Speed learning on the fly

Pierre-Yves Massé, Yann Ollivier

Abstract

The practical performance of online stochastic gradient descent algorithms is highly dependent on the chosen step size, which must be tediously hand-tuned in many applications. The same is true for more advanced variants of stochastic gradients, such as SAGA, SVRG, or AdaGrad. Here we propose to adapt the step size by performing a gradient descent on the step size itself, viewing the whole performance of the learning trajectory as a function of step size. Importantly, this adaptation can be computed online at little cost, without having to iterate backward passes over the full data.

Introduction

This work aims at improving gradient ascent procedures for use in machine learning contexts, by adapting the step size of the descent as it goes along.

Let $\ell_0, \ell_1, \ldots, \ell_t, \ldots$ be functions to be maximised over some parameter space Θ. At each time t, we wish to compute or approximate the parameter $\theta_t^* \in \Theta$ that maximizes the sum

$$L_t(\theta) := \sum_{s \leq t} \ell_s(\theta). \tag{1}$$

In the experiments below, as in many applications, $\ell_t(\theta)$ writes $\ell(x_t, \theta)$ for some data $x_0, x_1, \ldots, x_t, \ldots$

A common strategy, especially with large data size or dimensionality [Bot10], is the online stochastic gradient ascent (SG)

$$\theta_{t+1} = \theta_t + \eta \, \partial_\theta \ell_t(\theta_t) \tag{2}$$

with step size η, where $\partial_\theta \ell_t$ stands for the Euclidean gradient of ℓ_t with respect to θ.

Such an approach has become a mainstay of both the optimisation and machine learning communities [Bot10]. Various conditions for convergence exist, starting with the celebrated article of Robbins and Monro [RM51], or later [KC78]. Other types of results are proved in convex settings,

Several variants have since been introduced, in part to improve the convergence of the algorithm, which is much slower in stochastic than than in deterministic settings. For instance, algorithms such as SAGA, Stochastic Variance Reduced Gradient (SVRG) or Stochastic Average Gradient (SAG) [DBLJ14, JZ13, SLRB13], perform iterations using a comparison between the latest gradient and an average of past gradients. This reduces the variance of the resulting estimates and allows for nice convergence theorems [DBLJ14, SLRB13], provided a reasonable step size η is used.

Influence of the step size. The ascent requires a parameter, the step size η, usually called "learning rate" in the machine learning community. Empirical evidence highligting the sensitivity of the ascent to its actual numerical value exists aplenty; see for instance the graphs in Section 3.2.1. Slow and tedious hand-tuning is therefore mandatory in most applications. Moreover, admittable values of η depend on the parameterisation retained—except for descents described in terms of Riemannian metrics [Ama98], which provide some degree of parameterisation-invariance.

Automated procedures for setting reasonable value of η are therefore of much value. For instance, AdaGrad [DHS11] divides the derivative $\partial_\theta \ell_t$ by a root mean square average of the magnitude of its recent values, so that the steps are of size approximately 1; but this still requires a "master step size" η.

Shaul, Zhang and LeCun in [SZL13] study a simple separable quadratic loss model and compute the value of η which minimises the expected loss after each parameter update. This value can be expressed in terms of computable quantities depending on the trajectory of the descent. These quantities still make sense for non-quadratic models, making this idea amenable to practical use.

More recently, Maclaurin, Douglas and Duvenaud [MDA15] propose to directly conduct a gradient ascent on the hyperparameters (such as the learning rate η) of any algorithm. The gradients with respect to the hyperparameters are computed exactly by "chaining derivatives backwards through the entire training procedure" [MDA15]. Consequently, this approach is extremely impractical in an online setting, as it optimizes the learning rate by performing several passes, each of which goes backwards from time t to time 0.

Finding the best step size. The ideal value of the step size η would be the one that maximizes the cumulated objective function (1). Write $\theta_t(\eta)$ for the parameter value obtained after t iterations of the gradient step (2) using a given value η, and consider the sum

$$\sum_{s \leq t} \ell_s(\theta_s(\eta)). \tag{3}$$

Our goal is to find an online way to approximate the value of η that provides the best value of this sum. This can be viewed as an ascent on the space of stochastic ascent algorithms.

We suggest to update η through a stochastic gradient ascent on this sum:

$$\eta \leftarrow \eta + \alpha \frac{\partial}{\partial \eta} \ell_t(\theta_t(\eta)) \tag{4}$$

and then to use, at each time, the resulting value of η for the next gradient step (2).

The ascent (4) on η depends, in turn, on a step size α. Hopefully, the dependance on α of the whole procedure is somewhat lower than that of the original stochastic gradient scheme on its step size η.

This approach immediately extends to other stochastic gradient algorithms; in what follows we apply it both to the standard SG ascent and to the SVRG algorithm.

The main point in this approach is to find efficient ways to compute or approximate the derivatives $\frac{\partial}{\partial \eta} \ell_t(\theta_t(\eta))$. Indeed, the value $\theta_t(\eta)$ after t steps depends on the whole trajectory of the algorithm, and so does its derivative with respect to η.

