# SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives

Aaron Defazio, Francis Bach, Simon Lacoste-Julien

# SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives

Aaron Defazio
NICTA
Australian National University, Canberra

Francis Bach
INRIA - Sierra Project-Team
École Normale Supérieure, Paris, France

Simon Lacoste-Julien
INRIA - Sierra Project-Team
École Normale Supérieure, Paris, France

July 17, 2014

**Abstract**

In this work we introduce a new optimisation method called SAGA in the spirit of SAG, SDCA, MISO and SVRG, a set of recently proposed incremental gradient algorithms with fast linear convergence rates. SAGA improves on the theory behind SAG and SVRG, with better theoretical convergence rates, and has support for composite objectives where a proximal operator is used on the regulariser. Unlike SDCA, SAGA supports non-strongly convex problems directly, and is adaptive to any inherent strong convexity of the problem. We give experimental results showing the effectiveness of our method.

## 1 Introduction

Remarkably, recent advances [1, 2] have shown that it is possible to minimise strongly convex finite sums provably faster in expectation than is possible without the finite sum structure. This is significant for machine learning problems as a finite sum structure is common in the empirical risk minimisation setting. The requirement of strong convexity is likewise satisfied in machine learning problems in the typical case where a quadratic regulariser is used.

In particular, we are interested in minimising functions of the form

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x),$$

where $x \in \mathbb{R}^d$, each $f_i$ is continuous and convex, and has Lipschitz continuous derivative with constant $L$. We will also consider the case where each $f_i$ is strongly convex with constant $\mu$, and the "composite" (or proximal) case where an additional regularisation function is added:

$$F(x) = f(x) + h(x),$$

where $h$ is continuous but potentially non-differentiable, and where the proximal operation of $h$ is easy to compute.

Our contributions are as follows. In Section 2 we describe the SAGA algorithm, a novel incremental gradient method. In Section 5 we prove theoretical convergence rates for SAGA in the strongly convex case better than those for SAG [1] and SVRG [3], and a factor of 2 from the SDCA [2] convergence rates. These

rates also hold in the composite setting. Additionally, we show that like SAG but unlike SDCA, our method is applicable to non-strongly convex problems without modification. We establish theoretical convergence rates for this case also. In Section 3 we discuss the relation between each of the fast incremental gradient methods, showing that each stems from a very small modification of another.

## 2 SAGA Algorithm

We start with some known initial vector $x^0 \in \mathbb{R}^d$ and known derivatives $f_i'(\phi_i^0) \in \mathbb{R}^d$ with $\phi_i^0 = x^0$ for each $i$. These derivatives are stored in a table data-structure of length $n$, or alternatively a $n \times d$ matrix.

Given the value of $x^k$ and of each $f_i'(\phi_i^k)$ at the end of iteration $k$, the updates for iteration $k+1$ is as follows:

1. Pick a $j$ uniformly at random.
2. Take $\phi_j^{k+1} = x^k$, and store $f_j'(\phi_j^{k+1})$ in the table. All other entries in the table remain unchanged. The quantity $\phi_j^{k+1}$ is not explicitly stored.
3. Update $x$ using $f_j'(\phi_j^{k+1})$, $f_j'(\phi_j^k)$ and the table average:

$$w^{k+1} = x^k - \frac{1}{\eta} f_j'(\phi_j^{k+1}) + \frac{1}{\eta}\left[f_j'(\phi_j^k) - \frac{1}{n}\sum_{i=1}^{n} f_i'(\phi_i^k)\right], \tag{1}$$

$$x^{k+1} = \text{prox}_\eta^h\left(w^{k+1}\right). \tag{2}$$

The proximal operator we use above is defined as

$$\text{prox}_\eta^h(y) := \underset{x\in\mathbb{R}^d}{\text{argmin}} \left\{h(x) + \frac{\eta}{2}\|x-y\|^2\right\}. \tag{3}$$

In the strongly convex case, when a step size of $\eta = 2(\mu n + L)$ is chosen we have the following convergence rate in the composite and hence also the non-composite case:

$$\mathbb{E}\left\|x^k - x^*\right\|^2 \leq \left(1 - \frac{\mu}{2(\mu n + L)}\right)^k \left[\left\|x^0 - x^*\right\|^2 + \frac{1}{\mu n + L}\left[f(x^0) - \left\langle f'(x^*), x^0 - x^*\right\rangle - f(x^*)\right]\right].$$

We prove this result in Section 5. The requirement of strong convexity can be relaxed from needing to hold for each $f_i$ to just holding on average, but at the expense of a worse geometric rate $(1 - \frac{\mu}{6(\mu n+L)})$, requiring a step size of $\eta = 3(\mu n + L)$.

In the non-strongly convex case, we have established the convergence rate in terms of the average iterate, excluding step 0: $\bar{x}^k = \frac{1}{k}\sum_{t=1}^{k} x^t$. Using a step size of $\eta = 3L$ we have

$$\mathbb{E}\left[F(\bar{x}^k)\right] - F(x^*) \leq \frac{3n}{k}\left[\frac{3L}{2n}\left\|x^0 - x^*\right\|^2 + f(x^0) - \left\langle f'(x^*), x^0 - x^*\right\rangle - f(x^*)\right].$$

This result is proved in the Appendix. That same step size can work in both the strongly convex and non-strongly convex case, allowing the algorithm to automatically adapt to the level of strong convexity naturally present. If $\eta = 3L$ is used on strongly convex problems, the geometric constant changes to $\left(1 - \min\left\{\frac{1}{2n}, \frac{\mu}{3L}\right\}\right)$.

## 3 Related Work

In the non-composite case, it is possible to write the SAGA algorithm in terms of two quantities at each step instead of one: $x^k$ and $u^k$.

