad \text{(A22)}
\]

Furthermore, to keep our algorithm moving forward, i.e. \(\|w^t - w^*\|^2\) decreasing, we should also make sure such condition hold:
\[
\frac{|B|}{n} - \frac{2(1 + \beta^{-1})\gamma^2 L}{c} \geq 0 \quad \text{(A23)}
\]

A.4. Proof of Proposition 1

By plugging \(\beta = 2, c = \frac{n}{3L|B|}\) into (A22) it is easy to verify both inequalities hold.

A.5. Proof of Proposition 2

In this case we choose \(\beta = 1\). From Theorem 3 we know that with a suitable step size \(\gamma\) and \(c\), we have:
\[
\mathbb{E}\|w^t - w^*\|^2 \leq \mathbb{E}L_t \leq (1 - \rho)^t \mathcal{L}_0 = (1 - \rho)^t \left[ \|w^0 - w^*\|^2 + c\mathcal{H}_0 \right].
\]

For the optimal convergence rate, we try to maximize the geometric factor \(\rho = \min(\frac{|B|}{n} - \frac{4\gamma^2 L}{c}, \gamma \mu)\). Denote \(\gamma_0\) as the solution of: \(\frac{|B|}{n} - \frac{4\gamma_0^2 L}{c} = \gamma_0 \mu\). Notice that \(\rho(\gamma) = \gamma \mu\) is increasing with \(\gamma\) when \(\gamma \leq \gamma_0 = \frac{c}{8\mu} \left( \sqrt{1 + \frac{16\mu|B|}{c^2 \mu}} - 1 \right)\) and
We begin with non-expansiveness of proximal operation:
\[ f(w) = \frac{\|B\|}{n} - 4\gamma^2 L \] is decreasing when \( \gamma > \gamma_0 \). So the optimal step size should be \( \gamma = \gamma_0 \). However we should also verify that this step size indeed satisfies the condition in (A22). First of all:

\[
\gamma_0 = \frac{c}{8\kappa} \left( \sqrt{1 + \frac{16\kappa n E[B]}{cn\mu}} - 1 \right) \leq \frac{c}{8\kappa} \sqrt{\frac{16\kappa n E[B]}{cn\mu}} = \sqrt{\frac{c E[B]}{4n L}} \leq \frac{1}{2L}, \tag{A24}
\]

(1) comes from the fact that \( \sqrt{1 + \alpha} - 1 \leq \sqrt{\alpha} \), \( \leq ) \) holds by choosing \( c = \frac{\tau n}{E[B]} \), where \( \tau < 1 \) is a small constant. These two inequalities together ensure the upper bound part of (A22). As to the lower bound, we have \( \sqrt{1 + \alpha} - 1 > \sqrt{\alpha} - 1 \), so:

\[
\gamma_0 > \frac{c}{8\kappa} \left( \sqrt{\frac{16\kappa n E[B]}{cn\mu}} - 1 \right) \geq \frac{c E[B] L}{2n(L - \mu)} \Rightarrow \tau \leq \left( \frac{1}{L - \mu} + \frac{c n}{4E[B]} \right)^2 < 1.
\]

So if we choose \( \tau \) properly, both sides of (A22) can be satisfied.

### A.6. Proof of Corollary 1, 2

Following (7) we take a derivative to \( E[B] \):

\[
\frac{\partial f(E[B])}{\partial E[B]} = \frac{(\alpha E[B] - 1) E[B]}{\sqrt{1 + \alpha^2 E[B]^2} (\sqrt{1 + \alpha^2 E[B]^2} - 1)^2} \geq 0, \tag{A25}
\]

where \( \alpha = \frac{4\kappa}{\sqrt{n}} \), so there is no optimal batch size, and since we always want to access one data point, i.e. \( |B| \geq 1 \) and SAGA style update is optimal.

For Corollary 2, it is easy to see for our algorithm, which choose \( |B| = n \) with probability \( p \ll 1 \) and \( |B| = 1 \) with probability \( 1 - p \), has average batch size \( E[B] = np + 1 - p \approx np + 1 \). For each update, it takes on average time \( \tau = np\eta + (1 - p)\eta \approx (1 + np\eta)\tau \). If we want to get a \( \epsilon \)-suboptimal solution, the total iteration will be \( N = \frac{\log(\epsilon/\eta)}{\log(1 - p)} \propto 1/\rho \). So the running time will be:

\[
T \propto \frac{1 + np\eta}{\sqrt{\frac{1}{E[B]} + \frac{16n^2}{n E[B]}} - \frac{1}{E[B]}} \tag{A26}
\]

\[
\approx \frac{(E[B]^2 - E[B]|\eta + E[B]|)}{1 + \alpha^2 E[B]^2 - 1}. \tag{A27}
\]

