A Simple Practical Accelerated Method for Finite Sums

Aaron Defazio

February 9, 2016

Abstract

We describe a novel optimization method for finite sums (such as empirical risk minimization problems) building on the recently introduced SAGA method. Our method achieves an accelerated convergence rate on strongly convex smooth problems, matching the conjectured optimal rate. Our method has only one parameter (a step size), and is radically simpler than other accelerated methods for finite sums.

Introduction

A large body of recent developments in optimization have focused on minimization of convex finite sums of the form:

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

a very general class of problems including the empirical risk minimization (ERM) framework as a special case. Any function h can be written in this form by setting $f_1(x) = h(x)$ and $f_i = 0$ for $i \neq 1$, however when each f_i is sufficiently regular in a way that can be made precise, it is possible to optimize such sums more efficiently than by treating them as black box functions.

In most cases recently developed methods such as SAG [Schmidt et al., 2013] can find an ε-minimum faster than either stochastic gradient descent or accelerated black-box approaches, both in theory and in practice. We call this class of methods fast incremental gradient methods (FIG).

FIG methods are randomized methods similar to SGD, however unlike SGD they are able to achieve linear convergence rates under Lipschitz-smooth and strong convexity conditions [Mairal, 2014, Defazio et al., 2014b, Johnson and Zhang, 2013, Konečný and Richtárik, 2013]. The geometric term in the first wave of FIG methods had rates depending on the condition number L/μ of the problem, whereas recently several methods have been developed that depend on the square-root of the condition number [Lan and Zhou, 2015, Lin et al., 2015, Shalev-Shwartz and Zhang, 2013b, Nitanda, 2014]. In analogy to the black-box case, these methods are known as accelerated methods.

Algorithm 1

Pick some starting point x^0 and step size γ. Initialize each $g_i^0 = f_i'(x^0)$, where $f_i'(x^0)$ is any gradient/subgradient at x^0. Then at step $k+1$:

1. Pick index j from 1 to n uniformly at random.
2. Update x:

$$z_j^k = x^k + \gamma\left[g_j^k - \frac{1}{n}\sum_{i=1}^{n} g_i^k\right],$$

$$x^{k+1} = \text{prox}_j^{\gamma}\left(z_j^k\right).$$

3. Update the gradient table: Set $g_j^{k+1} = \frac{1}{\gamma}\left(z_j^k - x^{k+1}\right)$, and leave the rest of the entries unchanged ($g_i^{k+1} = g_i^k$ for $i \neq j$).

In this work we develop another accelerated method, which is significantly simpler and requires less tuning than existing accelerated methods. The method we give is a primal approach, however it makes use of the proximal operator oracle for each f_i, instead of a gradient oracle, unlike other primal approaches. The proximal operator is also used by dual methods such as some variants of SDCA [Shalev-Shwartz and Zhang, 2013a].

1 Algorithm

Our algorithm's main step makes use of the proximal operator for a randomly chosenf_i. For convenience, we define:

$$\text{prox}_i^{\gamma}(x) = \text{argmin}_y \left\{\gamma f_i(y) + \frac{1}{2}\|x-y\|^2\right\}.$$

This proximal operator can be computed efficiently or in closed form in many cases, see Section 4 for details. Like SAGA, we also maintain a table of gradients g_i, one for each function f_i. We denote the state of g_i at the end of step k by g_i^k. The iterate (our guess at the solution) at the end of step k is denoted x^k. The starting iterate x^0 may be chosen arbitrarily.

The full algorithm is given as Algorithm 1. The sum of gradients $\frac{1}{n}\sum_{i=1}^{n} g_i^k$ can be cached and updated efficiently at each step, and in most cases instead of storing a full vector for each g_i, only a single real value needs to be stored. This is the case for linear regression or binary classification with logistic loss or hinge loss. A full discussion of implementation is given in Section 4.

With step size $\gamma = \frac{\sqrt{4L+\mu(n-2+n^{-1})}-\sqrt{\mu(n+2+n^{-1})}}{2L\sqrt{\mu n}}$, the expected convergence rate in terms of squared distance to the solution is given by:

$$E\left\|x^k - x^*\right\|^2 \leq \left(1 - \frac{\mu\gamma}{1+\mu\gamma}\right)^k \frac{\mu+L}{\mu}\left\|x^0 - x^*\right\|^2,$$

when each $f_i : \mathbb{R}^d \to \mathbb{R}$ is L-smooth and μ-strongly convex. Using a smaller but simpler step size $\gamma = \frac{1}{2\sqrt{L\mu n}} - \frac{1}{4L}$ gives an easier to interpret geometric constant of

$$1 - \frac{\sqrt{\mu}}{2\sqrt{2Ln} + 2\sqrt{\mu}},$$

in the restricted case of $n \leq L/4\mu$. This slower rate can also be written in terms of the expected number of steps to reach a distance ε from the solution:

$$k = \tilde{O}\left(\frac{\sqrt{nL}}{\sqrt{\mu}}\log(1/\varepsilon)\right),$$

where $\tilde{O}$ hides terms that don't involve ε. This rate matches the fastest known methods for this problem. Unlike other accelerated approaches though, we have only a single tunable parameter (the step size γ), and the algorithm doesn't need knowledge of L or μ except for their appearance in the step size.