**Non-composite case**
Given the value of $u^k$ and of each $f_i'(\phi_i^k)$ at the end of iteration $k$, the updates for iteration $k+1$, is as follows:

1. Calculate $x^k$:
$$x^k = u^k - \frac{1}{\eta}\sum_{i=1}^{n} f_i'(\phi_i^k). \tag{4}$$
2. Update $u$ with $u^{k+1} = u^k + \frac{1}{n}(x^k - u^k)$.
3. Pick a $j$ uniformly at random.
4. Take $\phi_j^{k+1} = x^k$, and store $f_j'(\phi_j^{k+1})$ in the table replacing $f_j'(\phi_j^k)$. All other entries in the table remain unchanged. The quantity $\phi_j^{k+1}$ is not explicitly stored.

Writing the algorithm in this form makes the relationship with prior methods more apparent. We explore the relationship between SAGA and the other fast incremental gradient methods in this section. By using SAGA as a midpoint, we are able to provide a more unified view than is available in the existing literature. A brief summary of the properties of each method considered in this Section is given in Figure 1.

## SAG

If we eliminate $x^k$ we get an update for $u$ in SAGA of:

$$u^{k+1} = u^k - \frac{1}{\eta n}\sum_{i=1}^{n} f_i'(\phi_i^k). \tag{5}$$

After translating notation, this is identical to the SAG (Stochastic Average Gradient) [1] update, except instead of setting $\phi_j^{k+1} = u^k$, we are using a more aggressive update of $\phi_j^{k+1} = x^k = u^k - \frac{1}{\eta}\sum_i f_i'(\phi_i^k)$. The order of the steps is also changed, as in SAG the $j$th gradient is updated before the $x$ step is taken, where as above it is updated after. However the order of the steps for SAG doesn't effect the algorithm so the ordering change is not significant.

The significance of this change is the effect it has on our current estimate of the gradient. In SAG, the gradient approximation $\frac{1}{n}\sum_i f_i'(\phi_i^k)$ is biased away from the true gradient, whereas for all the other methods considered here, including SAGA, the gradient approximation is unbiased. The trade-off for the increased bias is a decreased variance in the $f_i'(\phi_i^k)$ gradients, due to the less aggressive $\phi_i^k$ updates used.

The per update cost of SAGA and SAG is essentially the same. The advantage over SAG is that SAGA has a much more complete, simple and tight theory. The theoretical convergence rate is better for SAGA, and no theory exists for the use of proximal operators in SAG.

## SVRG/S2GD

Recall the $x^{k+1}$ update for SAGA (Equation 1) in the non-composite case:

$$x^{k+1} = x^k - \frac{1}{\eta}f_j'(x^k) + \frac{1}{\eta}\left[f_j'(\phi_j^k) - \frac{1}{n}\sum_{i=1}^{n} f_i'(\phi_i^k)\right]. \tag{6}$$

This can be compared against the SVRG (Stochastic Variance Reduced Gradient) [3] update:

$$x^{k+1} = x^k - \frac{1}{\eta}f_j'(x^k) + \frac{1}{\eta}\left[f_j'(\tilde{x}) - \frac{1}{n}\sum_{i=1}^{n} f_i'(\tilde{x})\right].$$

The vector $\tilde{x}$ is not updated every step, but rather the loop over $k$ appears inside an outer loop, where $\tilde{x}$ is updated at the start of each outer iteration. Essentially SAGA is at the midpoint between SVRG and SAG; it

| | SAG | SDCA | SVRG | FINITO | SAGA |
|---|---|---|---|---|---|
| Strongly Convex (SC) | ✓ | ✓ | ✓ | ✓ | ✓ |
| Convex, Non-SC* | ✓ | ✗ | ? | ? | ✓ |
| Prox Reg. | ? | ✗ | ✓ | ✗ | ✓ |
| Non-smooth | ✗ | ✓ | ✗ | ✗ | ✗ |
| Low Storage Cost | ✗ | ✗ | ✓ | ✗ | ✗ |
| Simple(-ish) Proof | ✗ | ✓ | ✓ | ✓ | ✓ |
| Adaptive to SC | ✓ | ✗ | ? | ? | ✓ |

Figure 1: Basic summary of method properties. Question marks denote unproven, but not experimentally ruled out cases. (*) Note that any method can be applied to non-strongly convex problems by adding a small amount of L2 regularisation, this row describes methods that do not require this trick.

updates the $\phi_j$ value each time index $j$ is picked, whereas SVRG updates all of $\phi$'s as a batch. The S2GD method [4] has the same update as SVRG, just differing in how the number of inner loop iterations is chosen. We use SVRG henceforth to refer to both methods.

SVRG makes a trade-off between time and space. For the equivalent practical convergence rate it makes 2x-3x more gradient evaluations, but in doing so it does not need to store a table of gradients, but a single average gradient. The usage of SAG v.s. SVRG is problem dependent. For example for linear predictors where gradients can be stored as a reduced vector of dimension $p-1$ for $p$ classes, SAGA is preferred over SVRG both theoretically and in practice. For neural networks, where no theory is available for either method, the storage of gradients is generally more expensive than the additional backwards propagations, but this is computer architecture dependent.

SVRG also has an additional parameter besides step size that needs to be set, namely the number of iterations per inner loop ($m$). This parameter can be set via the theory, or conservatively as $m = n$, however doing so does not give anywhere near the best practical performance. Having to tune one parameter instead of two is a practical advantage for SAGA.