After reviewing the setting for gradient ascents in Section 1, in Section 2.1 we provide an exact but impractical way of computing the derivatives $\frac{\partial}{\partial \eta} \ell_t(\theta_t(\eta))$. Sections 2.2–2.3 contain the main contribution: SG/SG and SG/AG, practical algorithms to adjust η based on two approximations with respect to these exact derivatives.

Section 2.4 extends this to other base algorithms such as SVRG. In Section 4 one of the approximations is justified by showing that it computes a derivative, not with respect to a fixed value of η as in (4), but with respect to the sequences of values of η effectively used along the way. This also suggests improved algorithms.

Section 3 provides experimental comparisons of gradient ascents using traditional algorithms with various values of η, and the same algorithms where η is self-adjusted according to our scheme. The comparisons are done on three sets of synthetic data: a one-dimensional Gaussian model, a one-dimensional Bernoulli model and a 50-dimensional linear regression model: these simple models already exemplify the strong dependence of the traditional algorithms on the value of η.

Terminology. We say that an algorithm is of type "LLR" for "Learning the Learning Rate" when it updates its step size hyperparameter η as it unfolds. We refer to LLR algorithms by a compound abbreviation: "SVRG/SG", for instance, for an algorithm which updates its parameter θ through SVRG and its hyperparameter η through an SG algorithm on η.

1 The Stochastic Gradient algorithm

To fix ideas, we define the Stochastic Gradient (SG) algorithm as follows. In all that follows, $\Theta = \mathbb{R}^n$ for some n.[1] The functions ℓ_t are assumed to be smooth. In all our algorithms, the index t starts at 0.

Algorithm 1 (Stochastic Gradient)**.** *We maintain $\theta_t \in \Theta$ (current parameter), initialised at some arbitrary $\theta_0 \in \Theta$. We fix $\eta \in \mathbb{R}$. At each time t, we fix a rate $f(t) \in \mathbb{R}$. The update equation reads:*

$$\theta_{t+1} = \theta_t + \frac{\eta}{f(t)} \partial_\theta \ell_t(\theta_t). \tag{5}$$

The chosen rate $f(t)$ usually satisfies the well-known Robbins–Monro conditions [RM51]:

$$\sum_{t \geq 0} f(t)^{-1} = \infty, \qquad \sum_{t \geq 0} f(t)^{-2} < \infty. \tag{6}$$

The divergence of the sum of the rates allows the ascent to go anywhere in parameter space, while the convergence of the sum of the squares ensures that variance remains finite. Though custom had it that small such rates should be chosen, such as $f(t) = 1/t$, recently the trend bucked towards the use of large ones, to allow for quick exploration of the parameter space. Throughout the article and experiments we use one such rate:

$$f(t) = \sqrt{t+2} \log(t+3). \tag{7}$$

2 Learning the learning rate on a stochastic gradient algorithm

2.1 The loss as a function of step size

To formalise what we said in the introduction, let us define, for each $\eta \in \mathbb{R}$, the sequence

$$(\theta_0, \theta_1, \theta_2, \ldots) \tag{8}$$

obtained by iterating (5) from some initial value θ_0. Since they depend on η, we introduce, for each $t > 0$, the operator

$$T_t : \eta \in \mathbb{R} \mapsto T_t(\eta) \in \Theta, \tag{9}$$

which maps any $\eta \in \mathbb{R}$ to the parameter θ_t obtained after t iterations of (5). T_0 maps every η to θ_0. For each $t \geq 0$, the map T_t is a regular function of

[1] Θ may also be any Riemannian manifold, a natural setting when dealing with gradients. Most of the text is written in this spirit.

η. As explained in the introduction, we want to optimise η according to the function:

$$\mathcal{L}_t(\eta) := \sum_{s \leq t} \ell_s(T_s(\eta)), \tag{10}$$

by conducting an online stochastic gradient ascent on it. We therefore need to compute the derivative in (4):

$$\frac{\partial}{\partial \eta} \ell_t(T_t(\eta)). \tag{11}$$

To act more decisively on the order of magnitude of η, we perform an ascent on its logarithm, so that we actually need to compute[2]:

$$\frac{\partial}{\partial \log \eta} \ell_t(T_t(\eta)). \tag{12}$$

Now, the derivative of the loss at time t with respect to η can be computed as the product of the derivative of ℓ_t with respect to θ (the usual input of SG) and the derivative of θ_t with respect to η:

$$\frac{\partial}{\partial \log \eta} \ell_t(T_t(\eta)) = \partial_\theta \ell_t(T_t(\eta)) \cdot A_t(\eta) \tag{13}$$

where

$$A_t(\eta) := \frac{\partial T_t(\eta)}{\partial \log \eta}. \tag{14}$$

Computation of the quantity A_t and its approximation h_t to be introduced later, are the main focus of this text.