Compared to the $\tilde{O}((L/\mu)\log(1/\varepsilon))$ rate for SAGA and other non-accelerated FIG methods, accelerated FIG methods are significantly faster when n is small compared to L/μ, however for $n \geq L/\mu$ the performance is essentially the same. All known FIG methods hit a kind of wall at $n \approx L/\mu$, where they decrease the error at each step by no more than $1 - \frac{1}{n}$. Indeed, when $n \geq L/\mu$ the problem is so well conditioned so as to be easy for any FIG method to solve it efficiently. This is sometimes called the big data setting [Defazio et al., 2014b].

Our convergence rate can also be compared to that of optimal first-order black box methods, which have rates of the form $k = O\left(\left(\sqrt{L/\mu}\right)\log(1/\varepsilon)\right)$ per epoch equivalent. We are able to achieve a $\sqrt{n}$ speedup on a per epoch basis. Of course, all of the mentioned rates are significantly better then the $O((L/\mu)\log(1/\varepsilon))$ rate of gradient descent.

For non-smooth but strongly convex problems, we give a convergence rate with a $1/\varepsilon$-type rate under a standard iterate averaging scheme. This rate does not require the use of a decreasing step size scheme, which allows the algorithm to require less tuning than other primal approaches. This is similar to the rates obtained by dual methods such as SDCA.

2 Relation to other approaches

Our method is most closely related to the SAGA method. To make the relation clear, we may write our method's main step as:

$$x^{k+1} = x^k - \gamma\left[f_j'(x^{k+1}) + g_j^k + \frac{1}{n}\sum_{i=1}^{n} g_i^k\right],$$

whereas SAGA has a step of the form:

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

The difference is the point at which the gradient of f_j is evaluated at. The proximal operator has the effect of evaluating the gradient at x^{k+1} instead of x^k. While a small difference on the surface, this change has profound effects. It allows the method to be applied directly to non-smooth problems using fixed step sizes, a property not shared by SAGA or other primal FIG methods. Additionally, it allows for much larger step sizes to be used, which is why the method is able to achieve an accelerated rate.

It is also illustrative to look at how the methods behave at $n = 1$. SAGA degenerates into regular gradient descent, whereas our method becomes the proximal-point method [Rockafellar, 1976]:

$$x^{k+1} = \text{prox}_{\gamma f}(x^k).$$

The proximal point method has quite remarkable properties. For strongly convex problems, it converges *for any* $\gamma \geq 0$ at a linear rate for fixed γ. The convergence may be made as fast as you like by using a sufficiently large γ, the downside being the inherent difficulty of evaluating the proximal operator. Evaluating the proximal operator is normally easier then minimizing f, especially if it is solved to only a low accuracy, however the need to solve it repeatedly results in the method rarely being practical.

In our case we only need to solve the proximal operator for an individual term f_i at each step. This is practical for a lot of problems, but it does still limit the scope of applicability. For example, multi-class softmax classifiers typically have difficult proximal operators.

3 Theory

3.1 Proximal operator bounds

In this section we rehash some simple bounds from proximal operator theory that we will use in this work. Define the shorthand $p_{\gamma f}(x) = \text{prox}_{\gamma f}(x)$, and let $g_{\gamma f}(x) = \frac{1}{\gamma}\left(x - p_{\gamma f}(x)\right)$, so that $p_{\gamma f}(x) = x - \gamma g_{\gamma f}(x)$. Note that $g_{\gamma f}(x)$ is a subgradient of f at the point $p_{\gamma f}(x)$. This relation is known as the optimality condition of the proximal operator.

We will also use a few standard convexity bounds without proof. Let $f:\mathbb{R}^d\to\mathbb{R}$ be a convex function with strong convexity constant $\mu\geq 0$ and Lipschitz smoothness constant L. Let x^* be the minimizer of f, then for any $x,y\in\mathbb{R}^d$:

$$\left\langle f'(x)-f'(y),x-y\right\rangle \geq \mu\left\|x-y\right\|^2, \tag{1}$$

$$\left\|f'(x)-f'(y)\right\|^2 \leq L^2\left\|x-y\right\|^2. \tag{2}$$

Theorem 1. *(Firm non-expansiveness) For any $x,y\in\mathbb{R}^d$, and any convex function $f:\mathbb{R}^d\to\mathbb{R}$ with strong convexity constant $\mu\geq 0$,*

$$\left\langle x-y,p_{\gamma f}(x)-p_{\gamma f}(y)\right\rangle \geq (1+\mu\gamma)\left\|p_{\gamma f}(x)-p_{\gamma f}(y)\right\|^2.$$

Proof. Using strong convexity of f, we apply Equation 1 at the (sub-)gradients $g_{\gamma f}(x)$ and $g_{\gamma f}(y)$, and their corresponding points $p_{\gamma f}(x)$ and $p_{\gamma f}(y)$:

$$\left\langle g_{\gamma f}(x)-g_{\gamma f}(y),p_{\gamma f}(x)-p_{\gamma f}(y)\right\rangle \geq \mu\left\|p_{\gamma f}(x)-p_{\gamma f}(y)\right\|^2.$$

We now multiply both sides by γ, then add $\left\|p_{\gamma f}(x)-p_{\gamma f}(y)\right\|^2$ to both sides:

$$\left\langle p_{\gamma f}(x)+\gamma g_{\gamma f}(x)-p_{\gamma f}(y)-\gamma g_{\gamma f}(y),p_{\gamma f}(x)-p_{\gamma f}(y)\right\rangle$$
$$\geq (1+\mu\gamma)\left\|p_{\gamma f}(x)-\mathrm{p}_{\gamma f}(y)\right\|^2,$$

leading to the bound by using the optimality condition: $p_{\gamma f}(x)+\gamma g_{\gamma f}(x)=x$. □