For simplicity we denote \( B = E[B] \). By taking the partial derivative and set it to zero \( \partial T / \partial B = 0 \) can solve the best batch size:

\[
((2B) - 1)\eta + (1 + \sqrt{1 + \alpha^2 B^2}) = ((B^2 - B)\eta + B) \frac{\alpha^2 B}{\sqrt{1 + \alpha^2 B^2}}. \tag{A28}
\]

after solving the above equation we get:

\[
B = \left( \frac{1}{\eta} - 1 \right) \left( \xi - 1 \right) \left( \xi - 1 \right), \quad \xi = \frac{\alpha^2 B^2}{1 + \alpha^2 B^2 - \sqrt{1 + \alpha^2 B^2}}. \tag{A29}
\]

By showing the second order derivative \( \partial^2 T / \partial B^2 \geq 0 \) it’s easy to verify that this solution is actually a global minimum.

### A.7. Proof of Lemma 4

We begin with non-expansiveness of proximal operation:

\[
\|w^{t+1} - w^*\|^2 = \|\text{Prox}_{\gamma}(w^t - \gamma G(w^t)) - \text{Prox}_{\gamma}(w^* - \gamma f'(w^*))\|^2 \\
\leq \|w^t - \gamma G(w^t) - w^* + \gamma f'(w^*)\|^2 \\
= \|w^t - w^*\|^2 + 2\gamma(w^t - w^*, G(w^t) - f'(w^*)) + \gamma^2\|G(w^t) - f'(w^*)\|^2, \tag{A30}
\]

where \( f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w) \) By taking expectation on each side and notice \( G(w^*) \) is an unbiased estimation of \( f'(w^*) \):

\[
\mathbb{E}\|w^{t+1} - w^*\|^2 = \|w^t - w^*\|^2 + 2\mathbb{E}(w^t - w^*, f'(w^*) - f'(w^*)) + \gamma^2\mathbb{E}\|G(w^t) - f'(w^*)\|^2, \tag{A31}
\]
and then apply the following bounds for strongly convex function $f$:
\[
\langle w^t - w^*, f'(w^t) - f'(w^*) \rangle \geq \mu \|w^t - w^*\|^2
\]
\[
\langle w^t - w^*, f'(w^t) - f'(w^*) \rangle \geq \frac{1}{L} \|f'(w^t) - f'(w^*)\|^2
\]  
(A31)
so the inner product term have a composite upper bound:
\[
-2\gamma \langle w^t - w^*, f'(w^t) - f'(w^*) \rangle \leq -\gamma (\mu \|w^t - w^*\|^2 + \frac{1}{L} \|f'(w^t) - f'(w^*)\|^2)
\]  
(A32)
on the other hand, we can bound $\mathbb{E}\|G(w^t) - f'(w^*)\|^2$ as (A18) but we only need to care about one sample in a batch case, since we are comparing SAGA with SVRG update style:
\[
\mathbb{E}\|G(w^t) - f'(w^*)\|^2 \leq 2\mathbb{E}\|f'_i(\phi^t_i) - f'_i(w^*)\|^2 + 2\mathbb{E}\|f'_i(w^t) - f'_i(w^*)\|^2 - \|f'(w^t) - f'(w^*)\|^2.
\]  
(A33)
Remember we have proved above formula in (A20), for $\mathbb{E}\|f'_i(w^t) - f'_i(w^*)\|^2$ we have:
\[
\mathbb{E}\|f'_i(w^t) - f'_i(w^*)\|^2 \leq \frac{2L}{n} \sum_{i=1}^{n} f_i(w^t) - f_i(w^*) - f'_i(w^*)^\top (w^t - w^*)
\]  
(A34)
Similarly, for $\|f'(w^t) - f'(w^*)\|^2$ we recall $f$ is a $\mu$-strongly convex function:
\[
\|f'(w^t) - f'(w^*)\|^2 \geq 2\mu (f(w^t) - f(w^*) - f'(w^*)^\top (w^t - w^*)).
\]  
(A35)
Add those inequalities together:
\[
\mathbb{E}\|w^{t+1} - w^*\|^2 \leq (1 - \gamma \mu)\|w^t - w^*\|^2 + (4L\gamma^2 - \frac{2\mu \gamma}{L} - 2\mu \gamma^2) f^k(w^t) + 2\gamma^2 \mathbb{E}\|f'_i(\phi^t_i) - f'_i(w^*)\|^2.
\]  
(A36)

