Small Stochastic Average Gradient Steps

Tobias Glasmachers
Institute for Neural Computation
Ruhr-University Bochum, Germany
`tobias.glasmachers@ini.rub.de`

Abstract

The stochastic average gradient (SAG) method was the first of its kind achieving linear convergence based on cheap-to-evaluate stochastic gradients. As such it established a milestone of optimization techniques for machine learning. In the few years since its inception is has spawned a large amount of related work. In this short paper we analyze the behavior of the SAG algorithm when operated in a non-standard regime, namely with learning rates smaller than the default by a factor of roughly the data set size. The resulting analysis is far more simple and intuitive than the (more powerful) original proof. As a very practical benefit, the optimization scheme enjoys a rigorously justified stopping criterion.

1 Introduction

We consider unconstrained minimization of an objective function f of the form

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

for $\theta \in \mathbb{R}^d$. For all $i \in \{1, \dots, n\}$ let the components f_i be convex and differentiable, and let their gradients $g_i(\theta) = \nabla f_i(\theta)$ be Lipschitz with constant L [2]. Let g denote the gradient of f:

$$g(\theta) = \nabla f(\theta) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\theta) = \frac{1}{n} \sum_{i=1}^{n} g_i(\theta).$$

We assume the existence of an optimal solution θ^*, the objective value of which is denoted by $f^* = f(\theta^*)$. This setting is known as optimization of a finite sum. Applied to machine learning, θ is the vector of model parameters and f_i are data point or mini-batch-wise losses, plus the regularizer.

The stochastic average gradient (SAG) method [4, 5] keeps track of a current iterate $\theta^{(t)} \in \mathbb{R}^d$. Upper indices in parentheses indicate the iteration counter $t \in \{1, 2, 3, \dots\} = \mathbb{N}$ ("time"). For brevity and readability we introduce the short notations $g^{(t)} = g(\theta^{(t)})$ and $g_i^{(t)} = g_i(\theta^{(t)})$. In addition to the iterate, SAG maintains vectors $y_i^{(t)} \in \mathbb{R}^d$, $i \in \{1, \dots, n\}$, and their average $a^{(t)} = \frac{1}{n} \sum_{i=1}^{n} y_i^{(t)}$, which serves as the update direction. In each iteration a random index $j^{(t)} \in \{1, \dots, n\}$ is drawn. For this index SAG computes the stochastic gradient $g_{j^{(t)}}(\theta^{(t-1)}) = g_{j^{(t)}}^{(t-1)}$, which is an unbiased estimate of the full gradient $g^{(t-1)}$. The stochastic gradient replaces the corresponding y-vector as follows:

$$y_i^{(t)} = \begin{cases} g_{j^{(t)}}^{(t-1)} & \text{if } i = j^{(t)} \\ y_i^{(t-1)} & \text{otherwise} \end{cases}$$

Hence, $a^{(t)}$ is interpreted as an averaged gradient. The gradients of the individual summands are evaluated at different locations, corresponding to previous iterates.

The work was presented at the NIPS workshop on Optimizing the Optimizers. Barcelona, Spain, 2016.

This average gradient can be updated cheaply as

$$a^{(t)} = a^{(t-1)} + \frac{1}{n} \left[g_{j^{(t)}}^{(t-1)} - g_{j^{(t)}}^{(t-n-1)} \right],$$

provided that n old gradients $y_1^{(t)}, \dots, y_n^{(t)}$ are stored. The relatively high storage complexity of $\mathcal{O}(nd)$ can be a limiting factor for the applicability of SAG. This problem has been addressed by memory-free methods, e.g., [3]. However, for (generalized) linear models only n scalar coefficients need to be stored (see [5]), reducing the memory requirements to $\mathcal{O}(n)$.

The new iterate

$$\theta^{(t)} = \theta^{(t-1)} - \eta \cdot a^{(t)}$$

is obtained by following the averaged gradient $a^{(t)}$ with constant learning rate $\eta = \frac{1}{16L}$. Realistically, the algorithm is initialized with $y_1^{(0)} = \dots = y_n^{(0)} = 0$ and some initial iterate $\theta^{(0)} \in \mathbb{R}^d$. An alternative initialization giving rise to a slightly nicer analysis is $y_i^{(0)} = g_i^{(0)} - g^{(0)}$ [5].