Theorem 2. *(Moreau decomposition) For any $x\in\mathbb{R}^d$, and any convex function $f:\mathbb{R}^d\to\mathbb{R}$ with Fenchel conjugate f^* :*

$$p_{\gamma f}(x)=x-\gamma p_{\frac{1}{\gamma}f^*}(x/\gamma). \tag{3}$$

Recall our definition of $g_{\gamma f}(x)=\frac{1}{\gamma}\left(x-p_{\gamma f}(x)\right)$ also. After combining, the following relation thus holds between the proximal operator of the conjugate f^* and $g_{\gamma f}$:

$$p_{\frac{1}{\gamma}f^*}(x/\gamma)=\frac{1}{\gamma}\left(x-p_{\gamma f}(x)\right)=g_{\gamma f}(x). \tag{4}$$

Proof. Let $u=p_{\gamma f}(x)$, and $v=\frac{1}{\gamma}(x-u)$. Then $v\in\partial f(u)$ by the optimality condition of the proximal operator of f (namely if $u=p_{\gamma f}(x)$ then $u=x-\gamma v\Leftrightarrow v\in\partial f(u)$). It follows by conjugacy of f that $u\in\partial f^*(v)$. Thus we may interpret $v=\frac{1}{\gamma}(x-u)$ as the optimality condition of a proximal operator of f^* :

$$v=p_{\frac{1}{\gamma}f^*}(\frac{1}{\gamma}x).$$

Plugging in the definition of v then gives:

$$\frac{1}{\gamma}(x-u)=p_{\frac{1}{\gamma}f^*}(\frac{1}{\gamma}x).$$

Further plugging in $u=p_{\gamma f}(x)$ and rearranging gives the result. □

Theorem 3. *For any $x,y\in\mathbb{R}^d$, and any convex L-smooth function $f:\mathbb{R}^d\to\mathbb{R}$:*

$$\left\langle g_{\gamma f}(x)-g_{\gamma f}(y),x-y\right\rangle \geq \gamma\left(1+\frac{1}{L\gamma}\right)\left\|g_{\gamma f}(x)-g_{\gamma f}(y)\right\|^2,$$

(Note $g_{\gamma f}(x)=\frac{1}{\gamma}\left(x-p_{\gamma f}(x)\right)$ defined above).

Proof. We will apply firm-nonexpansiveness to the proximal operator of f^* as it appears in the decomposition. Note that L-smoothness of f implies $1/L$-strong convexity of f^*. In particular we apply it to the points $\frac{1}{\gamma}x$ and $\frac{1}{\gamma}y$:

$$\left\langle p_{\frac{1}{\gamma}f^*}(\frac{1}{\gamma}x)-p_{\frac{1}{\gamma}f^*}(\frac{1}{\gamma}y),\frac{1}{\gamma}x-\frac{1}{\gamma}y\right\rangle \geq$$
$$\left(1+\frac{1}{L\gamma}\right)\left\|p_{\frac{1}{\gamma}f^*}(\frac{1}{\gamma}x)-p_{\frac{1}{\gamma}f^*}(\frac{1}{\gamma}y)\right\|^2.$$

We now pull $\frac{1}{\gamma}$ from the right side of the inner product out, and plug in Equation 4:

$$\left\langle g_{\gamma f}(x)-g_{\gamma f}(y),x-y\right\rangle \geq \gamma\left(1+\frac{1}{L\gamma}\right)\left\|g_{\gamma f}(x)-g_{\gamma f}(y)\right\|^2.$$

□

3.2 Lemmas

Let x^* be the unique minimizer (due to strong convexity) of f. In addition to the notation used in the description of the algorithm, we also fix a set of subgradients g_j^*, one for each of f_j at x^*, chosen such that $\sum_j^n g_j^*=0$. Then we also define:

$$v_j=x^*+\gamma g_j^*.$$

It follows that when we are at the solution x^*, we have a proximal step for component j of:

$$x^*=\mathrm{prox}_j^\gamma\left(x^*+\gamma g_j^*\right)=\mathrm{prox}_j^\gamma(v_j).$$

Lemma 4. *Under Algorithm 1, taking the expectation over the random choice of j, conditioning on x^k and each g_i^k, allows us to bound the following inner product at step k:*

$$E\left\langle \gamma\left[g_j^k-\frac{1}{n}\sum_{i=1}^n g_i^k\right]-\gamma g_j^*, \right. \tag{5}$$
$$\left.\left(x^k-x^*\right)+\gamma\left[g_j^k-\frac{1}{n}\sum_{i=1}^n g_i^k\right]-\gamma g_j^*\right\rangle \tag{6}$$
$$\leq \gamma^2\frac{1}{n}\sum_{i=1}^n\left\|g_i^k-g_i^*\right\|^2. \tag{7}$$

Notation	Description	Additional relation
x^k	Current iterate at step k	$x^k \in R^d$
x^*	Solution	$x^* \in R^d$
γ	Step size	
$p_{\gamma f}(x)$	Short-hand in results for generic f	$p_{\gamma f}(x) = \text{prox}_{\gamma f}(x)$
$\text{prox}_i^\gamma(x)$	Proximal operator of γf_i at x	$= \text{argmin}_y \left\{ \gamma f_i(y) + \frac{1}{2}\|x-y\|^2 \right\}$
g_i^k	A stored subgradient of f_i as seen at step k	
g_i^*	A subgradient of f_i at x^*	$\sum_{i=1}^n g_i^* = 0$
v_i	$v_i = x^* + \gamma g_i^*$	$x^* = \text{prox}_i^\gamma(v_i)$
j	Chosen component index (random variable)	
z_j^k	$z_j^k = x^k + \gamma \left[g_j^k - \frac{1}{n}\sum_{i=1}^n g_i^k \right]$	$x_j^{k+1} = \text{prox}_j^\gamma\left(z_j^k\right)$

