A Stochastic Gradient Method with an Exponential Convergence Rate for Finite Training Sets

Nicolas Le Roux
SIERRA Project-Team
INRIA - ENS
Paris, France
nicolas@le-roux.name

Mark Schmidt
SIERRA Project-Team
INRIA - ENS
Paris, France
mark.schmidt@inria.fr

Francis Bach
SIERRA Project-Team
INRIA - ENS
Paris, France
francis.bach@ens.fr

Abstract

We propose a new stochastic gradient method for optimizing the sum of a finite set of smooth functions, where the sum is strongly convex. While standard stochastic gradient methods converge at sublinear rates for this problem, the proposed method incorporates a memory of previous gradient values in order to achieve a linear convergence rate. In a machine learning context, numerical experiments indicate that the new algorithm can dramatically outperform standard algorithms, both in terms of optimizing the training error and reducing the test error quickly.

1 Introduction

A plethora of the problems arising in machine learning involve computing an approximate minimizer of the sum of a loss function over a large number of training examples, where there is a large amount of redundancy between examples. The most wildly successful class of algorithms for taking advantage of this type of problem structure are *stochastic gradient* (SG) methods [1, 2]. Although the theory behind SG methods allows them to be applied more generally, in the context of machine learning SG methods are typically used to solve the problem of optimizing a sample average over a finite training set, i.e.,

$$\underset{x\in\mathbb{R}^p}{\text{minimize}} \quad g(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x). \tag{1}$$

In this work, we focus on such *finite training data* problems where each f_i is *smooth* and the average function g is *strongly-convex*.

As an example, in the case of ℓ_2-regularized logistic regression we have $f_i(x) := \frac{\lambda}{2}\|x\|^2 + \log(1 + \exp(-b_i a_i^T x))$, where $a_i \in \mathbb{R}^p$ and $b_i \in \{-1, 1\}$ are the training examples associated with a binary classification problem and λ is a regularization parameter. More generally, any ℓ_2-regularized empirical risk minimization problem of the form

$$\underset{x\in\mathbb{R}^p}{\text{minimize}} \quad \frac{\lambda}{2}\|x\|^2 + \frac{1}{n}\sum_{i=1}^{n} l_i(x), \tag{2}$$

falls in the framework of (1) provided that the loss functions l_i are convex and smooth. An extensive list of convex loss functions used in machine learning is given by [3], and we can even include non-smooth loss functions (or regularizers) by using smooth approximations.

The standard *full gradient* (FG) method, which dates back to [4], uses iterations of the form

$$x^{k+1} = x^k - \alpha_k g'(x^k) = x^k - \frac{\alpha_k}{n}\sum_{i=1}^{n} f_i'(x^k). \tag{3}$$

Using x^* to denote the unique minimizer of g, the FG method with a constant step size achieves a *linear* convergence rate:

$$g(x^k) - g(x^*) = O(\rho^k),$$

for some $\rho < 1$ which depends on the condition number of g [5, Theorem 2.1.15]. Linear convergence is also known as *geometric* or *exponential* convergence, because the cost is cut by a fixed fraction on each iteration. Despite the fast convergence rate of the FG method, it can be unappealing when n is large because its iteration cost scales linearly in n. SG methods, on the other hand, have an iteration cost which is *independent* of n, making them suited for that setting. The basic SG method for optimizing (1) uses iterations of the form

$$x^{k+1} = x^k - \alpha_k f'_{i_k}(x^k), \tag{4}$$

where α_k is a step-size and a training example i_k is selected uniformly among the set $\{1, \dots, n\}$. The randomly chosen gradient $f'_{i_k}(x^k)$ yields an unbiased estimate of the true gradient $g'(x^k)$, and one can show under standard assumptions that, for a suitably chosen decreasing step-size sequence $\{\alpha_k\}$, the SG iterations achieve the sublinear convergence rate

$$\mathbb{E}[g(x^k)] - g(x^*) = O(1/k),$$

where the expectation is taken with respect to the selection of the i_k variables. Under certain assumptions this convergence rate is *optimal* for strongly-convex optimization in a model of computation where the algorithm only accesses the function through unbiased measurements of its objective and gradient (see [6, 7, 8]). Thus, we cannot hope to obtain a better convergence rate if the algorithm only relies on unbiased gradient measurements. Nevertheless, by using the stronger assumption that the functions are sampled from a finite dataset, in this paper we show that we can achieve an exponential converengence rate while preserving the iteration cost of SG methods.