The merit of the SAG method is that it achieves convergence speed comparable with the full gradient method at the rather low iteration complexity of stochastic gradient descent (SGD) [1]. The saving in iteration complexity is a factor of n, which is a decisive advantage in particular for machine learning applications. In the general convex case, $\mathbf{E}\left[f\left(\frac{1}{t}\sum_{k=1}^{t}\theta^{(k)}\right)\right]$ converges to f^* at a rate of $\mathcal{O}(1/t)$. If f is strongly convex with constant $\mu > 0$, i.e., if $f(\theta) - \frac{\mu}{2}\|\theta - \theta^*\|^2$ is convex, then the SAG iterates converge at a linear rate: $\mathbf{E}\left[f(\theta^{(t)})\right] - f^* \in \mathcal{O}(\rho^t)$.

For the rate

$$\rho = 1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8n}\right\}$$

there are two regimes. They have different interpretations, depending on the point of view: we can fix either the number of summands n (usually corresponding to the size of the data set in machine learning), or the conditioning number L/μ of f. For fixed n, the regime $n \leq \frac{2L}{\mu}$ corresponds to ill-conditioned problems, while for fixed conditioning number L/μ it corresponds to small-scale problems. In this case the convergence rate is limited by the conditioning number. For well-conditioned or big data problems fulfilling $n \geq \frac{2L}{\mu}$ the convergence rate is limited by the data set size.

In this paper we focus on the latter case, i.e., strongly convex "big data" problems. This problem type is of utmost relevance for machine learning, where the number of data points n can grow extremely large (e.g., millions of points), while the conditioning number L/μ depends on the regularizer and the loss. It is hence under the control of the experimenter, and in particular independent of n. In this regime SAG converges at the rate $\rho = 1 - \frac{1}{8n}$. Although the learning rate is independent of n, the convergence rate slows down for growing n. This suggests that the same convergence speed can be achieved with a smaller learning rate. In the next section we derive such a scheme and analyze its convergence properties. We obtain a different type of rate, which is slightly improved in the best case. Our main point is that the smaller learning rate allows for a much simpler and more intuitive analysis, as compared to the original proof. This is achieved by keeping the average gradient close to the full gradient and applying a standard analysis. In contrast, the original analysis is based on the rather involved construction of a Lyapunov function dominating the Markov chain induced by the SAG algorithm. Although that method is very powerful, the central argument remains implicit. In contrast, our analysis is direct and gets along with elementary tools, and is hence far more accessible.

2 Analysis of SAG with small learning rate

In the sequel we analyze the SAG algorithm as defined above, with three changes. First, we replace the original setting $\eta = \frac{1}{16L}$ of the learning rate with the significantly smaller value $\eta = \frac{2}{5Ln}$. Note that this learning rate is smaller by a factor $\mathcal{O}(1/n)$, but still independent of t. Second, we select the indices $j^{(t)}$ in a round-robin fashion: $j^{(t)} = t \mod n$. This turns SAG into a deterministic method. Third, we initialize the algorithm with $y_i^{(0)} = g_i^{(0)}$, which is closely related to the above mentioned alternative initialization. With this setup we obtain the following linear convergence rate:

Theorem 1. *The SAG algorithm as outlined above converges at the linear rate*

$$\rho = 1 - \frac{2}{5n}\left(1 - \frac{9}{10n}\right)\frac{\mu}{L}.$$

Proof. The statement follows immediately from lemma 2 applied to a standard convergence analysis of the gradient descent method: in each iteration the function value improves by at least

$$f(\theta^{(t-1)}) - f(\theta^{(t)}) = f(\theta^{(t-1)}) - f(\theta^{(t-1)} - \eta a^{(t)}) \geq \eta \|a^{(t)}\| \|g^{(t-1)}\| - \frac{L}{2}\eta^2 \|a^{(t)}\|^2$$
$$\geq \frac{2}{5Ln}\frac{1}{2}\|g^{(t-1)}\|^2 - \frac{L}{2}\frac{4}{25L^2n^2}\frac{9}{4}\|g^{(t-1)}\|^2 = \frac{1}{5Ln}\left(1 - \frac{9}{10n}\right)\|g^{(t-1)}\|^2$$