### A.8. Proof of Lemma 5

Since we know the distribution of random variable $\tau$, also denote $t_s$ as the index of the latest gradient snapshot so for SVRG/SAGA++ $t_s = kT$ where $k$ is the number of outer iteration and $T$ is the length of inner iteration, for SAGA $t_s = 0$, so in either method we have $t_s \geq 0$ then by conditional expectation relationship:
\[
\mathbb{E}\|\alpha_i - f'_i(w^*)\|^2 | \mathcal{F}_0 \]  
\[
\overset{(1)}{=} \frac{1}{n} \sum_{k=1}^{n} \mathbb{E}\|\alpha_k - f'_k(w^*)\|^2 | \mathcal{F}_0 \]
\[
\overset{(2)}{=} \frac{1}{n} \sum_{k=1}^{n} \sum_{i=t_s}^{t} p_i \|f'_k(w^t) - f'_k(w^*)\|^2
\]
\[
= \frac{1}{n} \sum_{i=t_s}^{t} p_i \|f'_i(w^t) - f'_i(w^*)\|^2
\]
\[
\leq 2L \sum_{i=t_s}^{t} p_i (f(w^t) - f(w^*) - f'(w^*) (w^t - w^*))
\]  
(A37)
$(1)$ is taken over the choices of $i$, while $(2)$ is taken over the random variable $\tau$ in $\alpha_k = f'_k(w^\tau)$. Because the regularization function $g(w)$ is convex, and from optimal condition we know: $-f'(w^*) \in \partial g(w^*)$, we have:
\[
f(w^t) - f(w^*) - f'(w^*) (w^t - w^*) = f(w^t) - f(w^*) + v^t (w^t - w^*)
\]
\[
\leq f(w^t) - f(w^*) + g(w^t) - g(w^*)
\]  
(A38)
\[
= F(w^t) - F(w^*),
\]where $v^t \in \partial g(w^t)$. Finally we have $\mathbb{E}\|\alpha_i - f'_i(w^*)\|^2 | \mathcal{F}_0 \]  
\[
\leq 2L \sum_{i=t_s}^{t} p_i (F(w^t) - F(w^*)).
A.9. Proof of Proposition 3

Recall the quadratic upper bound of $L$-Lipschitz function:

$$f(w^t - \gamma G(w^t)) \leq f(w^t) - \gamma \nabla f^T(w^t)G(w^t) + \frac{L\gamma^2}{2} \|G(w^t)\|^2.$$  \hfill (A39)

By taking the expectation,

$$
\mathbb{E}[f(w^t - \gamma G(w^t))|\mathcal{F}_t] \leq f(w^t) - \gamma \mathbb{E}[\|f(w^t)\|^2] + \frac{L\gamma^2}{2} \mathbb{E}[\|G(w^t)\|^2|\mathcal{F}_t] \\
\leq f(w^t) - (\gamma - \frac{L\gamma^2}{2}) \|\nabla f(w^t)\|^2 + \frac{L\gamma^2}{2} \mathbb{E}[\text{Var}(G(w^t))].
$$  \hfill (A40)

On the other hand, for $\mu$-strongly convex $f$, we have:

$$\|\nabla f(w^t)\|^2 \geq 2\mu(f(w^t) - f^*),$$  \hfill (A41)

so if $\text{Var}(G(w^t))$ also converges to zero at the order of $f^*(w^t) = f(w^t) - f^*$ then $\gamma$ can keep to a small constant rather than damping like SGD. In fact (Xiao & Zhang, 2014)(Corollary 3) already proved it for SVRG, here we prove a similar result for SAGA style update:

$$
\text{Var}(G(w^t)|\mathcal{F}_s) = \mathbb{E}[\left\|\nabla f_k(w^t) - \nabla f_k(\phi_{k,j}^t) - \frac{1}{n} \sum_{j=1}^{n} (\nabla f_j(w^t) - \nabla f_j(\phi_{j,k}^t))\right\|^2 |\mathcal{F}_s] \\
= \mathbb{E}[\left\|\nabla f_k(w^t) - \nabla f_k(\phi_{k,j}^t)\right\|^2 |\mathcal{F}_s] - \left\|\frac{1}{n} \sum_{j=1}^{n} (\nabla f_j(w^t) - \nabla f_j(\phi_{j,k}^t))\right\|^2 \\
\leq \mathbb{E}[\left\|\nabla f_k(w^t) - \nabla f_k(\phi_{k,j}^t)\right\|^2 |\mathcal{F}_s] \\
= \frac{2}{n} \sum_{j=1}^{n} \mathbb{E}[\left\|\nabla f_j(w^t) - \nabla f_j(\phi_{j,k}^t)\right\|^2 |\mathcal{F}_s] + \frac{2}{n} \sum_{j=1}^{n} \mathbb{E}[\left\|\nabla f_j(\phi_{j,k}^t) - \nabla f_j(w^*)\right\|^2 |\mathcal{F}_s] \\
\leq 4L(\mathbb{E}[f(w^t)|\mathcal{F}_s] - f(w^*)) + \frac{4L}{n} \sum_{j=1}^{t} \sum_{\tau=s}^{m} p_{\tau}(\mathbb{E}[f(w_{\tau}|\mathcal{F}_s] - f(w^*)) \\
= 4L(\mathbb{E}[f(w^t)|\mathcal{F}_s] - f(w^*)) + 4L \sum_{\tau=s}^{m} p_{\tau}(\mathbb{E}[f(w_{\tau}|\mathcal{F}_s] - f(w^*)�),
$$  \hfill (A42)