The primay contribution of this work is the analysis of a new algorithm that we call the *stochastic average gradient* (SAG) method, a randomized variant of the incremental aggregated gradient (IAG) method of [9], which combines the low iteration cost of SG methods with a linear convergence rate as in FG methods. The SAG method uses iterations of the form

$$x^{k+1} = x^k - \frac{\alpha_k}{n} \sum_{i=1}^{n} y_i^k, \tag{5}$$

where at each iteration a random training example i_k is selected and we set

$$y_i^k = \begin{cases} f'_i(x^k) & \text{if } i = i_k, \\ y_i^{k-1} & \text{otherwise.} \end{cases}$$

That is, like the FG method, the step incorporates a gradient with respect to each training example. But, like the SG method, each iteration only computes the gradient with respect to a single training example and the cost of the iterations is independent of n. Despite the low cost of the SAG iterations, in this paper we show that *the SAG iterations have a linear convergence rate*, like the FG method. That is, by having access to i_k and by keeping a *memory* of the most recent gradient value computed for each training example i, this iteration achieves a faster convergence rate than is possible for standard SG methods. Further, in terms of effective passes through the data, we also show that for certain problems the convergence rate of SAG is faster than is possible for standard FG method.

In a machine learning context where $g(x)$ is a *training cost* associated with a predictor parameterized by x, we are often ultimately interested in the *testing cost*, the expected loss on unseen data points. Note that a linear convergence rate for the training cost does not translate into a similar rate for the testing cost, and an appealing propertly of SG methods is that they achieve the optimal $O(1/k)$ rate for the *testing cost* as long as every datapoint is seen *only once*. However, as is common in machine learning, we assume that we are only given a finite training data set and thus that datapoints are revisited multiple times. In this context, the analysis of SG methods only applies to the training cost and, although our analysis also focuses on the training cost, in our experiments the SAG method typically reached the optimal testing cost faster than both FG and SG methods.

The next section reviews closely-related algorithms from the literature, including previous attempts to combine the appealing aspects of FG and SG methods. However, despite 60 years of extensive research on SG methods, most of the applications focusing on finite datasets, we are not aware of any other SG method that achieves a linear convergence rate while preserving the iteration cost of standard SG methods. Section 3 states the (standard) assumptions underlying our analysis and gives the main technical results; we first give a slow linear convergence rate that applies for any problem, and then give a very fast linear convergence rate that applies when n is sufficiently large. Section 4 discusses practical implementation issues, including how to reduce the storage cost from $O(np)$ to $O(n)$ when each f_i only depends on a linear combination of x. Section 5 presents a numerical comparison of an implementation based on SAG to SG and FG methods, indicating that the method may be very useful for problems where we can only afford to do a few passes through a data set.

2 Related Work

There is a large variety of approaches available to accelerate the convergence of SG methods, and a full review of this immense literature would be outside the scope of this work. Below, we comment on the relationships between the new method and several of the most closely-related ideas.

Momentum: SG methods that incorporate a momentum term use iterations of the form

$$x^{k+1} = x^k - \alpha_k f'_{i_k}(x^k) + \beta_k(x^k - x^{k-1}),$$

see [10]. It is common to set all $\beta_k = \beta$ for some constant β, and in this case we can rewrite the SG with momentum method as

$$x^{k+1} = x^k - \textstyle\sum_{j=1}^{k} \alpha_j \beta^{k-j} f'_{i_j}(x^j).$$

We can re-write the SAG updates (5) in a similar form as

$$x^{k+1} = x^k - \textstyle\sum_{j=1}^{k} \alpha_k S(j, i_{1:k}) f'_{i_j}(x^j), \tag{6}$$

where the selection function $S(j, i_{1:k})$ is equal to $1/n$ if j corresponds to the last iteration where $j = i_k$ and is set to 0 otherwise. Thus, momentum uses a *geometric weighting* of previous gradients while the SAG iterations *select and average* the most recent evaluation of each previous gradient. While momentum can lead to improved practical performance, it still requires the use of a decreasing sequence of step sizes and is not known to lead to a faster convergence rate.

Gradient Averaging: Closely related to momentum is using the sample average of all previous gradients,

$$x^{k+1} = x^k - \tfrac{\alpha_k}{k} \textstyle\sum_{j=1}^{k} f'_{i_j}(x_j),$$

which is similar to the SAG iteration in the form (5) but where *all* previous gradients are used. This approach is used in the dual averaging method [11], and while this averaging procedure leads to convergence for a constant step size and can improve the constants in the convergence rate [12], it does not improve on the $O(1/k)$ rate.

Iterate Averaging: Rather than averaging the gradients, some authors use the basic SG iteration but take an average over x^k values. With a suitable choice of step-sizes, this gives the same asymptotic efficiency as Newton-like second-order SG methods and also leads to increased robustness of the convergence rate to the exact sequence of step sizes [13]. Baher's method [14, §1.3.4] combines gradient averaging with online iterate averaging, and also displays appealing asymptotic properties. The epoch SG method uses averaging to obtain the $O(1/k)$ rate even for non-smooth objectives [15]. However, the convergence rates of these averaging methods remain sublinear.