Table 1: Notation quick reference

Proof. We start by splitting on the right hand side of the inner product:

$$= E\left\langle \gamma\left[g_j^k - \frac{1}{n}\sum_{i=1}^n g_i^k\right] - \gamma g_j^*, x^k - x^* \right\rangle$$
$$+E\left\langle \gamma\left[g_j^k - \frac{1}{n}\sum_{i=1}^n g_i^k\right] - \gamma g_j^*, \gamma\left[g_j^k - \frac{1}{n}\sum_{i=1}^n g_i^k\right] - \gamma g_j^* \right\rangle \quad (8)$$

The first inner product has expectation 0 on the left hand side (Recall that $E[g_j^*] = 0$), so it's simply 0 in expectation (we may take expectation on the left since the right doesn't depend on j). The second inner product is the same on both sides, so we may convert it to a norm-squared term. So we have:

$$\begin{aligned} &= \gamma^2 E\left\| g_j^k - \frac{1}{n}\sum_{i=1}^n g_i^k - g_j^* \right\|^2 \\ &\leq \gamma^2 E\left\| g_j^k - g_j^* \right\|^2 \\ &= \gamma^2 \frac{1}{n}\sum_{i=1}^n \left\| g_i^k - g_i^* \right\|^2. \end{aligned}$$

□

3.3 Main result

Theorem 5. *We define the Lyapunov function T^k of our algorithm (Point-SAGA) at step k as:*

$$T^k = \frac{c}{n}\sum_{i=1}^n \left\| g_i^k - g_i^* \right\|^2 + \left\| x^k - x^* \right\|^2,$$

for $c = 1/\mu L$. Then using step size, $\gamma = \frac{\sqrt{4L+\mu(n-2+n^{-1})} - \sqrt{\mu(n+2+n^{-1})}}{2L\sqrt{\mu n}}$, the expectation of T^{k+1}, over the random choice of j, conditioning on x^k and each g_i^k, is:

$$E\left[T^{k+1}\right] \leq (1-\kappa)T^k,$$
$$\text{for } \kappa = \frac{\mu\gamma}{1+\mu\gamma},$$

when each $f_i : \mathbb{R}^d \to \mathbb{R}$ is L-smooth and μ-strongly convex. This is the same Lyapunov function as used by Hofmann et al. [2015].

Proof. Term 1 of T^{k+1} is straight-forward to simplify:

$$\frac{c}{n}E\sum_{i=1}^n \left\| g_i^{k+1} - g_i^* \right\|^2$$
$$= \left(1 - \frac{1}{n}\right)\frac{c}{n}\sum_{i=1}^n \left\| g_i^k - g_i^* \right\|^2 + \frac{c}{n}E\left\| g_j^{k+1} - g_j^* \right\|^2.$$

For term 2 of T^{k+1} we start by applying firm non-expansiveness (Theorem 1):

$$(1+\mu\gamma)E\left\| x^{k+1} - x^* \right\|^2$$

$$\begin{aligned} &= (1+\mu\gamma)E\left\| \text{prox}_j^\gamma(z_j^k) - \text{prox}_j^\gamma(v_j) \right\|^2 \\ &\leq E\left\langle \text{prox}_j^\gamma(z_j^k) - \text{prox}_j^\gamma(v_j), z_j^k - v_j \right\rangle \\ &= E\left\langle x^{k+1} - x^*, z_j^k - v_j \right\rangle. \end{aligned}$$

Now we add and subtract x^k :

$$\begin{aligned} &= E\left\langle x^{k+1} - x^k + x^k - x^*, z_j^k - v_j \right\rangle \\ &= E\left\langle x^k - x^*, z_j^k - v_j \right\rangle + E\left\langle x^{k+1} - x^k, z_j^k - v_j \right\rangle \\ &= \left\| x^k - x^* \right\|^2 + E\left\langle x^{k+1} - x^k, z_j^k - v_j \right\rangle, \end{aligned}$$

where we have pulled out the quadratic term by using $E[z_j^k - v_j] = x^k - x^*$ (we can take the expectation since the left hand side of the inner product doesn't depend on j). We now expand $E\left\langle x^{k+1} - x^k, z_j^k - v_j \right\rangle$ further:

$$E\left\langle x^{k+1} - x^k, z_j^k - v_j \right\rangle$$

$$= E\left\langle x^{k+1} - \gamma g_j^* + \gamma g_j^* - x^k, z_j^k - v_j \right\rangle$$
$$= E\left\langle x^k - \gamma g_j^{k+1} + \gamma\left[g_j^k - \frac{1}{n}\sum_{i=1}^n g_i^k\right] - \gamma g_j^* + \gamma g_j^* - x^k, \right.$$
$$\left. \left(x^k - x^*\right) + \gamma\left[g_j^k - \frac{1}{n}\sum_{i=1}^n g_i^k\right] - \gamma g_j^* \right\rangle. \quad (9)$$