## Finito/MISO$\mu$

The Finito [5] and MISO$\mu$ [6] methods are also closely related to SAGA. Both Finito and MISO$\mu$ use updates of the following form, for a step length $\eta$:

$$x^{k+1} = \frac{1}{n}\sum_i \phi_i^k - \frac{1}{\eta}\sum_{i=1}^n f_i'(\phi_i^k).$$

Note that the step sized used is of the order of $\mu n$, roughly comparable to the $\eta$ in SAGA. This should be contrasted with the much smaller $\eta n$ step size used in SAG. We will introduce the notation $\bar{\phi} = \frac{1}{n}\sum_i \phi_i^k$ to simplify the discussion of this algorithm.

SAGA can be interpreted as Finito, but with the quantity $\bar{\phi}$ replaced with $u$, which is updated in the same way as $\bar{\phi}$, but *in expectation*. To see this, consider how $\bar{\phi}$ changes in value and in expectation:

$$\mathbb{E}\left[\bar{\phi}^{k+1}\right] = \mathbb{E}\left[\bar{\phi}^k + \frac{1}{n}\left(x^k - \phi_j^k\right)\right] = \bar{\phi}^k + \frac{1}{n}\left(x^k - \bar{\phi}^k\right).$$

The update is identical in expectation to the update for $u$, $u^{k+1} = u^k + \frac{1}{n}(x^k - u^k)$.

There are three advantages of SAGA over Finito/MISO$\mu$. SAGA doesn't require strong convexity to work, it has support for proximal operators, and it doesn't require storing the $\phi_i$ values. MISO has proven support for proximal operators only in the case where impractically small step sizes are used [6]. The big advantage of Finito/MISO$\mu$ is that when it is applicable, it can be used with a per-pass repermuted access ordering, which can make it up to 2x faster. Finito/MISO$\mu$ is particularly useful when $f_i$ is computationally expensive to compute compared to the extra storage costs required over the other methods.

## SDCA

The Stochastic Dual Coordinate Descent (SDCA) [2] method on the surface appears quite different from the other methods considered. It works with the convex conjugates of the $f_i$ functions. However, in this section we show a novel transformation of SDCA into an equivalent method that only works with primal quantities, and is closely related to the MISO$\mu$ method.

Consider the following algorithm:

**SDCA algorithm in the primal**

Step $k+1$:

1. Pick an index $j$ uniformly at random.
2. Compute $\phi_j^{k+1} = \text{prox}_\eta^{f_j}(z)$, where $\eta = \mu n$ and $z = -\frac{1}{\eta}\sum_{i\neq j}^n f_i'(\phi_i^k)$.
3. Store the gradient $f_j'(\phi_j^{k+1}) = \eta\left(z - \phi_j^{k+1}\right)$ in the table at location $j$. For $i \neq j$, the table entries are unchanged ($f_i'(\phi_i^{k+1}) = f_i'(\phi_i^k)$).

At completion, return $x^k = -\frac{1}{\eta}\sum_i^n f_i'(\phi_i^k)$ .

We claim that this algorithm is equivalent to the version of SDCA where exact block-coordinate maximisation is used on the dual.[1] Firstly, note that while SDCA was originally described for one-dimensional outputs (binary classification or regression), it has been expanded to cover the multi-class predictor case [7] (called Prox-SDCA there). In this case, the primal objective has a separate strongly convex regulariser, and the functions $f_i$ are restricted to the form $f_i(x) := \psi_i(X_i^T x)$, where $X_i$ is a $d \times p$ feature matrix, and $\psi_i$ is the loss function that takes a $p$ dimensional input, for $p$ classes. To stay in the same general setting as the other incremental gradient methods, we work directly with the $f_i(x)$ functions rather than the more structured $\psi_i(X_i^T x)$. The dual objective to maximise then becomes

$$D(\alpha) = \left[-\frac{\mu}{2}\left\|\frac{1}{\mu n}\sum_{i=1}^n \alpha_i\right\|^2 - \frac{1}{n}\sum_{i=1}^n f_i^*(-\alpha_i)\right],$$

where $\alpha_i$'s are $d$-dimensional dual variables. Generalising the exact block-coordinate maximisation update that SDCA performs to this form, we get the dual update for block $j$ (with $x^k$ the current primal iterate):

$$\alpha_j^{k+1} = \alpha_j^k + \underset{\Delta a_j \in \mathbb{R}^d}{\text{argmax}}\left\{-f_j^*\left(-\alpha_j^k - \Delta\alpha_j\right) - \frac{\mu n}{2}\left\|x^k + \frac{1}{\mu n}\Delta\alpha_j\right\|^2\right\}. \tag{7}$$

In the special case where $f_i(x) = \psi_i(X_i^T x)$, we can see that (7) gives exactly the same update as Option I of Prox-SDCA in [7, Figure 1], which operates instead on the equivalent $p$-dimensional dual variables $\tilde{\alpha}_i$ with the relationship that $\alpha_i = X_i\tilde{\alpha}_i$.[2] As noted by Shalev-Shwartz & Zhang [7], the update (7) is actually an instance of the proximal operator of the convex conjugate of $f_j$. Our primal formulation exploits this fact by using a relation between the proximal operator of a function and its convex conjugate known as the Moreau decomposition:

$$\text{prox}^{f^*}(v) = v - \text{prox}^f(v).$$

This decomposition allows us to compute the proximal operator of conjugate via the primal proximal operator. As this is the only use in the basic SDCA method of the conjugate function, applying this decomposition

[1] More precisely, to Option I of Prox-SDCA as described in [7, Figure 1]. We will simply refer to this method as "SDCA" in this paper for brevity.