Stochastic versions of FG methods: Various options are available to accelerate the convergence of the FG method for smooth functions, such as the accelerated full gradient (AFG) method of Nesterov [16], as well as classical techniques based on quadratic approximations such as non-linear conjugate gradient, quasi-Newton, and Hessian-free Newton methods. Several authors have analyzed stochastic variants of these algorithms [17, 18, 19, 20, 12]. Under certain conditions these variants are convergent with an $O(1/k)$ rate [18]. Alternately, if we split the convergence rate into a deterministic and stochastic part, these methods can improve the dependency on the deterministic part [19, 12]. However, as with all other methods we have discussed thus far in this section, we are not aware of any existing method of this flavor that improves on the $O(1/k)$ rate.

Constant step size: If the SG iterations are used with a *constant* step size (rather than a decreasing sequence), then the convergence rate of the method can be split into two parts [21, Proposition 2.4], where the first part depends on k and converges linearly to 0 and the second part is independent of k but does not converge to 0. Thus, with a constant step size the SG iterations have a linear convergence rate up to some tolerance, and in general after this point the iterations do not make further progress. Indeed, convergence of the basic SG method with a constant step size has only been shown under extremely strong assumptions about the relationship between the functions f_i [22]. This contrasts with the method we present in this work which converges to the optimal solution using a constant step size *and* does so with a linear rate (without additional assumptions).

Accelerated methods: Accelerated SG methods, which despite their name are not related to the aforementioned AFG method, take advantage of the fast convergence rate of SG methods with a constant step size. In particular, accelerated SG methods use a constant step size by default, and only decrease the step size on iterations where the inner-product between successive gradient estimates

is negative [23, 24]. This leads to convergence of the method and allows it to potentially achieve periods of linear convergence where the step size stays constant. However, the overall convergence rate of the method remains sublinear.

Hybrid Methods: Some authors have proposed variants of the SG method for problems of the form (1) that seek to gradually transform the iterates into the FG method in order to achieve a linear convergence rate. Bertsekas proposes to go through the data cyclically with a specialized weighting that allows the method to achieve a linear convergence rate for strongly-convex quadratic functions [25]. However, the weighting is numerically unstable and the linear convergence rate treats full passes through the data as iterations. A related strategy is to group the f_i functions into 'batches' of increasing size and perform SG iterations on the batches [26]. In both cases, the iterations that achieve the linear rate have a cost that is not independent of n, as opposed to SAG.

Incremental Aggregated Gradient: Finally, Blatt et al. presents the most closely-related algorithm, the IAG method [9]. This method is identical to the SAG iteration (5), but uses a cyclic choice of i_k rather than sampling the i_k values. This distinction has several important consequences. In particular, Blatt et al. are only able to show that the convergence rate is linear for strongly-convex quadratic functions (without deriving an explicit rate), and their analysis treats full passes through the data as iterations. Using a non-trivial extension of their analysis and a proof technique involving bounding the gradients and iterates simultaneously by a Lyapunov potential function, in this work *we give an explicit linear convergence rate for general strongly-convex functions using the SAG iterations that only examine a single training example*. Further, as our analysis and experiments show, when the number of training examples is sufficiently large, the SAG iterations achieve a linear convergence rate under a much larger set of step sizes than the IAG method. This leads to more robustness to the selection of the step size and also, if suitably chosen, leads to a faster convergence rate and improved practical performance. We also emphasize that in our experiments IAG and the basic FG method perform similarly, while SAG performs much better, showing that the simple change (random selection vs. cycling) can dramatically improve optimization performance.

3 Convergence Analysis

In our analysis we assume that each function f_i in (1) is differentiable and that each gradient f_i' is Lipschitz-continuous with constant L, meaning that for all x and y in $\mathbb{R}^p$ we have

$$\|f_i'(x) - f_i'(y)\| \leq L\|x - y\|.$$

This is a fairly weak assumption on the f_i functions, and in cases where the f_i are twice-differentiable it is equivalent to saying that the eigenvalues of the Hessians of each f_i are bounded above by L. In addition, we also assume that the average function $g = \frac{1}{n}\sum_{i=1}^n f_i$ is strongly-convex with constant $\mu > 0$, meaning that the function $x \mapsto g(x) - \frac{\mu}{2}\|x\|^2$ is convex. This is a stronger assumption and is not satisfied by all machine learning models. However, note that in machine learning we are typically free to choose the regularizer, and we can always add an ℓ_2-regularization term as in Eq. (2) to transform any convex problem into a strongly-convex problem (in this case we have $\mu \geq \lambda$). Note that strong-convexity implies that the problem is solvable, meaning that there exists some unique x^* that achieves the optimal function value. Our convergence results assume that we initialize y_i^0 to a zero vector for all i, and our results depend on the variance of the gradient norms at the optimum x^*, denoted by $\sigma^2 = \frac{1}{n}\sum_i \|f_i'(x^*)\|^2$. Finally, all our convergence results consider expectations with respect to the internal randomization of the algorithm, and not with respect to the data (which are assumed to be deterministic and fixed).