We further split the left side of the inner product to give two separate inner products:

$$= E\left\langle \gamma\left[g_j^k - \frac{1}{n}\sum_{i=1}^n g_i^k\right] - \gamma g_j^*, \right. \quad (10)$$
$$\left. \left(x^k - x^*\right) + \gamma\left[g_j^k - \frac{1}{n}\sum_{i=1}^n g_i^k\right] - \gamma g_j^* \right\rangle \quad (11)$$
$$+ E\left\langle \gamma g_j^* - \gamma g_j^{k+1}, \left(x^k - x^*\right) + \gamma\left[g_j^k - \frac{1}{n}\sum_{i=1}^n g_i^k\right] - \gamma g_j^* \right\rangle. \quad (12)$$

The first inner product in Equation 12 is the quantity we bounded in Lemma 4 by $\gamma^2 \frac{1}{n}\sum_{i=1}^n \left\|g_i^k - g_i^*\right\|^2$. The second inner product in Equation 12, can be simplified using Theorem 3 (note the right side of the inner product is equal to $z_j^k - v_j$):

$$-\gamma E\left\langle g_j^{k+1} - g_j^*, z_j^k - v_j \right\rangle \leq -\gamma^2\left(1 + \frac{1}{L\gamma}\right) E\left\|g_j^{k+1} - g_j^*\right\|^2.$$

Combing these gives the following bound on $(1 + \mu\gamma)E\left\|x^{k+1} - x^*\right\|^2$:

$$(1+\mu\gamma)E\left\|x^{k+1} - x^*\right\|^2$$
$$\leq \left\|x^k - x^*\right\|^2 + \gamma^2\frac{1}{n}\sum_{i=1}^n \left\|g_i^k - g_i^*\right\|^2$$
$$- \gamma^2\left(1 + \frac{1}{L\gamma}\right) E\left\|g_j^{k+1} - g_j^*\right\|^2.$$

Define $\alpha = \frac{1}{1+\mu\gamma} = 1 - \kappa$, where $\kappa = \frac{\mu\gamma}{1+\mu\gamma}$. Now we multiply the above inequality through by α and combine with the rest of the Lyapunov function, giving:

$$E\left[T^{k+1}\right] \leq T^k + \left(\alpha\gamma^2 - \frac{c}{n}\right)\frac{1}{n}\sum_i^n \left\|g_i^k - g_i^*\right\|^2$$
$$+ \left(\frac{c}{n} - \alpha\gamma^2 - \frac{\alpha\gamma}{L}\right) E\left\|g_j^{k+1} - g_j^*\right\|^2$$
$$- \kappa E\left\|x^k - x^*\right\|^2.$$

We want an α convergence rate, so we pull out the required terms:

$$E\left[T^{k+1}\right] \leq \alpha T^k + \left(\alpha\gamma^2 + \kappa c - \frac{c}{n}\right)\frac{1}{n}\sum_i^n \left\|g_i^k - g_i^*\right\|^2$$
$$+ \left(\frac{c}{n} - \alpha\gamma^2 - \frac{\alpha\gamma}{L}\right) E\left\|g_j^{k+1} - g_j^*\right\|^2.$$

Now to complete the proof we note that $c = 1/\mu L$ and $\gamma = \frac{\sqrt{4L+\mu(n-2+n^{-1})} - \sqrt{\mu(n+2+n^{-1})}}{2L\sqrt{\mu n}}$ ensure that both terms inside the round brackets are non-positive, giving $ET^{k+1} \leq \alpha T^k$. These constants were found by equating the equations in the brackets to zero, and solving with respect to the two unknowns, γ and c.

□

Corollary 6. *Chaining Theorem 5 gives a convergence rate for point-saga at step k under the constants given in Theorem 5 of:*

$$E\left\|x^k - x^*\right\|^2 \leq (1-\kappa)^k \frac{\mu + L}{\mu}\left\|x^0 - x^*\right\|^2,$$

if each $f_i : \mathbb{R}^d \to \mathbb{R}$ is L-smooth and μ-strongly convex.

Proof. First we simplify T^0 using $c = 1/\mu L$ and use Lipschitz smoothness (Equation 2):

$$\begin{aligned} T^0 &= \frac{1}{\mu L}\cdot\frac{1}{n}\sum_i \left\|g_i^0 - g_i^*\right\|^2 + \left\|x^0 - x^*\right\|^2 \\ &\leq \frac{L}{\mu}\cdot\left\|x^0 - x^*\right\|^2 + \left\|x^0 - x^*\right\|^2 \\ &= \frac{\mu + L}{\mu}\left\|x^0 - x^*\right\|^2. \end{aligned}$$

Now recall that Theorem 5 gives a bound $E\left[T^{k+1}\right] \leq (1-\kappa)T^k$ where the expectation is conditional on x^k and each g_i^k from step k, taking expectation over the randomness in the choice of j. We can further take expectation with respect to x^k and each g_i^k, giving the unconditional bound:

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

Chaining over k gives the result. □

Theorem 7. *Suppose each $f_i : \mathbb{R}^d \to \mathbb{R}$ is μ-strongly convex, $\left\|g_i^0 - g_i^*\right\| \leq B$ and $\left\|x^0 - x^*\right\| \leq R$. Then after k iterations of Point-SAGA with step size $\gamma = R/B\sqrt{n}$:*

$$E\left\|\bar{x}^k - x^*\right\|^2 \leq 2\frac{\sqrt{n}\left(1+\mu\left(R/B\sqrt{n}\right)\right)}{\mu k}RB,$$

where $\bar{x}^k = \frac{1}{k}E\sum_{t=1}^k x^t$.