[2] This is because $f_i^*(\alpha_i) = \inf_{\tilde{\alpha}_i \text{ s.t. } \alpha_i = X_i\tilde{\alpha}_i} \psi_i^*(\tilde{\alpha}_i)$.

allows us to completely eliminate the "dual" aspect of the algorithm, yielding the above primal form of SDCA. The dual variables are related to the primal representatives $\phi_i$'s through $\alpha_i = -f_i'(\phi_i)$. The KKT conditions ensure that if the $\alpha_i$ values are dual optimal then $x^k = \frac{1}{\eta}\sum_i \alpha_i$ as defined above is primal optimal. The same trick is commonly used to interpret Dijkstra's set intersection as a primal algorithm instead of a dual block coordinate descent algorithm [8].

The primal form of SDCA differs from the other incremental gradient methods described in this section in that it assumes strong convexity is induced by a separate strongly convex regulariser, rather than each $f_i$ being strongly convex. We now show how to modify SDCA so that it works without a separate regulariser, giving a method that is at the midpoint between Finito and SDCA. Using Lagrangian duality theory, SDCA can be shown at step $k$ as minimising the following lower bound:

$$A^k(x) = \frac{1}{n} f_j(x) + \frac{1}{n}\sum_{i \neq j}^{n} \left[ f_i(\phi_i^k) + \left\langle f_i'(\phi_i^k), x - \phi_i^k \right\rangle \right] + \frac{\mu}{2}\left\|x\right\|^2.$$

Instead of directly including the regulariser in this bound, we can use the standard strong convexity lower bound for each $f_i$, by removing $\frac{\mu}{2}\left\|x\right\|^2$ and changing the expression in the summation to $f_i(\phi_i^k) + \left\langle f_i'(\phi_i^k), x - \phi_i^k \right\rangle + \frac{\mu}{2}\left\|x - \phi_i\right\|^2$. The transformation to having strong convexity within the $f_i$ functions yields the following simple modification to the algorithm: $\phi_j^{k+1} = \text{prox}_{\mu(n-1)}^{f_j}(z)$, where:

$$z = \frac{1}{n-1}\sum_{i \neq j} \phi_i^k - \frac{1}{\mu(n-1)}\sum_{i \neq j} f_i'(\phi_i^k).$$

It can be shown that after this update:

$$x^{k+1} = \phi_j^{k+1} = \frac{1}{n}\sum_i \phi_i^{k+1} - \frac{1}{\mu n}\sum_i f_i'(\phi_i^{k+1}).$$

Now the similarity to Finito is apparent if this equation is compared Equation 3: $x^{k+1} = \frac{1}{n}\sum_i \phi_i^k - \frac{1}{\eta}\sum_{i=1}^n f_i'(\phi_i^k)$. The only difference is that the vectors on the right hand side of the equation are at their values at step $k+1$ instead of $k$. Note that there is a circular dependency here, as $\phi_j^{k+1} := x^{k+1}$ but $\phi_j^{k+1}$ appears in the definition of $x^{k+1}$. Solving the proximal operator is the resolution of the circular dependency. This mid-point between Finito and SDCA is interesting in it's own right, as it appears experimentally to have similar robustness to permuted orderings as Finito, but it has no tunable parameters like SDCA.

When the proximal operator above is fast to compute, say on the same order as just evaluating $f_j$, then SDCA can be the best method among those discussed. It is a little slower than the other methods discussed here, but it has no tuneable parameters at all. It is also the only choice when each $f_i$ is not differentiable. The major disadvantage of SDCA is that it can not handle non-strongly convex problems directly. Although like most methods, adding a small amount of quadratic regularisation can be used to recover a convergence rate. It is also not adapted to use proximal operators *for the regulariser* in the composite objective case. The requirement of computing the proximal operator of each loss $f_i$ initially appears to be a big disadvantage, however there are variants of SDCA, discussed in the next section, that remove this requirement, but they introduce additional downsides.

## Other SDCA variants

The SDCA theory has been expanded to cover a number of other methods of performing the coordinate step [7]. These variants replace the proximal operation in our primal interpretation in the previous section with an update where $\phi_j^{k+1}$ is chosen so that:

$$f_j'(\phi_j^{k+1}) = (1-\beta) f_j'(\phi_j^k) + \beta f_j'(x^k).$$

Where $x^k = -\frac{1}{\mu n}\sum_i f_i'(\phi_i^k)$. The variants differ in how $\beta \in [0,1]$ is chosen. Note that $\phi_j^{k+1}$ does not actually have to be explicitly known, just the gradient $f_j'(\phi_j^{k+1})$, which is the result of the above interpolation. Variant 5 by Shalev-Shwartz & Zhang [7] does not require operations on the conjugate function, it simply uses $\beta = \frac{\mu n}{L+\mu n}$. The most practical variant performs a line search involving the convex conjugate to determine $\beta$. As far as we are aware, there is no simple primal equivalent of this line search. So in cases where we can not compute the proximal operator from the standard SDCA variant, we have the choice of either introducing a tuneable parameter into the algorithm ($\beta$), or the use of a dual line search, which requires an efficient way to evaluate the convex conjugates of each $f_i$.

# 4 Implementation

We briefly discuss some implementation concerns:

- We give three equivalent formulations of the SAGA algorithm in this paper, Equations (4), (5) and (6). If adapting existing SAG code, it may be best to implement using (5). For the composite loss case, Equation (6) is the most natural.
- The SAGA update as stated is slower than necessary when derivatives are sparse. A just-in-time updating of $u$ or $x$ may be performed just as is suggested for SAG [1], which ensures that only sparse updates are done at each iteration.
- We give the form of SAGA for the case where each $f_i$ are strongly convex. However in practice we usually have only convex $f_i$, with strong convexity in $f$ induced by the addition of a quadratic regulariser. This quadratic regulariser may be split amount the $f_i$ functions evenly, to satisfy our assumptions. It is perhaps easier to use a variant of SAGA where the regulariser $\frac{\mu}{2}||x||^2$ is explicit, such as the following modification of Equation (6):

$$x^{k+1} = \left(1 - \frac{\mu}{\eta}\right) x^k - \frac{1}{\eta} f_j'(x^k) + \frac{1}{\eta}\left[f_j'(\phi_j^k) - \frac{1}{n}\sum_i f_i'(\phi_i^k)\right].$$

  For sparse implementations instead of scaling $x^k$ at each step, a separate scaling constant $\beta^k$ may be scaled instead, with $\beta^k x^k$ being used in place of $x^k$. This is a standard trick used with stochastic gradient methods.