We first consider the convergence rate of the method when using a constant step size of $\alpha_k = \frac{1}{2nL}$, which is similar to the step size needed for convergence of the IAG method in practice.

Proposition 1 *With a constant step size of $\alpha_k = \frac{1}{2nL}$, the SAG iterations satisfy for $k \geq 1$:*

$$\mathbb{E}\left[\|x^k - x^*\|^2\right] \leqslant \left(1 - \frac{\mu}{8Ln}\right)^k \left[3\|x_0 - x^*\|^2 + \frac{9\sigma^2}{4L^2}\right].$$

The proof is given in the supplementary material. Note that the SAG iterations also trivially obtain the $O(1/k)$ rate achieved by SG methods, since

$$\left(1 - \frac{\mu}{8Ln}\right)^k \leqslant \exp\left(-\frac{k\mu}{8Ln}\right) \leqslant \frac{8Ln}{k\mu} = O(n/k),$$

albeit with a constant which is proportional to n. Despite this constant, they are advantageous over SG methods in later iterations because they obtain an exponential convergence rate as in FG

methods. We also note that an exponential convergence rate is obtained for any constant step size smaller than $\frac{1}{2nL}$.

In terms of passes through the data, the rate in Proposition 1 is similar to that achieved by the basic FG method. However, our next result shows that, if the number of training examples is slightly larger than L/μ (which will often be the case, as discussed in Section 6), then the SAG iterations can use a larger step size and obtain a better convergence rate that is independent of μ and L (see proof in the supplementary material).

Proposition 2 *If $n \geqslant \frac{8L}{\mu}$, with a step size of $\alpha_k = \frac{1}{2n\mu}$ the SAG iterations satisfy for $k \geqslant n$:*

$$\mathbb{E}\left[g(x^k) - g(x^*)\right] \leqslant C\Big(1 - \frac{1}{8n}\Big)^k,$$

$$\textit{with } C = \left[\frac{16L}{3n}\|x^0 - x^*\|^2 + \frac{4\sigma^2}{3n\mu}\Big(8\log\Big(1 + \frac{\mu n}{4L}\Big) + 1\Big)\right].$$

We state this result for $k \geqslant n$ because we assume that the first n iterations of the algorithm use an SG method and that we initialize the subsequent SAG iterations with the average of the iterates, which leads to an $O((\log n)/k)$ rate. In contrast, using the SAG iterations from the beginning gives the same rate but with a constant proportional to n. Note that this bound is obtained when initializing all y_i to zero after the SG phase.[1] However, in our experiments we do not use the SG initialization but rather use a minor variant of SAG (discussed in the next section), which appears more difficult to analyze but which gives better performance.

It is interesting to compare this convergence rate with the known convergence rates of first-order methods [5, see §2]. For example, if we take $n = 100000$, $L = 100$, and $\mu = 0.01$ then the basic FG method has a rate of $((L-\mu)/(L+\mu))^2 = 0.9996$ and the 'optimal' AFG method has a faster rate of $(1 - \sqrt{\mu/L}) = 0.9900$. In contrast, running n iterations of SAG has a much faster rate of $(1 - 1/8n)^n = 0.8825$ using the same number of evaluations of f_i'. Further, the lower-bound for a black-box first-order method is $((\sqrt{L} - \sqrt{\mu})/(\sqrt{L} + \sqrt{\mu}))^2 = 0.9608$, indicating that SAG can be substantially faster than any FG method that does not use the structure of the problem.[2] In the supplementary material, we compare Propositions 1 and 2 to the rates of primal and dual FG and coordinate-wise methods for the special case of ℓ_2-regularized leasts squares.