Proof. Recall the bound on the Lyapunov function established in Theorem 5:

$$\begin{aligned}E\left[T^{k+1}\right] \leq T^k &+ \left(\alpha\gamma^2 - \frac{c}{n}\right)\frac{1}{n}\sum_i^n\left\|g_i^k - g_i^*\right\|^2 \\ &+ \left(\frac{c}{n} - \alpha\gamma^2 - \frac{\alpha\gamma}{L}\right)E\left\|g_j^{k+1} - g_j^*\right\|^2 \\ &- \kappa E\left\|x^k - x^*\right\|^2.\end{aligned}$$

In the non-smooth case this holds with $L = \infty$. In particular, if we take $c = \alpha\gamma^2 n$, then:

$$-\kappa E\left\|x^{k+1} - x^*\right\|^2 \geq E\left[T^{k+1}\right] - T^k.$$

Recall that this expectation is (implicitly) conditional on x^k and each g_i^k from step k, Taking expectation over the randomness in the choice of j. We can further take expectation with respect to x^k and each g_i^k, and negate the inequality, giving the unconditional bound:

$$\kappa E\left\|x^{k+1} - x^*\right\|^2 \leq E\left[T^k\right] - E\left[T^{k+1}\right].$$

We now sum this over $t = 0 \ldots k$:

$$\kappa E\sum_{t=1}^k\left\|x^t - x^*\right\|^2 \leq T^0 - E\left[T^k\right].$$

We can drop the $-E\left[T^k\right]$ since it is always negative. Dividing through by k:

$$\frac{1}{k}E\sum_{t=1}^k\left\|x^t - x^*\right\|^2 \leq \frac{1}{\kappa k}T^0.$$

Now using Jensen's inequality on the right gives:

$$E\left\|\bar{x}^k - x^*\right\|^2 \leq \frac{1}{\kappa k}T^0,$$

where $\bar{x}^k = \frac{1}{k}E\sum_{t=1}^k x^t$. Now we plug in $T^0 = \frac{c}{n}\sum_i\left\|g_i^0 - g_i^*\right\|^2 + \left\|x^0 - x^*\right\|^2$ with $c = \alpha\gamma^2 n \leq \gamma^2 n$:

$$E\left\|\bar{x}^k - x^*\right\|^2 \leq \frac{\gamma^2 n}{\kappa k}\frac{1}{n}\sum_i\left\|g_i^0 - g_i^*\right\|^2 + \frac{1}{\kappa k}\left\|x^0 - x^*\right\|^2.$$

Now we plug in the bounds in terms of B and R:

$$E\left\|\bar{x}^k - x^*\right\|^2 \leq \frac{\gamma^2 n}{\kappa k}B^2 + \frac{1}{\kappa k}R^2.$$

In order to balance the terms on the right, we need:

$$\begin{aligned}\frac{\gamma^2 n}{\kappa k}B^2 &= \frac{1}{\kappa k}R^2, \\ \therefore \gamma^2 n B^2 &= R^2, \\ \therefore \gamma^2 &= \frac{R^2}{nB^2}.\end{aligned}$$

So we can take $\gamma = R/B\sqrt{n}$, giving a rate of:

$$\begin{aligned}E\left\|\bar{x}^k - x^*\right\|^2 &\leq \frac{2}{\kappa k}R^2 \\ &= 2\frac{1+\mu\gamma}{\mu\gamma k}R^2 \\ &= 2\frac{\sqrt{n}\left(1+\mu\left(R/B\sqrt{n}\right)\right)}{\mu k}RB.\end{aligned}$$

□

4 Implementation

Care must be taken for efficient implementation, particularly in the sparse gradient case. We discuss the key points below. A fast Cython implementation is available on the authors websites incorporating these techniques.

Proximal operators

For the most common binary classification and regression methods, implementing the proximal operator is straightforward. In this section let y_j be the label or target for regression, and X_j the data instance vector. We assume for binary classification that $y_j \in \{-1, 1\}$.

Hinge loss:

$$f_j(z) = l(z; y_j, X_j) = \max\left\{0,\, 1 - y_j\left\langle z, X_j\right\rangle\right\}.$$

The proximal operator has a closed form expression:

$$\text{prox}_{\gamma f_j}(z) = z - \gamma y_j \nu X_j,$$

where:

$$s = \frac{1 - y_j\left\langle z, X_j\right\rangle}{\gamma\left\|X_j\right\|^2}.$$

$$\nu = \begin{cases}-1 & s \geq 1 \\ 0 & s \leq 0 \\ -s & \text{otherwise}\end{cases}.$$

Logistic loss:

$$f_j(z) = l(z; y_j, X_j) = \log\left(1 + \exp\left(-y_j X_j^T z\right)\right).$$

There is no closed form expression, however it can be computed very efficiently using Newton iteration, since it can be reduced to a 1D minimization problem. In particular, let $c_0 = 0$, $\gamma' = \gamma \left\|X_j\right\|^2$, and $a = \left\langle z, X_j \right\rangle$. Then iterate until convergence:

$$s^k = \frac{-y_j}{1 + \exp\left(y_j c^k\right)},$$

$$c^{k+1} = c^k - \frac{\gamma' s^k + c^k - a}{1 - y' s^k - \gamma' s^k s^k}.$$

The prox operator is then $\text{prox}_{\gamma f_j}(z) = z - \left(a - c^k\right) X_j / \left\|X_j\right\|^2$. Three iterations are generally enough, but ill-conditioned problems or large step sizes may require up to 12. Correct initialization is important, as it will diverge when initialized with a point on the opposite side of 0 from the solution.