where we have used the elementary inequalities $\frac{1}{2}\|g^{(t-1)}\| \leq \|a^{(t)}\| \leq \frac{3}{2}\|g^{(t-1)}\|$ from the proof of lemma 2. The sub-optimality $f(\theta^{(t-1)}) - f(\theta^*)$ is upper bounded by $\frac{\mu}{2}\|\theta^{(t-1)} - \theta^*\|^2 \leq \frac{1}{2\mu}\|g^{(t-1)}\|^2$. The rate

$$\frac{f(\theta^{(t)}) - f(\theta^*)}{f(\theta^{(t-1)}) - f(\theta^*)} = 1 - \frac{f(\theta^{(t-1)}) - f(\theta^{(t)})}{f(\theta^{(t-1)}) - f(\theta^*)} \leq 1 - \frac{2}{5n}\left(1 - \frac{9}{10n}\right)\frac{\mu}{L}$$

follows. □

The following lemma establishes the proximity of average and full gradient.

Lemma 2. *The average gradient $a^{(t)}$ deviates from the full gradient $g^{(t-1)}$ by at most half of its norm, i.e.,*

$$\left\|a^{(t)} - g^{(t-1)}\right\| \leq \frac{1}{2} \cdot \left\|g^{(t-1)}\right\|.$$

Proof. With the given initialization it holds $a^{(0)} = g^{(0)}$. The first update does not change a, hence we obtain $a^{(1)} = g^{(0)}$. So the statements holds trivially for the first iteration $t = 1$. We apply an inductive argument for handling $t > 1$. Assume that the statement holds for all $t' < t$. In the sequel we will apply the update equation $\theta^{(t)} = \theta^{(t-1)} - \eta a^{(t)}$ and the Lipschitz property, which holds for g as well as for each component g_i, and the triangle inequality.

Due to the round robin selection the difference of interest can be written as

$$a^{(t)} - g^{(t-1)} = \frac{1}{n}\sum_{i=1}^{n}\left[g_{i(t-i+1)}^{(t-i)} - g_{i(t-i+1)}^{(t-1)}\right],$$

where we assume for simplicity that negative time indices are clamped to zero. The norm of the difference is bounded by

$$\begin{aligned}\left\|a^{(t)} - g^{(t-1)}\right\| &\leq \frac{L}{n}\sum_{i=1}^{n}\left\|\theta^{(t-j)} - \theta^{(t-1)}\right\| \\ &\leq \frac{\eta L}{n}\sum_{i=1}^{n}\sum_{k=1}^{i-1}\left\|a^{t-k}\right\| \\ &= \frac{2}{5n^2}\sum_{k=1}^{n-1}(n-k)\cdot\left\|a^{t-k}\right\|. \qquad (1)\end{aligned}$$

From the inductive assumption that the lemma holds for all $t' < t$ we obtain $\|a^{(t')}\| \leq \frac{3}{2}\|g^{(t'-1)}\|$. However, we need a bound in terms of the subsequent gradient in iteration t', hence one update step must be taken into account:

$$\left\|a^{(t')}\right\| \leq \frac{3}{2}\left[\left\|g^{(t)}\right\| + \eta L\left\|a^{(t)}\right\|\right]$$
$$\Rightarrow \quad \left\|a^{(t')}\right\| \leq \frac{1}{1 - \frac{3}{2}\eta L}\left\|g^{(t')}\right\| = \frac{1}{1 - \frac{3}{5n}}\left\|g^{(t')}\right\| = \frac{5n}{5n-3}\left\|g^{(t')}\right\| \qquad (2)$$

In order to handle the bound (1) we need the ability to bridge longer time gaps, as follows:

$$\begin{aligned}
\left\|g^{(t')} - g^{(t'-1)}\right\| &\leq \eta L \left\|a^{(t')}\right\| \leq \eta L \frac{5n}{5n-3} \left\|g^{(t')}\right\| = \frac{2}{5n-3} \left\|g^{(t')}\right\| \\
\Rightarrow \quad \left\|g^{(t'-1)}\right\| &\leq \left(1 + \frac{2}{5n-3}\right) \left\|g^{(t')}\right\| = \frac{5n-1}{5n-3} \left\|g^{(t')}\right\| \\
\Rightarrow \quad \left\|g^{(t'-k)}\right\| &\leq \left(\frac{5n-1}{5n-3}\right)^k \left\|g^{(t')}\right\| \qquad (3)
\end{aligned}$$