Even though n appears in the convergence rate, if we perform n iterations of SAG (i.e., one effective pass through the data), the error is multiplied by $(1 - 1/8n)^n \leq \exp(-1/8)$, which is independent of n. Thus, each pass through the data reduces the excess cost by a constant multiplicative factor that is independent of the problem, as long as $n \geqslant 8L/\mu$. Further, while the step size in Proposition 2 depends on μ and n, we can obtain the same convergence rate by using a step size as large as $\alpha_k = \frac{1}{16L}$. This is because the proposition is true for all values of μ satisfying $\frac{\mu}{L} \geqslant \frac{8}{n}$, so we can choose the smallest possible value of $\mu = \frac{8L}{n}$. We have observed in practice that the IAG method with a step size of $\alpha_k = \frac{1}{2n\mu}$ may diverge, even under these assumptions. Thus, for certain problems the SAG iterations can tolerate a much larger step size, which leads to increased robustness to the selection of the step size. Further, as our analysis and experiments indicate, the ability to use a large step size leads to improved performance of the SAG iterations.

While we have stated Proposition 1 in terms of the iterates and Proposition 2 in terms of the function values, the rates obtained on iterates and function values are equivalent because, by the Lipschitz and strong-convexity assumptions, we have $\frac{\mu}{2}\|x^k - x^*\|^2 \leqslant g(x^k) - g(x^*) \leqslant \frac{L}{2}\|x^k - x^*\|^2$.

4 Implementation Details

In this section we describe modifications that substantially reduce the SAG iteration's memory requirements, as well as modifications that lead to better practical performance.

Structured gradients: For many problems the storage cost of $O(np)$ for the y_i^k vectors is prohibitive, but we can often use structure in the f_i' to reduce this cost. For example, many loss functions f_i take the form $f_i(a_i^T x)$ for a vector a_i. Since a_i is constant, for these problems we only

[1] While it may appear suboptimal to not use the gradients computed during the n iterations of stochastic gradient descent, using them only improves the bound by a constant.

[2] Note that L in the SAG rates is based on the f_i' functions, while in the FG methods it is based on g' which can be much smaller.

need to store the scalar $f'_{i_k}(u_i^k)$ for $u_i^k = a_{i_k}^T x^k$ rather than the full gradient $a_i^T f'_i(u_i^k)$, reducing the storage cost to $O(n)$. Further, because of the simple form of the SAG updates, if a_i is sparse we can use 'lazy updates' in order to reduce the iteration cost from $O(p)$ down to the sparsity level of a_i.

Mini-batches: To employ vectorization and parallelism, practical SG implementations often group training examples into 'mini-batches' and perform SG iterations on the mini-batches. We can also use mini-batches within the SAG iterations, and for problems with dense gradients this decreases the storage requirements of the algorithm since we only need a y_i^k for each mini-batch. Thus, for example, using mini-batches of size 100 leads to a 100-fold reduction in the storage cost.

Step-size re-weighting: On early iterations of the SAG algorithm, when most y_i^k are set to the uninformative zero vector, rather than dividing α_k in (5) by n we found it was more effective to divide by m, the number of unique i_k values that we have sampled so far (which converges to n). This modification appears more difficult to analyze, but with this modification we found that the SAG algorithm outperformed the SG/SAG hybrid algorithm analyzed in Proposition 2.

Exact regularization: For regularized objectives like (2) we can use the exact gradient of the regularizer, rather than approximating it. For example, our experiments on ℓ_2-regularized optimization problems used the recursion

$$d \leftarrow d - y_i, \qquad y_i \leftarrow l'_i(x^k), \qquad d \leftarrow d + y_i, \qquad x \leftarrow \big(1 - \alpha\lambda\big)x - \frac{\alpha}{m}d\,. \tag{7}$$

This can be implemented efficiently for sparse data sets by using the representation $x = \kappa z$, where κ is a scalar and z is a vector, since the update based on the regularizer simply updates κ.

Large step sizes: Proposition 1 requires $\alpha_k \leqslant 1/2Ln$ while under an additional assumption Proposition 2 allows $\alpha_k \leqslant 1/16L$. In practice we observed better performance using step sizes of $\alpha_k = 1/L$ and $\alpha_k = 2/(L + n\mu)$. These step sizes seem to work even when the additional assumption of Proposition 2 is not satisfied, and we conjecture that the convergence rates under these step sizes are much faster than the rate obtained in Proposition 1 for the general case.

Line search: Since L is generally not known, we experimented with a basic line-search, where we start with an initial estimate L_0, and we double this estimate whenever we do not satisfy the instantiated Lipschitz inequality

$$f_{i_k}(x^k - (1/L_k)f'_{i_k}(x^k)) \leqslant f_{i_k}(x^k) - \frac{1}{2L_k}\|f'_{i_k}(x^k)\|^2.$$

To avoid instability caused by comparing very small numbers, we only do this test when $\|f'_{i_k}(x^k)\|^2 > 10^{-8}$. To allow the algorithm to potentially achieve a faster rate due to a higher degree of local smoothness, we multiply L_k by $2^{(-1/n)}$ after each iteration.