Squared loss:

$$f_j(z) = l(z; y_j, X_j) = \frac{1}{2}\left(X_j^T z - y_j\right)^2.$$

Let $\gamma' = \gamma \left\|X_j\right\|^2$ and $a = \left\langle z, X_j \right\rangle$. Define:

$$c = \frac{a + \gamma' y}{1 + \gamma'}.$$

Then $\text{prox}_{\gamma f_j}(z) = z - (a - c) X_j / \left\|X_j\right\|^2$.

L2 regularization

Including a regularizer within each f_i, i.e. $F_i(x) = f_i(x) + \frac{\mu}{2}\|x\|^2$, can be done using the proximal operator of f_i. Define the scaling factor:

$$\rho = 1 - \frac{\mu\gamma}{1 + \mu\gamma}.$$

Then $\text{prox}_{\gamma F_i}(z) = \text{prox}_{\rho\gamma f_i}(\rho z)$.

Initialization

Instead of setting $g_i^0 = f_i'(x^0)$ before commencing the algorithm, we recommend using $g_i^0 = 0$ instead. This avoids the cost of a initial pass over the data. Note that the theory does support this initialization with a small modification. In practical effect this is similar to the SDCA initialization of each dual variable to 0.

5 Experiments

We tested our algorithm which we call Point-SAGA against SAGA [Defazio et al., 2014a], SDCA [Shalev-Shwartz and Zhang, 2013a], Pegasos/SGD [Shalev-Shwartz et al., 2011] and the catalyst acceleration scheme [Lin et al., 2015]. SDCA was chosen as the inner algorithm for the catalyst scheme as it doesn't require a step-size, making it the most practical of the variants. A single epoch was used for each SDCA invocation. Accelerated MISO as well as the primal-dual FIG method [Lan and Zhou, 2015] were excluded due to their non-sparse updates. The step-size parameter for each method (κ for catalyst-SDCA) was chosen using a grid search of powers of 2.

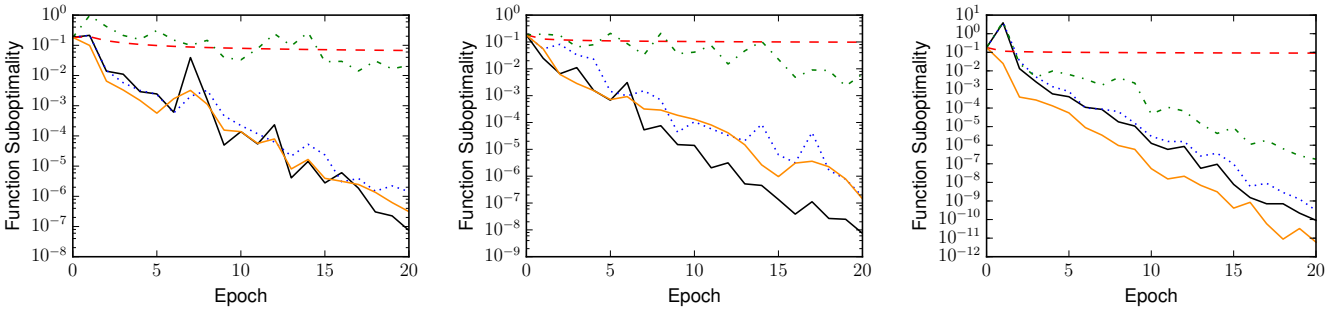
We selected a set of commonly used datasets from the LIBSVM repository [Chang and Lin, 2011]. Pre-scaled versions were used when available. The L_2 regularization constant was set for each problem to ensure f was not in the big data regime $n \geq L/\mu$. The full suite of experiments are reproducible using the code on the authors websites.

Algorithm scaling with respect to n The key property that distinguishes accelerated FIG methods from their non-accelerated counterparts is their performance scale with respect to the dataset size. For large datasets on well-conditioned problems we expect from the theory to see little difference between the methods. To this end, we ran experiments where we subsetted randomly without replacement in 10%, 5%, increments, in order to show the scaling empirically. The amount of regularization was left constant over the different subsets.

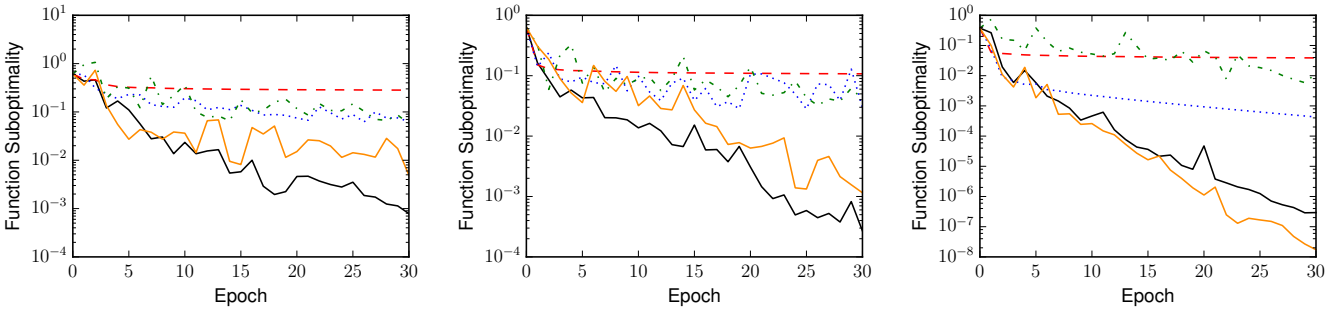
Figure 1 shows the function value sub-optimality for each dataset-subset combination. We see that in general accelerated methods dominate the performance of their non-accelerated counter-parts. Both SDCA and SAGA are much slower on some datasets comparatively than others. For example, SDCA is very slow on the 5 and 10% COVTYPE datasets, whereas both SAGA and SDCA are much slower than the accelerated methods on the AUSTRALIAN dataset. These differences reflect known properties of the two methods. SAGA is able to adapt to inherent strong convexity while SDCA can be faster on very well-conditioned problems.