# 5 Theory

In this section, to lighten the notation, we drop the superscript on quantities whose values are at iteration $k$ (i.e. $x \triangleq x^k$). All expectations are taken with respect to the choice of $j$ at iteration $k+1$ and conditioned on $x^k$ and each $f_i'(\phi_i^k)$ unless stated otherwise.

We start with two basic lemmas that just state properties of convex functions.

**Lemma 1.** *Let $f$ be $\mu$-strongly convex and have Lipschitz continuous gradients with constant $L$. Then we have for all $x$ and $y$:*

$$\begin{aligned} f(x) \quad \geq \quad & f(y) + \langle f'(y), x - y\rangle + \frac{1}{2(L-\mu)}\left\|f'(x) - f'(y)\right\|^2 \\ & + \frac{\mu L}{2(L-\mu)}\left\|y - x\right\|^2 + \frac{\mu}{(L-\mu)}\left\langle f'(x) - f'(y), y - x\right\rangle. \end{aligned}$$

*Proof.* Define the function $g$ as $g(x) = f(x) - \frac{\mu}{2}\|x\|^2$. Then the gradient is $g'(x) = f'(x) - \mu x$. $g$ has a lipschitz gradient with with constant $L - \mu$. By convexity we have [9, Thm. 2.1.5]:

$$g(x) \geq g(y) + \langle g'(y), x - y\rangle + \frac{1}{2(L-\mu)}\|g'(x) - g'(y)\|^2.$$

Substituting in the definition of $g$ and $g'$, and simplifying terms gives the result. □

**Corollary 1.** *We can apply Lemma 1 to our finite sum structure, where each $f_i$ is $\mu$-strongly convex and has Lipschitz continuous gradients with constant $L$. We get that for all $x$ and $x^*$:*

$$\langle f'(x), x^* - x\rangle \leq \frac{L-\mu}{L}\left[f(x^*) - f(x)\right] - \frac{\mu}{2}\|x^* - x\|^2 - \frac{1}{2Ln}\sum_i \|f_i'(x^*) - f_i'(x)\|^2 - \frac{\mu}{L}\langle f'(x^*), x - x^*\rangle.$$

**Lemma 2.** *We have that for all $\phi_i$ and $x^*$:*

$$\frac{1}{n}\sum_i \|f_i'(\phi_i) - f_i'(x^*)\|^2 \leq 2L\left[\frac{1}{n}\sum_i f_i(\phi_i) - f(x^*) - \frac{1}{n}\sum_i \langle f_i'(x^*), \phi_i - x^*\rangle\right].$$

*Proof.* Apply the standard inequality $f(y) \geq f(x) + \langle f'(x), y - x\rangle + \frac{1}{2L}\|f'(x) - f'(y)\|^2$, with $y = \phi_i$ and $x = x^*$, for each $f_i$, and sum. □

**Lemma 3.** *It holds that for any $\phi_i$, $x^*$, $x^k$ and $\beta > 0$, with $w^{k+1}$ as defined in Equation 1:*

$$\mathbb{E}\left\|w^{k+1} - x^k - \frac{1}{\eta}f'(x^*)\right\|^2 \leq \frac{1+\beta^{-1}}{\eta^2}\mathbb{E}\left\|f_j'(\phi_j) - f_j'(x^*)\right\|^2 + \frac{1+\beta}{\eta^2}\mathbb{E}\left\|f_j'(x^k) - f_j'(x^*)\right\|^2 - \frac{\beta}{\eta^2}\left\|f'(x^k) - f'(x^*)\right\|^2.$$

*Proof.* We follow a similar argument as occurs in the SVRG proof [3] for this term, but with a tighter argument. The tightening comes from using $\|x+y\|^2 \leq (1+\beta^{-1})\|x\|^2 + (1+\beta)\|y\|^2$ instead of the simpler $\beta = 1$ case they use. The other key trick is the use of the standard variance decomposition $\mathbb{E}[\|x - \mathbb{E}[x]\|^2] = \mathbb{E}[\|x\|^2] - \|\mathbb{E}[x]\|^2$ three times.