5 Experimental Results

Our experiments compared an extensive variety of competitive FG and SG methods. In the supplementary material we compare to the IAG method and an extensive variety of SG methods, and we allow these competing methods to choose the best step-size in hindsight. However, our experiments in the main paper focus on the following methods, which we chose because they have no dataset-dependent tuning parameters:

- **Steepest**: The full gradient method described by iteration (3), with a line-search that uses cubic Hermite polynomial interpolation to find a step size satisfying the strong Wolfe conditions, and where the parameters of the line-search were tuned for the problems at hand.
- **AFG**: Nesterov's accelerated full gradient method [16], where iterations of (3) with a fixed step size are interleaved with an extrapolation step, and we use an adaptive line-search based on [27].
- **L-BFGS**: A publicly-available limited-memory quasi-Newton method that has been tuned for log-linear models.[3] This method is by far the most complicated method we considered.
- **Pegasos**: The state-of-the-art SG method described by iteration (4) with a step size of $\alpha_k = 1/\mu k$ and a projection step onto a norm-ball known to contain the optimal solution [28].
- **RDA**: The regularized dual averaging method of [12], another recent state-of-the-art SG method.
- **ESG**: The epoch SG method of [15], which runs SG with a constant step size and averaging in a series of epochs, and is optimal for non-smooth stochastic strongly-convex optimization.

[3] `http://www.di.ens.fr/~mschmidt/Software/minFunc.html`

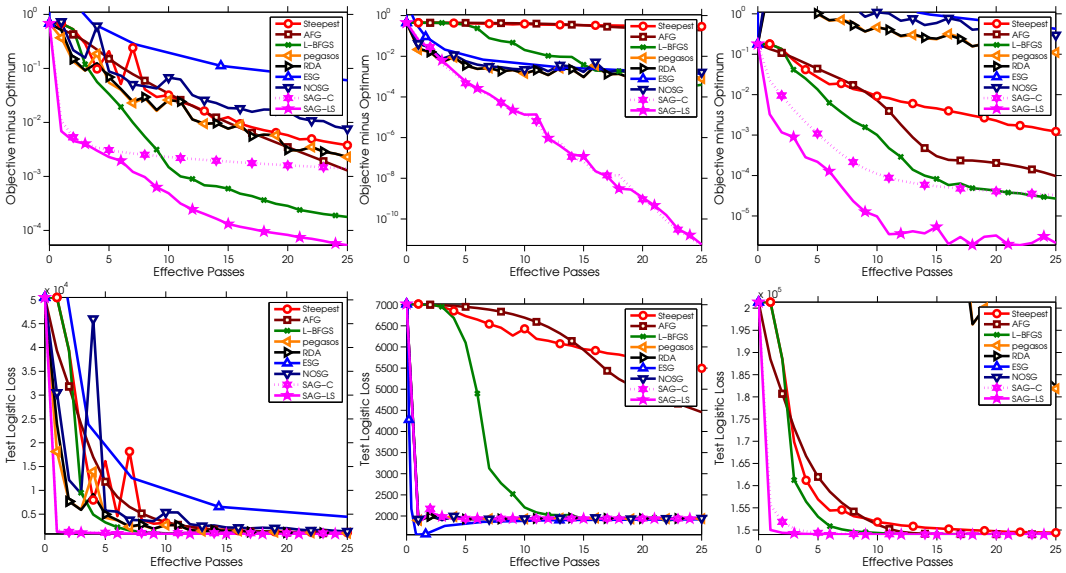

Figure 1: Comparison of optimization strategies for ℓ_2-regularized logistic regression. Top: training excess cost. Bottom: testing cost. From left to right are the results on the *protein*, *rcv1* and *covertype* data sets. This figure is best viewed in colour.

– **NOSG**: The nearly-optimal SG method of [19], which combines ideas from SG and AFG methods to obtain a nearly-optimal dependency on a variety of problem-dependent constants.

– **SAG**: The proposed stochastic average gradient method described by iteration (5) using the modifications discussed in the previous section. We used a step-size of $\alpha_k = 2/(L_k + n\lambda)$ where L_k is either set constant to the global Lipschitz constant (SAG-C) or set by adaptively estimating the constant with respect to the logistic loss function using the line-search described in the previous section (SAG-LS). The SAG-LS method was initialized with $L_0 = 1$.

In all the experiments, we measure the training and testing costs as a function of the number of effective passes through the data, measured as the number of f_i' evaluations divided by n. These results are thus independent of the practical implementation of the algorithms.

The theoretical convergence rates suggest the following strategies for deciding on whether to use an FG or an SG method:

1. If we can only afford one pass through the data, then an SG method should be used.
2. If we can afford to do many passes through the data (say, several hundred), then an FG method should be used.