There is no clear winner between the two accelerated methods, each gives excellent results on each problem. The Pegasos (stochastic gradient descent variant)with its slower than linear rate is a clear loser on each problem, almost appearing as a horizontal line on the log scale of these plots.

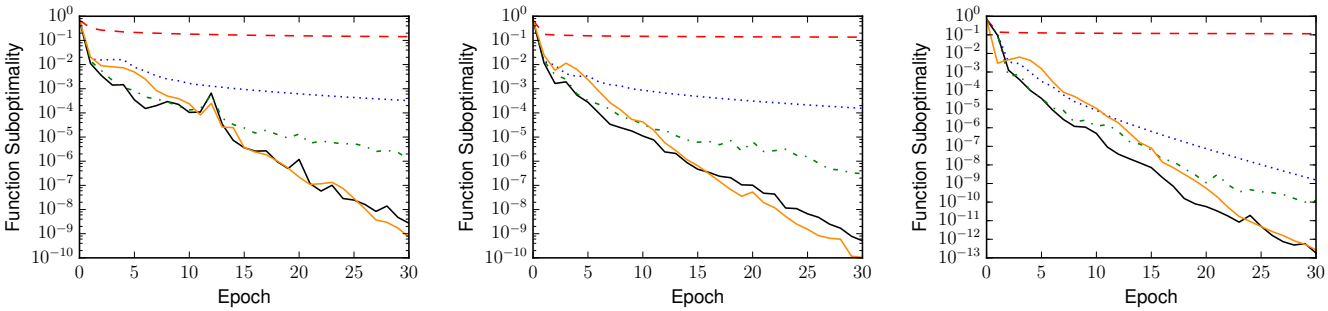
Non-smooth problems We also tested the RCV1 dataset on the hinge loss. In general we don't expect a accelerated rate for this problem, and indeed we observe that Point-SAGA is roughly as fast as SDCA across the different dataset sizes.



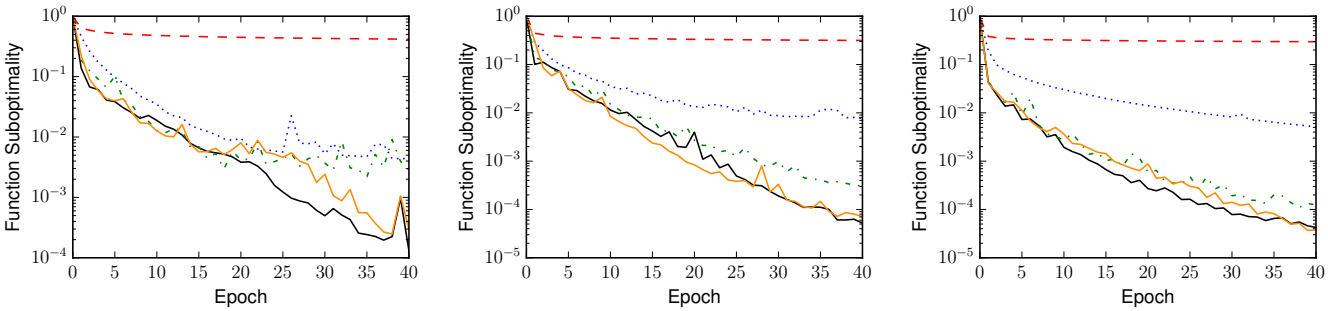
(a) COVTYPE $\mu = 2 \times 10^{-6}$: 5%, 10%, 100%



(b) AUSTRALIAN $\mu = 10^{-4}$: 5%, 10%, 100%



(c) MUSHROOMS $\mu = 10^{-4}$: 5%, 10%, 100%



(d) RCV1 $\mu = 5 \times 10^{-5}$: 5%, 10%, 100% (hinge loss)

Figure 1: Experimental results

References

Chih-Chung Chang and Chih-Jen Lin. Libsvm : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, 2014a.

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*, 2014b.

Thomas Hofmann, Aurelien Lucchi, Simon Lacoste-Julien, and Brian McWilliams. Variance reduced stochastic gradient descent with neighbors. In C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2296–2304. Curran Associates, Inc., 2015. URL `http://papers.nips.cc/paper/5919-variance-reduced-stochastic-gradient-descent-with-neighbors.pdf`.

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

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

G. Lan and Y. Zhou. An optimal randomized incremental gradient method. *ArXiv e-prints*, July 2015.

Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3366–3374. Curran Associates, Inc., 2015. URL `http://papers.nips.cc/paper/5928-a-universal-catalyst-for-first-order-optimization.pdf`.

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

Atsushi Nitanda. Stochastic proximal gradient descent with acceleration techniques. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1574–1582. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5610-stochastic-proximal-gradient-descent-with-acceleration-techniques.pdf`.

R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

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

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

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, 2013b.

Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.