$$\begin{aligned}
& \mathbb{E}\left\|w^{k+1} - x^k + \frac{1}{\eta}f'(x^*)\right\|^2 \\
= \;& \mathbb{E}\left\|-\frac{1}{\eta n}\sum_i f_i'(\phi_i) + \frac{1}{\eta}f'(x^*) + \frac{1}{\eta}\left[f_j'(\phi_j) - f_j'(x^k)\right]\right\|^2 \\
= \;& \frac{1}{\eta^2}\mathbb{E}\left\|\left[f_j'(\phi_j) - f_j'(x^*) - \frac{1}{n}\sum_i f_i'(\phi_i) + f'(x^*)\right] - \left[f_j'(x^k) - f_j'(x^*) - f'(x^k) + f'(x^*)\right]\right\|^2 \\
& + \frac{1}{\eta^2}\left\|f'(x^k) - f'(x^*)\right\|^2 \\
\leq \;& \frac{1+\beta^{-1}}{\eta^2}\mathbb{E}\left\|f_j'(\phi_j) - f_j'(x^*) - \frac{1}{n}\sum_i f_i'(\phi_i) + f'(x^*)\right\|^2 \\
& + \frac{1+\beta}{\eta^2}\mathbb{E}\left\|f_j'(x^k) - f_j'(x^*) - f'(x^k) + f'(x^*)\right\|^2 + \frac{1}{\eta^2}\left\|f'(x^k) - f'(x^*)\right\|^2 \\
\leq \;& \frac{1+\beta^{-1}}{\eta^2}\mathbb{E}\left\|f_j'(\phi_j) - f_j'(x^*)\right\|^2 + \frac{1+\beta}{\eta^2}\mathbb{E}\left\|f_j'(x^k) - f_j'(x^*)\right\|^2 - \frac{\beta}{\eta^2}\left\|f'(x^k) - f'(x^*)\right\|^2.
\end{aligned}$$

□

**Theorem 1.** *With $x^*$ the optimal solution, take*

$$T = \frac{1}{n}\sum_i f_i(\phi_i) - f(x^*) - \frac{1}{n}\sum_i \langle f_i'(x^*), \phi_i - x^*\rangle + c\|x - x^*\|^2.$$

*Then with $\eta = 2(\mu n + L)$, $c = \frac{\eta^2}{2(\eta-\mu)n}$, and $\kappa = \eta/\mu$, we have that:*

$$\mathbb{E}[T^{k+1}] \leq (1 - \frac{1}{\kappa})T^k.$$

*Proof.* The first three terms in $T^{k+1}$ are straight-forward to simplify:

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{n}\sum_i f_i(\phi_i^{k+1})\right] &= \frac{1}{n}f(x) + \left(1-\frac{1}{n}\right)\frac{1}{n}\sum_i f_i(\phi_i). \\
\mathbb{E}\left[-\frac{1}{n}\sum_i \left\langle f_i'(x^*), \phi_i^{k+1} - x^*\right\rangle\right] &= -\frac{1}{n}\left\langle f'(x^*), x - x^*\right\rangle - \left(1-\frac{1}{n}\right)\frac{1}{n}\sum_i \left\langle f_i'(x^*), \phi_i - x^*\right\rangle.
\end{aligned}$$

For the change in the last term of $T$ we apply the non-expansiveness of the proximal operator[3]:

$$\begin{aligned}
c\left\|x^{k+1} - x^*\right\|^2 &= c\left\|\text{prox}_\eta(w^{k+1}) - \text{prox}_\eta(x^* - \frac{1}{\eta}f'(x^*))\right\|^2 \\
&\leq c\left\|w^{k+1} - x^* + \frac{1}{\eta}f'(x^*)\right\|^2.
\end{aligned}$$

Then we expand the quadratic and apply $\mathbb{E}[w^{k+1}] = x^k - \frac{1}{\eta}f'(x^k)$ to simplify the inner product term:

$$\begin{aligned}
& c\mathbb{E}\left\|w^{k+1} - x^* + \frac{1}{\eta}f'(x^*)\right\|^2 \\
= \;& c\mathbb{E}\left\|x - x^* + w^{k+1} - x + \frac{1}{\eta}f'(x^*)\right\|^2 \\
= \;& c\left\|x - x^*\right\|^2 + 2c\mathbb{E}\left[\left\langle w^{k+1} - x + \frac{1}{\eta}f'(x^*), x - x^*\right\rangle\right] + c\mathbb{E}\left\|w^{k+1} - x + \frac{1}{\eta}f'(x^*)\right\|^2 \\
= \;& c\left\|x - x^*\right\|^2 - \frac{2c}{\eta}\left\langle f'(x) - f'(x^*), x - x^*\right\rangle + c\mathbb{E}\left\|w^{k+1} - x + \frac{1}{\eta}f'(x^*)\right\|^2 \\
\leq \;& c\left\|x - x^*\right\|^2 - \frac{2c}{\eta}\left\langle f'(x), x - x^*\right\rangle + \frac{2c}{\eta}\left\langle f'(x^*), x - x^*\right\rangle - \frac{c\beta}{\eta^2}\left\|f'(x) - f'(x^*)\right\|^2 \\
& + \frac{\left(1+\beta^{-1}\right)c}{\eta^2}\mathbb{E}\left\|f_j'(\phi_j) - f_j'(x^*)\right\|^2 + \frac{(1+\beta)\,c}{\eta^2}\mathbb{E}\left\|f_j'(x) - f_j'(x^*)\right\|^2. \quad \text{(Lemma 3)}
\end{aligned}$$

The value of $\beta$ shall be fixed later. Now we apply Corollary 1 to bound $-\frac{2c}{\eta}\left\langle f'(x), x - x^*\right\rangle$ and Lemma 2 to bound $\mathbb{E}\left\|f_j'(\phi_j) - f_j'(x^*)\right\|^2$:

$$\begin{aligned}
c\mathbb{E}\left\|x^{k+1} - x^*\right\|^2 \leq\; & \left(c - \frac{c\mu}{\eta}\right)\left\|x - x^*\right\|^2 + \left(\frac{(1+\beta)c}{\eta^2} - \frac{c}{\eta L}\right)\mathbb{E}\left\|f_j'(x) - f_j'(x^*)\right\|^2 \\
& - \frac{2c(L-\mu)}{\eta L}\left[f(x) - f(x^*) - \left\langle f'(x^*), x - x^*\right\rangle\right] - \frac{c\beta}{\eta^2}\left\|f'(x) - f'(x^*)\right\|^2 \\
& + \frac{2\left(1+\beta^{-1}\right)cL}{\eta^2}\left[\frac{1}{n}\sum_i f_i(\phi_i) - f(x^*) - \frac{1}{n}\sum_i \left\langle f_i'(x^*), \phi_i - x^*\right\rangle\right].
\end{aligned}$$

We can now combine the bounds we have derived for each term in $T$, and pull out a fraction $\frac{1}{\kappa}$ of $T^k$ (for any $\kappa$ at this point). Together with the inequality $-\left\|f'(x) - f'(x^*)\right\|^2 \leq -2\mu\left[f(x) - f(x^*) - \left\langle f'(x^*), x - x^*\right\rangle\right]$ [9,

[3]Note that the first equality below is the only place in the proof where we use the fact that $x^*$ is an optimality point.