We expect that the SAG iterations will be most useful between these two extremes, where we can afford to do more than one pass through the data but cannot afford to do enough passes to warrant using FG algorithms like L-BFGS. To test whether this is indeed the case on real data sets, we performed experiments on a set of freely available benchmark binary classification data sets. The *protein* ($n = 145751$, $p = 74$) data set was obtained from the KDD Cup 2004 website,[4] while the *rcv1* ($n = 20242$, $p = 47236$) and *covertype* ($n = 581012$, $p = 54$) data sets were obtained from the LIBSVM data website.[5] Although our method can be applied to any differentiable function, on these data sets we focus on an ℓ_2-regularized logistic regression problem, with $\lambda = 1/n$. We split each dataset in two, training on one half and testing on the other half. We added a (regularized) bias term to all data sets, and for dense features we standardized so that they would have a mean of zero and a variance of one. We plot the training and testing costs of the different methods for 30 effective passes through the data in Figure 1. In the supplementary material, we present additional experimental results including the test classification accuracy and results on different data sets.

We can observe several trends across the experiments from both the main paper and the supplementary material.

[4] http://osmot.cs.cornell.edu/kddcup

[5] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets

– **FG vs. SG**: Although the performance of SG methods can be catastrophic if the step size is not chosen carefully (e.g., the *covertype* data), with a carefully-chosen step-size the SG methods do substantially better than FG methods on the first few passes through the data (e.g., the *rcv1* data). In contrast, FG methods are not sensitive to the step size and because of their steady progress we also see that FG methods slowly catch up to the SG methods and eventually (or will eventually) pass them (e.g., the *protein* data).
– **(FG and SG) vs. SAG**: The SAG iterations seem to achieve the best of both worlds. They start out substantially better than FG methods, but continue to make steady (linear) progress which leads to better performance than SG methods. In some cases (*protein* and *covertype*), the significant speed-up observed for SAG in reaching low training costs also translates to reaching the optimal testing cost more quickly than the other methods.
– **IAG vs. SAG**: Our experiments (in the supplementary material) show that the IAG method performs similar to the regular FG method, and they also show the surprising result that the randomized SAG method outperforms the closely-related deterministic IAG method by a very large margin. This is due to the larger step sizes used by the SAG iterations, which would cause IAG to diverge.

6 Discussion

Optimal regularization strength: One might wonder if the additional hypothesis in Proposition 2 is satisfied in practice. In a learning context, where each function f_i is the loss associated to a single data point, L is equal to the largest value of the loss second derivative ξ (1 for the square loss, 1/4 for the logistic loss) times R^2, where R is a the uniform bound on the norm of each data point. Thus, the constraint $\frac{\mu}{L} \geqslant \frac{8}{n}$ is satisfied when $\lambda \geqslant \frac{8\xi R^2}{n}$. In low-dimensional settings, the optimal regularization parameter is of the form C/n [29] where C is a scalar constant, and may thus violate the constraint. However, the improvement with respect to regularization parameters of the form $\lambda = C/\sqrt{n}$ is known to be asymptotically negligible, and in any case in such low-dimensional settings, regular stochastic or batch gradient descent may be efficient enough in practice. In the more interesting high-dimensional settings where the dimension p of our covariates is not small compared to the sample size n, then all theoretical analyses we are aware of advocate settings of λ which satisfy this constraint. For example, [30] considers parameters of the form $\lambda = \frac{C}{\sqrt{n}}$ in the parametric setting, while [31] considers $\lambda = \frac{C}{n^\beta}$ with $\beta < 1$ in a non-parametric setting.

Training cost vs. testing cost: The theoretical contribution of this work is limited to the convergence rate of the training cost. Though there are several settings where this is the metric of interest (e.g., variational inference in graphical models), in many cases one will be interested in the convergence speed of the testing cost. Since the $O(1/k)$ convergence rate of the testing cost, achieved by SG methods with decreasing step sizes (and a single pass through the data), is provably optimal when the algorithm only accesses the function through unbiased measurements of the objective and its gradient, it is unlikely that one can obtain a linear convergence rate for the testing cost with the SAG iterations. However, as shown in our experiments, the testing cost of the SAG iterates often reaches its minimum quicker than existing SG methods, and we could expect to improve the constant in the $O(1/k)$ convergence rate, as is the case with online second-order methods [32].

