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
aaron.defazio@anu.edu.au

Francis Bach
INRIA - Sierra Project-Team
École Normale Supérieure, Paris, France
francis.bach@inria.fr

Simon Lacoste-Julien
INRIA - Sierra Project-Team
École Normale Supérieure, Paris, France
simon.lacoste-julien@ens.fr

July 1, 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 μ, and the "composite" (or proximal) case where an additional regularization 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 $u^0 \in \mathbb{R}^d$ and known derivatives $f_i'(u^0) \in \mathbb{R}^d$ for each i. These derivatives are stored in a table data-structure of length n, or alternatively a $n \times d$ matrix.

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 w^k:
$$w^k = u^k - \frac{1}{\eta}\sum_{i=1}^{n} f_i'(\phi_i^k). \tag{1}$$
2. Update u with $u^{k+1} = u^k + \frac{1}{n}(w^k - u^k)$.
3. Pick a j uniformly at random.
4. Take $\phi_j^{k+1} = w^k$, 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 ϕ_j^{k+1} is not explicitly stored.

In the composite case we leave w defined as above, and we just change step 4 to $\phi_j^{k+1} = x^k$, where

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

Note that the u update remains in terms of w^k rather than x^k.

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

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

We prove this result in Section 5. Note that $L \leq 2\left(L-\mu\right)n$ holds except in extremely easy problems where the condition number is less than or equal 2, and n is very small, such as 1 or 2. The technical condition can be removed, and the requirement of strong convexity can be relaxed from needing to hold for each f_i to just holding on f, but at the expense of a slightly 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

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

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

The non-composite SAGA algorithm in the previous section uses two iterate vectors, w^k and u^k. Writing the algorithm in terms of just one or the other makes the relation 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 much more unified view than is in the existing literature.

SAG

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

$$u^{k+1} = u^k - \frac{1}{\eta n} \sum_{i=1}^{n} f_i'(\phi_i^k). \tag{3}$$

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 $\phi_j^{k+1} = w^k = u^k - \frac{1}{\eta} \sum_i f_i'(\phi_i^k)$. The order of the steps is also changed, as in SAG the jth gradient is updated before the w 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 ϕ_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

If we eliminate u^k from the SAGA algorithm, we get an update of the form:

$$w^{k+1} = w^k - \frac{1}{\eta} f_j'(w^k) + \frac{1}{\eta} \left[f_j'(\phi_j^k) - \frac{1}{n} \sum_{i=1}^{n} f_i'(\phi_i^k) \right]. \tag{4}$$

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

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

The vector $\tilde{w}$ is not updated every step, but rather the loop over k appears inside an outer loop, where $\tilde{w}$ is updated at the start of each outer iteration. Essentially SAGA is at the midpoint between SVRG and SAG; it updates the ϕ_j value each time index j is picked, whereas SVRG updates all of ϕ'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.

	SAG	SDCA	SVRG	FINITO	SAGA
Strongly Convex (SC)	✓	✓	✓	✓	✓
Convex, Non-SC*	✓	✗	?	?	✓
Prox	?	✗	✓	✗	✓
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.

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 1 parameter instead of two is a significant practical advantage for SAGA.

Finito/Misoμ

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

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

Note that the step sized used is of the order of sn, roughly comparable to the η in SAGA. This should be contrasted with the much smaller ηn 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:

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

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

There are three advantages of SAGA over Finito/MISOμ. SAGA doesn't require strong convexity to work, it has support for proximal operators, and it doesn't require storing the ϕ_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/MISOμ 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/MISOμ is particularly useful when f_i is computationally expensive to compute compared to the extra storage costs required over the other methods.

SDCA

The SDCA (Stochastic Dual Coordinate Descent) [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 MISOμ method.

Consider the following algorithm:

SDCA algorithm in the primal
Step $k+1$:

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

We claim that this algorithm is exactly equivalent to SDCA when exact coordinate minimisation is used (the standard variant). Firstly note that while SDCA is normally described as being applicable to linear predictors only, it has been expanded to cover the multi-class predictor case [7]. The functions f_i are restricted to the form $f_i = \psi_i(X_i^T w)$, where X_i is a matrix, and ψ_i takes a p dimensional input, for p classes. Now it is easy to see that if X is taken to be the identity matrix, then we are actually in the same general setting as the other incremental gradient methods, where we just take $f_i = \psi_i$.