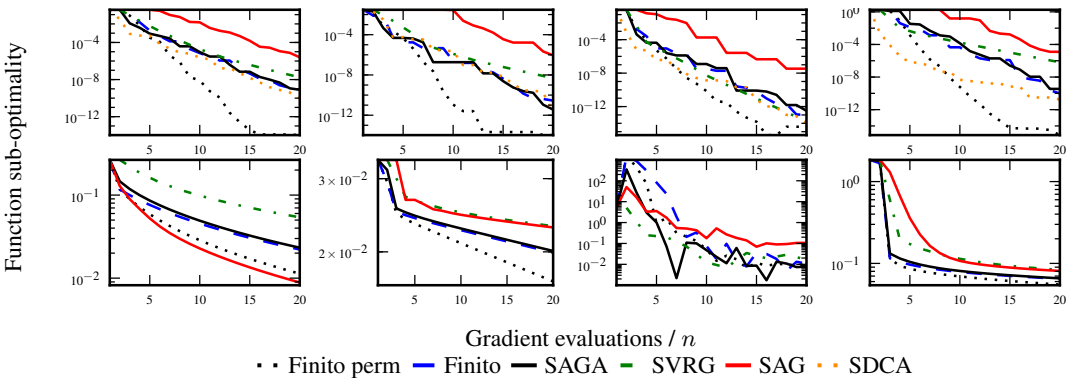


Figure 2: From left to right we have the MNIST, COVTYPE, IJCNN1 and MILLIONSONG datasets. Top row is the L2 regularised case, bottom row the L1 regularised case.

Thm. 2.1.10], that yields:

$$\begin{aligned}\mathbb{E}[T^{k+1}] - T^k \quad \leq \quad & -\frac{1}{\kappa}T^k + \left(\frac{1}{n} - \frac{2c(L-\mu)}{\eta L} - \frac{2c\mu\beta}{\eta^2}\right)\left[f(x) - f(x^*) - \left\langle f'(x^*), x - x^*\right\rangle\right] \\ & + \left(\frac{1}{\kappa} + \frac{2(1+\beta^{-1})cL}{\eta^2} - \frac{1}{n}\right)\left[\frac{1}{n}\sum_i f_i(\phi_i) - f(x^*) - \frac{1}{n}\sum_i \left\langle f_i'(x^*), \phi_i - x^*\right\rangle\right] \\ & + \left(\frac{1}{\kappa} - \frac{\mu}{\eta}\right) c\left\|x - x^*\right\|^2 + \left(\frac{(1+\beta)c}{\eta^2} - \frac{c}{\eta L}\right)\mathbb{E}\left\|f_j'(x) - f_j'(x^*)\right\|^2. \end{aligned} \tag{8}$$

Note that each of the terms in square brackets are positive, and it can be readily verified that our assumed values for the constants ($\eta = 2(\mu n + L)$, $c = \frac{\eta^2}{2(\eta-\mu)n}$, and $\kappa = \eta/\mu$), together with $\beta = \frac{2\mu n + L}{L}$ ensure that each of the quantities in round brackets are non-positive. □

**Corollary 2.** *Note that $c\left\|x^k - x^*\right\|^2 \leq T^k$, and therefore by chaining the expectations, plugging in the constants explicitly and using $\mu(n - 0.5) \leq \mu n$ to simplify the expression, we get:*

$$\mathbb{E}\left[\left\|x^k - x^*\right\|^2\right] \leq \left(1 - \frac{\mu}{2(\mu n + L)}\right)^k \left[\left\|x^0 - x^*\right\|^2 + \frac{1}{\mu n + L}\left[f(x^0) - \left\langle f'(x^*), x^0 - x^*\right\rangle - f(x^*)\right]\right].$$

*Here the expectation is over all choices of index $j^k$ up to step $k$.*

# 6 Experiments

We performed a series of experiments to validate the effectiveness of SAGA. We tested a binary classifier on MNIST, COVTYPE, IJCNN1 and a least squares predictor on MILLIONSONG. Details of these datasets can be found in [5]. We used the same code base for each method, just changing the main update rule. SVRG was tested with the recalibration pass used every $n$ iterations, as suggested in [4]. Each method had its step size parameter chosen so as to give the fastest convergence.

We tested with a L2 regulariser, which all methods support, and with a L1 regulariser on a subset of the methods. The results are shown in Figure 2. We can see that Finito (perm) performs the best on a per epoch equivalent basis, but it can be the most expensive method per step. SVRG is similarly fast on a per epoch basis, but when considering the number of gradient evaluations per epoch is double that of the other methods for this problem, it is middle of the pack. SAGA can be seen to perform similar to the non-permuted Finito case, and to SDCA. Note that SAG is slower than the other methods at the beginning. To get the optimal results for SAG, an adaptive step size rule needs to be used rather than the constant step size we used.

In general, these tests confirm that the choice of methods should be done based on their properties as discussed in Section 3, rather than their convergence rate.