Step-size selection and termination criteria: The three major disadvantages of SG methods are: (i) the slow convergence rate, (ii) deciding when to terminate the algorithm, and (iii) choosing the step size while running the algorithm. This paper showed that the SAG iterations achieve a much faster convergence rate, but the SAG iterations may also be advantageous in terms of tuning step sizes and designing termination criteria. In particular, the SAG iterations suggest a natural termination criterion; since the average of the y_i^k variables converges to $g'(x^k)$ as $\|x^k - x^{k-1}\|$ converges to zero, we can use $(1/n)\|\sum_i y_i^k\|$ as an approximation of the optimality of x^k. Further, while SG methods require specifying a sequence of step sizes and mispecifying this sequence can have a disastrous effect on the convergence rate [7, §2.1], our theory shows that the SAG iterations iterations achieve a linear convergence rate for any sufficiently small constant step size and our experiments indicate that a simple line-search gives strong performance.

Acknowledgements

Nicolas Le Roux, Mark Schmidt, and Francis Bach are supported by the European Research Council (SIERRA-ERC-239993). Mark Schmidt is also supported by a postdoctoral fellowship from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[2] L. Bottou and Y. LeCun. Large scale online learning. *NIPS*, 2003.

[3] C. H. Teo, Q. Le, A. J. Smola, and S. V. N. Vishwanathan. A scalable modular convex solver for regularized risk minimization. *KDD*, 2007.

[4] M. A. Cauchy. Méthode générale pour la résolution des systèmes d'équations simultanées. *Comptes rendus des séances de l'Académie des sciences de Paris*, 25:536–538, 1847.

[5] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*. Springer, 2004.

[6] A. Nemirovski and D. B. Yudin. *Problem complexity and method efficiency in optimization.* Wiley, 1983.

[7] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[8] A. Agarwal, P. L. Bartlett, P. Ravikumar, and M. J. Wainwright. Information-theoretic lower bounds on the oracle complexity of stochastic convex optimization. *IEEE Transactions on Information Theory*, 58(5), 2012.

[9] D. Blatt, A. O. Hero, and H. Gauchman. A convergent incremental gradient method with a constant step size. *SIAM Journal on Optimization*, 18(1):29–51, 2007.

[10] P. Tseng. An incremental gradient(-projection) method with momentum term and adaptive stepsize rule. *SIAM Journal on Optimization*, 8(2):506–531, 1998.

[11] Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

[12] L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, 2010.

[13] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[14] H. J. Kushner and G. Yin. *Stochastic approximation and recursive algorithms and applications*. Springer-Verlag, Second edition, 2003.

[15] E. Hazan and S. Kale. Beyond the regret minimization barrier: an optimal algorithm for stochastic strongly-convex optimization. *COLT*, 2011.

[16] Y. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. *Doklady AN SSSR*, 269(3):543–547, 1983.

[17] N.N. Schraudolph. Local gain adaptation in stochastic gradient descent. *ICANN*, 1999.

[18] P. Sunehag, J. Trumpf, SVN Vishwanathan, and N. Schraudolph. Variable metric stochastic approximation theory. *International Conference on Artificial Intelligence and Statistics*, 2009.

[19] S. Ghadimi and G. Lan. Optimal stochastic‘ approximation algorithms for strongly convex stochastic composite optimization. *Optimization Online*, July, 2010.

[20] J. Martens. Deep learning via Hessian-free optimization. *ICML*, 2010.

[21] A. Nedic and D. Bertsekas. Convergence rate of incremental subgradient algorithms. In *Stochastic Optimization: Algorithms and Applications*, pages 263–304. Kluwer Academic, 2000.

[22] M.V. Solodov. Incremental gradient algorithms with stepsizes bounded away from zero. *Computational Optimization and Applications*, 11(1):23–35, 1998.

[23] H. Kesten. Accelerated stochastic approximation. *Annals of Mathematical Statistics*, 29(1):41–59, 1958.

[24] B. Delyon and A. Juditsky. Accelerated stochastic approximation. *SIAM Journal on Optimization*, 3(4):868–881, 1993.

[25] D. P. Bertsekas. A new class of incremental gradient methods for least squares problems. *SIAM Journal on Optimization*, 7(4):913–926, 1997.

[26] M. P. Friedlander and M. Schmidt. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal of Scientific Computing*, 34(3):A1351–A1379, 2012.

[27] J. Liu, J. Chen, and J. Ye. Large-scale sparse logistic regression. *KDD*, 2009.

[28] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. *ICML*, 2007.

[29] P. Liang, F. Bach, and M. I. Jordan. Asymptotically optimal regularization in smooth parametric models. *NIPS*, 2009.

[30] K. Sridharan, S. Shalev-Shwartz, and N. Srebro. Fast rates for regularized objectives. *NIPS*, 2008.

[31] M. Eberts and I. Steinwart. Optimal learning rates for least squares SVMs using Gaussian kernels. *NIPS*, 2011.

[32] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. *NIPS*, 2007.