here $\{\mathcal{F}_t\}_{t\geq0}$ is the filtered probability space, $t - T \leq s \leq t$ (recall $T$ is the length of inner iteration) is the latest available full gradient time stamp, $p_\tau$ is the probability distribution of stored gradient discussed in (10). Since $t - s$ is upper bounded (this is true for SVRG/SAGA++, as to SAGA, the expectation is $n \log n$ by “Coupon collection problem”), together with linear convergence, we know the second term is close to the first term up to a constant.

A.10. Proof of Theorem 6

First of all, we have the following recursive formula:

$$P_g(x, \eta, c, n) = \text{Prox}_g(P_g(x, \eta, c, n - 1) - c) \\
= \begin{cases} 
P(x, \eta, c, n - 1) - c - \eta, & \text{if } P(x, \eta, c, n - 1) \geq c + \eta \\
0, & \text{if } c - \eta \leq P(x, \eta, c, n - 1) \leq c + \eta \\
P(x, \eta, c, n - 1) - c + \eta, & \text{if } P(x, \eta, c, n - 1) \leq c - \eta 
\end{cases}.$$  \hfill (A43)

Because $c$ can be either positive or negative but $\eta$ is always positive, we consider about following cases:

- $(c < -\eta)$ In this case $0 > c + \eta > c - \eta$, if:
1. $x \geq c + \eta$, then $P(x, \eta, c, n) = x - n(c + \eta)$;
2. $x < c + \eta$, then suppose $x = q(c - \eta) + \epsilon$, $q \in \mathbb{N}$, $\epsilon \in [c - \eta, c + \eta]$, if $q \geq n$ then $P(x, \eta, c, n) = x - n(c - \eta)$; else $P(x, \eta, c, q) = \epsilon$, $P(x, \eta, c, q + 1) = 0$, $P(x, \eta, c, n) = -(n - q - 1)(c + \eta)$.

- $(c > \eta)$ In this case $0 < c - \eta < c + \eta$ which is symmetric to previous case, if:
  1. $x \leq c - \eta$, then $P(x, \eta, c, n) = x - n(c - \eta)$;
  2. $x > c - \eta$, then suppose $x = q(c + \eta) + \epsilon$, $q \in \mathbb{N}$, $\epsilon \in [c - \eta, c + \eta]$, if $q \geq n$ then $P(x, \eta, c, n) = x - n(c - \eta)$; else $P(x, \eta, c, q) = \epsilon$, $P(x, \eta, c, q + 1) = 0$, $P(x, \eta, c, n) = -(n - q - 1)(c - \eta)$.
- $(-\eta \leq c \leq \eta)$ finally, $c - \eta \leq 0 \leq c + \eta$, if:
  1. $x \geq n(c + \eta)$, then $P(x, \eta, c, n) = x - n(c + \eta)$;
  2. $x \leq n(c - \eta)$, then $P(x, \eta, c, n) = x + n(c - \eta)$;
  3. otherwise, $[\frac{c - \eta}{c + \eta}] < n$ or $[\frac{c + \eta}{c - \eta}] < n$ then we know it will eventually be zero: $P(x, \eta, c, n) = 0$.

Clearly this is a piecewise linear function with tangent either 1 or 0.

### A.11 $\ell_2$ Logistic Regression Experiment

In this supplemental experiment, we conduct the $\ell_2$ logistic regression experiment, formulated as follows

$$w^* = \arg\min_w \frac{1}{n} \sum_{i=1}^{n} \log (1 + \exp(y_i x_i^T w)) + \frac{\lambda}{2} \|w\|_2^2. \quad (A44)$$

The datasets and settings are the same as $\ell_1$ experiment discussed in the main text. The experiment result is exhibited in Figure 5.

![Figure 5](image)

**Figure 5.** Running time comparison among different data ($\lambda = 1.0 \times 10^{-7}$ for all data).