For $k \leq n$ the powers of the growth factor are bounded by

$$\left(\frac{5n-1}{5n-3}\right)^k \leq \left(\frac{5n-1}{5n-3}\right)^n \leq 2 \qquad \text{for all } n \in \mathbb{N}. \qquad (4)$$

For $n = 1$ the last bound holds with equality. For $n \geq 2$ we consider the natural logarithm $\ell = n\Big[\ln(5n-1) - \ln(5n-3)\Big]$ of the left hand side. In the interval $[5n-3, 5n-1]$, the slope of the natural logarithm is upper bounded by $1/(5n-3)$, hence we get $\ell \leq n \int_{5n-3}^{5n-1} \frac{1}{5n-3} dx = \frac{2n}{5n-3}$. For $n \geq 2$ the fraction is upper bounded by $4/7 < \log(2)$.

We put everything together by plugging (4) into (3) and then (2) and (3) into (1):

$$\begin{aligned}
\left\|a^{(t)} - g^{(t-1)}\right\| &\leq \frac{2}{5n^2} \sum_{k=1}^{n-1} (n-k) \cdot \left\|a^{t-k}\right\| \\
&\leq \frac{2}{n(5n-3)} \sum_{k=1}^{n-1} (n-k) \cdot \left\|g^{t-k}\right\| \\
&\leq \left[\frac{4}{n(5n-3)} \sum_{k=1}^{n-1} (n-k)\right] \cdot \left\|g^{t-1}\right\| \\
&= \frac{2(n-1)}{5n-3} \cdot \left\|g^{t-1}\right\| \\
&\leq \frac{1}{2} \cdot \left\|g^{t-1}\right\| \qquad \text{for all } n \in \mathbb{N}
\end{aligned}$$

This proves the inequality for iteration t, and by induction for all iterations. $\square$

3 Discussion and Conclusion

In this paper we have provided an alternative analysis of the SAG algorithm, operated with a learning rate that is smaller than the default by a factor of order n. The analysis is based on the insight that the evaluation of the component gradients at different locations should not be too problematic since the resulting effect is bounded by the Lipschitz constant of the gradient—in contrast to the variability of g_i w.r.t. i, which is unbounded. With the relatively large learning rate of $\frac{1}{16L}$ this effect does not clearly show up. When operating the algorithm with a relatively small learning rate of $\frac{2}{5Ln}$ the effect is pronounced, since Lemma 2 ensures that the average gradient is indeed close to the exact gradient. Unsurprisingly, the resulting convergence rate suffers for badly conditioned problems, which is also the case for the full gradient method.

At least in the "big data" regime $n \geq \frac{2L}{\mu}$ there is no disadvantage in terms of the dependency of the convergence rate on n, which is $\mathcal{O}(1/n)$ in both cases. However, our rate depends on the conditioning number, which is a clear disadvantage. Our method is hence most relevant for noisy problems with a relatively high Bayes risk, requiring strong regularization. This keeps the conditioning number close to one. In that case the proposed scheme can outperform standard SAG. However, the main merit of the approach is the greatly simplified analysis.

Last but not least, it was already proposed in [5] to use $\|a^{(t)}\|$ as a stopping criterion. Due to Lemma 2 this is a rigorously justified criterion, provided that $\|g^{(t)}|$ is a sufficient indicator of approximate optimality. This solves one of the practically most annoying obstacles when training learning machines with stochastic gradients.

References

[1] L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012.

[2] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[3] R. Johnson and T. Zhang. Accelerating Stochastic Gradient Descent using Predictive Variance Reduction. In *Advances in Neural Information Processing Systems (NIPS)*, pages 315–323, 2013.

[4] N. Le Roux, M. Schmidt, and F. Bach. A Stochastic Gradient Method with an Exponential Convergence Rate for Finite Training Sets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2663–2671, 2012.

[5] M. Schmidt, N. Le Roux, and F. Bach. Minimizing Finite Sums with the Stochastic Average Gradient. Technical Report arXiv:1309.2388v2, arxiv.org, 2016.