# A Non-strongly-convex problems

**Theorem 2.** *When each $f_i$ is convex, using $\eta = 3L$, we have for $\bar{x}^k = \frac{1}{k}\sum_{t=1}^{k} x^t$ that:*

$$\mathbb{E}\left[F(\bar{x}^k)\right] - F(x^*) \leq \frac{3n}{k}\left[\frac{3L}{2n}\left\|x^0 - x^*\right\|^2 + f(x^0) - \left\langle f'(x^*), x^0 - x^*\right\rangle - f(x^*)\right].$$

*Here the expectation is over all choices of index $j^k$ up to step $k$.*

*Proof.* We proceed by using a similar argument as in Theorem 1, but we add an additional $\alpha\left\|x - x^*\right\|^2$ together with the existing $c\left\|x - x^*\right\|^2$ term in the Lyapunov function.

We will bound $\alpha\left\|x - x^*\right\|^2$ in a different manner to $c\left\|x - x^*\right\|^2$. Define $\Delta = -\eta\left(w^{k+1} - x\right) - f'(x)$, the difference between our approximation to the gradient at $x$ and true gradient. Then instead of using the non-expansiveness property at the beginning, we use a result proved for prox-SVRG [10, 2nd eq. on p.12]:

$$\alpha\mathbb{E}\left\|x^{k+1} - x^*\right\|^2 \leq \alpha\left\|x - x^*\right\|^2 - \frac{2\alpha}{\eta}\mathbb{E}\left[F(x^{k+1}) - F(x^*)\right] + \frac{2\alpha}{\eta^2}\mathbb{E}\left\|\Delta\right\|^2.$$

Although their quantity $\Delta$ is different, they only use the property that $\mathbb{E}[\Delta] = 0$ to prove the above equation. Essentially the same argument as in Lemma 3 can be used to bound the $\Delta$ term yielding

$$\mathbb{E}\left\|\Delta\right\|^2 \leq \left(1 + \beta^{-1}\right)\mathbb{E}\left\|f_j'(\phi_j) - f_j'(x^*)\right\|^2 + (1+\beta)\,\mathbb{E}\left\|f_j'(x^k) - f_j'(x^*)\right\|^2.$$

Applying this gives:

$$\begin{aligned}\alpha\mathbb{E}\left\|x^{k+1} - x^*\right\|^2 \quad &\leq \quad \alpha\left\|x - x^*\right\|^2 - \frac{2\alpha}{\eta}\mathbb{E}\left[F(x^{k+1}) - F(x^*)\right] \\ &\quad + \frac{2(1+\beta^{-1})\alpha}{\eta^2}\mathbb{E}\left\|f_j'(\phi_j) - f_j'(x^*)\right\|^2 + \frac{2\,(1+\beta)\,\alpha}{\eta^2}\mathbb{E}\left\|f_j'(x) - f_j'(x^*)\right\|^2.\end{aligned}$$

As in Theorem 1, we then apply Lemma 2 to bound $\mathbb{E}\left\|f_j'(\phi_j) - f_j'(x^*)\right\|^2$. Combining with the rest of the Lyapunov function, if we take $\eta = 3L$, $\beta = 2$, $c = \frac{L}{n}$ and $\alpha = \frac{L}{2n}$, Then we are left with the following after removing other non-positive terms:

$$\mathbb{E}[T^{k+1}] - T^k \leq -\frac{1}{3n}\mathbb{E}\left[F(x^{k+1}) - F(x^*)\right].$$

These expectations are conditional on information from step $k$. We now take the expectation with respect to all previous steps, yielding $\mathbb{E}[T^{k+1}] - \mathbb{E}[T^k] \leq -\frac{1}{3n}\mathbb{E}\left[F(x^{k+1}) - F(x^*)\right]$, where all expectations are unconditional. Further negating and summing for $k$ from 0 to $k-1$ results in telescoping of the $T$ terms, giving:

$$\frac{1}{3n}\mathbb{E}\left[\sum_{t=1}^{k}\left[F(x^t) - F(x^*)\right]\right] \leq T^0 - \mathbb{E}[T^k].$$

We can drop the $-\mathbb{E}\left[T^k\right]$ term since $T^k$ is always positive. Then we apply convexity to pull the summation inside of $F$, and multiply through by $3n/k$, giving:

$$\mathbb{E}\left[F(\frac{1}{k}\sum_{t=1}^{k}x^t) - F(x^*)\right] \leq \frac{1}{k}\mathbb{E}\left[\sum_{t=1}^{k}\left[F(x^t) - F(x^*)\right]\right] \leq \frac{3n}{k}T^0.$$

□

# References


[1] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. Technical report, INRIA, 2013.

[2] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *JMLR*, 2013.

[3] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *NIPS*, 2013.

[4] Jakub Konečný and Peter Richtárik. Semi-Stochastic Gradient Descent Methods. *ArXiv e-prints*, December 2013.

[5] Aaron Defazio, Tiberio Caetano, and Justin Domke. Finito: A faster, permutable incremental gradient method for big data problems. *Proceedings of the 31st International Conference on Machine Learning*, 2014.

[6] Julien Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. Technical report, INRIA Grenoble Rhne-Alpes / LJK Laboratoire Jean Kuntzmann, 2014.

[7] Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. Technical report, The Hebrew University, Jerusalem and Rutgers University, NJ, USA, 2013.

[8] Patrick Combettes and Jean-Christophe Pesquet. *Proximal Splitting Methods in Signal Processing. In Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. Springer, 2011.

[9] Yu. Nesterov. *Introductory Lectures On Convex Programming*. Springer, 1998.

[10] Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. Technical report, Microsoft Research, Redmond and Rutgers University, Piscataway, NJ, 2014.