The relation between the SDCA and the primal variant we described stems from the fact that the standard dual version performs a minimisation of the following form at each step:

$$\alpha_j^{k+1} = \alpha_j^k + \arg\max_{\Delta a_j} \left[-f_j^*\left(-\alpha^k + \Delta\alpha_j e_j\right)\right] - \frac{\mu n}{2}\left\|w^k + \frac{1}{\mu n}\Delta\alpha_i x_i\right\|^2.$$

Here α are dual variables, e_j the indicator vector, x_i the data point and w^k the current primal location. As noted by Shalev-Shwartz & Zhang [7], this 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 it's 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 allows us to completely eliminate the "dual" aspect of the algorithm. The same trick can be 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 does resemble the other incremental gradient methods described in this section, the primary difference being 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. SDCA can be seen as at step k minimising the following lower bound:

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

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

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

Now the similarity to MISOμ is apparent. The only difference is that the vectors on the right hand side of the equation are at their values at $k+1$ instead of k. Note that there is a circular dependency here, as $\phi_j^{k+1} = w^{k+1}$. Solving the proximal operator is the resolution of the circular dependency.

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 ϕ_j^{k+1} is chosen so that:

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

Where $w^k = -\frac{1}{\mu n}\sum_i f_i'(\phi_i^k)$.The variants differ in how $\beta \in (0,1]$ is chosen. Note that ϕ_j^{k+1} does not actually have to be explicitly known. Variant 5 by Shalev-Shwartz & Zhang [7] does not require operations on the conjugate function, it simply uses $\beta = \frac{\mu n/L}{(1+\mu n/L)}$. The most practical variant performs a line search involving the convex conjugate to determine β. 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 (β), 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 (1), (3) and (4). If adapting existing SAG code, it may be best to implement using (3). For the composite loss case, Equation (4) is the most natural.
- The SAGA update as stated is slower then necessary when derivatives are sparse. A just-in-time updating of u or w 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 $\frac{\mu}{2}||w||^2$ is explicit, such as the following modification of Equation (4):

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

For sparse implementations instead of scaling w^k at each step, a separate scaling constant β may be scaled instead, with βw being used in place of w. 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. $w \triangleq w^k$). All expectations are taken with respect to the choice of j at iteration $k+1$ and conditioned on the quantities w^k, u^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 μ-strongly convex and have Lipschitz continuous gradients with constant L. Then we have:*

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

Proof. Define the function g as $g(x) = f(x) - \frac{\mu}{2}\left\|x\right\|^2$. Then the gradient is $g'(x) = f'(x) - sx$. g has a lipschitz gradient with with constant $L - s$. By convexity we have:

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

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 μ-strongly convex and has Lipschitz continuous gradients with constant L. We get that for all ϕ_i, x and x^*:*

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

Lemma 2. *We have that for all ϕ_i and x^*:*

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

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

Lemma 3. *It holds that for any ϕ_i, x, x^* and $\beta > 0$:*

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

Proof. We follow a similar argument as occurs in the SVRG proof [3] for this term:

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

The first inequality is from $\|x+y\|^2 \leq (1+\beta^{-1})\|x\|^2 + (1+\beta)\|y\|^2$, and the second inequality is an application of the standard probability equality for the variance: $\mathbb{E}[\|x - \mathbb{E}[x]\|^2] = \mathbb{E}[\|x\|^2] - \|\mathbb{E}[x]\|^2 \leq \mathbb{E}[\|x\|^2]$. □

Theorem 1. *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 $\eta = 2(\mu n + L)$, $c = \frac{\eta L}{2(L-\mu)n}$, and $d = \eta/\mu$, as long as $L \leq 2(L-\mu)n$, we have that:

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

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

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

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

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

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:

$$\begin{aligned}
& c\mathbb{E}\left\|w^{k+1} - x^* + \frac{1}{\eta}f'(x^*)\right\|^2 \\
= \; & c\mathbb{E}\left\|x - x^* + w^{k+1} - x + \frac{1}{\eta}f'(x^*)\right\|^2 \\
= \; & c\|x - x^*\|^2 + 2c\mathbb{E}\left[\left\langle w^{k+1} - x + \frac{1}{\eta}f'(x^*), x - x^*\right\rangle\right] + c\mathbb{E}\left\|w^{k+1} - x + \frac{1}{\eta}g'(x^*)\right\|^2
\end{aligned}$$

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

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

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

We can now combine the bounds we have derived for each term in T, and pull out a fraction $\frac{1}{d}$ of T^k to get:

$$
\begin{aligned}
\mathbb{E}[T^{k+1}] - T^k &\leq -\frac{1}{d}T^k + \left(\frac{1}{n} - \frac{2c(L-\mu)}{\eta L}\right)\left[f(x)-f(x^*)-\left\langle f'(x^*), x-x^*\right\rangle\right] \\
&\quad + \left(\frac{1}{d} + \frac{2(1+\beta^{-1})cL}{\eta^2} - \frac{1}{n}\right)\left[\frac{1}{n}\sum_i f_i(\phi_i) - f(x^*) - \frac{1}{n}\sum_i \left\langle f_i'(x^*), \phi_i - x^*\right\rangle\right] \\
&\quad + \left(\frac{1}{d}-\frac{\mu}{\eta}\right)c\left\|x-x^*\right\|^2 + \left(\frac{(1+\beta)c}{\eta^2}-\frac{c}{\eta L}\right)\mathbb{E}\left\|f_j'(x)-f_j'(x^*)\right\|^2. \qquad (5)
\end{aligned}
$$

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

Corollary 2. *Note that $c\left\|x^k-x^*\right\|^2 \leq T^k$, and therefore by chaining expectation and pluging in the constants explicitly, we get:*

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

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

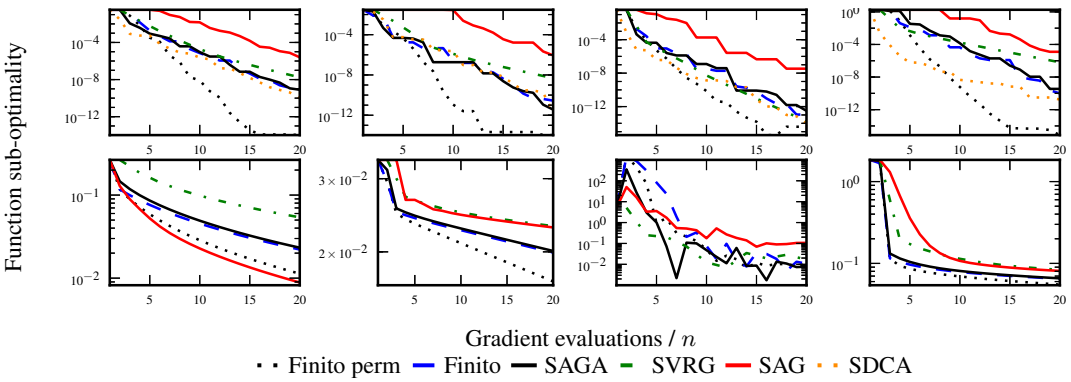

Figure 2: From left to right we have the MNIST, COVTYPE, IJCNN1 and MILLIONSONG datasets. Top row is the L2 regularised case, bottom row the L1 regularised case.

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 not strongly convex, then using $\eta = 3L$, we have for $\bar{x}^k = \frac{1}{k}\sum_{t=1}^{k} x^t$ that:*

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

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 manor 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 use the non-expansiveness property at the beginning, we use a result proved for prox-SVRG [9]:

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

Although their quantity Δ is different, they only use the property that $\mathbb{E}[\Delta] = 0$ to prove the above equation. The same argument as in Lemma 3 can be used to bound $\eta\Delta$ by the same quantity, so we get by using $\beta = 1$ that:

$$\begin{aligned}\alpha\mathbb{E}\left\|x^{k+1} - x^*\right\|^2 \quad \leq \quad & \alpha\left\|x - x^*\right\|^2 - \frac{2\alpha}{\eta}\mathbb{E}\left[F(x^{k+1}) - F(x^*)\right] \\ & + \frac{4\alpha}{\eta^2}\mathbb{E}\left\|f_j'(\phi_j) - f_j'(x^*)\right\|^2 + \frac{4\alpha}{\eta^2}\mathbb{E}\left\|f_j'(x) - f_j'(x^*)\right\|^2.\end{aligned}$$

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 $\eta = 3L$, $c = \frac{3L}{2n}$ and $\alpha = \frac{L}{2n}$, Then we are left with the following after removing other non-positive terms:

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

Negating and summing for k from 1 to k gives:

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

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 divide by k:

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

□

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.