Lemma 1. *The derivative $A_t(\eta)$ may be computed through the following recursion equation. $A_0(\eta) = 0$ and, for $t \geq 0$,*

$$A_{t+1}(\eta) = A_t(\eta) + \frac{\eta}{f(t)} \partial_\theta \ell_t(T_t(\eta)) + \frac{\eta}{f(t)} \partial_\theta^2 \ell_t(T_t(\eta)) \cdot A_t(\eta). \tag{15}$$

The proof lies in Section C.1. This update of A involves the Hessian of the loss function with respect to θ, evaluated in the direction of A_t. Often this quantity is unavailable or too costly. Therefore we will use a finite difference approximation instead:

$$\partial_\theta^2 \ell_t(T_t(\eta)) \cdot A_t(\eta) \approx \partial_\theta \ell_t \left(T_t(\eta) + A_t(\eta)\right) - \partial_\theta \ell_t(T_t(\eta)). \tag{16}$$

This design ensures that the resulting update on $A_t(\eta)$ uses the gradient of ℓ_t only once:

$$A_{t+1}(\eta) \approx A_t(\eta) + \frac{\eta}{f(t)} \partial_\theta \ell_t \left(T_t(\eta) + A_t(\eta)\right). \tag{17}$$

An alternative approach would be to compute the Hessian in the direction A_t by numerical differentiation.

[2] This is an abuse of notation as T_t is not a function of $\log \eta$ but of η. Formally, we would need to replace T_t with $T_t \circ \exp$, which we refrain from doing to avoid burdensome notation.

2.2 LLR on SG: preliminary version with simplified expressions (SG/SG)

Even with the approximation above, computing the quantities A_t would have a quadratic cost in t: each time we update η thanks to (4), we would need to compute anew all the $A_s(\eta)$, $s \leq t$, as well as the whole trajectory $\theta_t = T_t(\eta)$, at each iteration t. We therefore replace the $A_t(\eta)$'s by online approximations, the quantities h_t, which implement the same evolution equation (17) as A_t, disregarding the fact that η may have changed in the meantime. These quantities will be interpreted more properly in Section 4 as derivatives taken along the effective trajectory of the ascent. This yields the SG/SG algorithm.

Algorithm 2 (SG/SG)**.** *We maintain $\theta_t \in \Theta$ (current parameter), $\eta_t \in \mathbb{R}$ (current step size) and $h_t \in T_{\theta_t}\Theta$ (approximation of the derivative of θ_t with respect to $\log(\eta)$).*

The first two are initialised arbitrarily, and h_0 is set to 0.

The update equations read:

$$\begin{cases} \log \eta_{t+1} = \log \eta_t + \dfrac{1}{\mu_t}\, \partial_\theta \ell_t(\theta_t) \cdot h_t \\ h_{t+1} = h_t + \dfrac{\eta_{t+1}}{f(t)}\, \partial_\theta \ell_t\left(\theta_t + h_t\right) \\ \theta_{t+1} = \theta_t + \dfrac{\eta_{t+1}}{f(t)} \partial_\theta \ell_t(\theta_t)\,, \end{cases} \tag{18}$$

where μ_t is some learning rate on $\log \eta$, such as $\mu_t = \sqrt{t+2}\log(t+3)$.

2.3 LLR on SG: efficient version (SG/AG)

To obtain better performances, we actually use an adagrad-inspired scheme to update the logarithm of the step size.

Algorithm 3 (SG/AG)**.** *We maintain $\theta_t \in \Theta$ (current parameter), $\eta_t \in \mathbb{R}$ (current step size), $h_t \in T_{\theta_t}\Theta$ (approximation of the derivative of θ_t with respect to $\log(\eta)$), $n_t \in \mathbb{R}$ (average of the squared norms of $\partial \ell_t \circ T_t / \partial \log \eta$), and $d_t \in \mathbb{R}$ (renormalising factor for the computation of n_t).*

θ et η are initially set to θ_0 and η_0, the other variables are set to 0.

At each time t, we compute $\mu_t \in \mathbb{R}$ (a rate used in several updates), and $\lambda_t \in \mathbb{R}$ (the approximate derivative of $\ell_t \circ \theta_t$ with respect to $\log(\eta)$ at η_t).

The update equations read:

$$\begin{cases} \mu_t = \sqrt{t+2}\log(t+3) \\ \lambda_t = \partial_\theta \ell_t(\theta_t) \cdot h_t \\ d_{t+1} = \left(1 - \frac{1}{\mu_t}\right) d_t + \frac{1}{\mu_t} \\ n_{t+1}^2 = \left(\left(1 - \frac{1}{\mu_t}\right) n_t^2 + \frac{1}{\mu_t}\lambda_t^2\right) d_{t+1}^{-1} \\ \log \eta_{t+1} = \log \eta_t + \frac{1}{\mu_t}\frac{\lambda_t}{n_{t+1}} \\ h_{t+1} = h_t + \frac{\eta_{t+1}}{f(t)}\,\partial_\theta \ell_t\left(\theta_t + h_t\right) \\ \theta_{t+1} = \theta_t + \frac{\eta_{t+1}}{f(t)}\,\partial_\theta \ell_t(\theta_t). \end{cases} \tag{19}$$

2.4 LLR on other Stochastic Gradient algorithms

The LLR procedure may be applied to any stochastic gradient algorithm of the form

$$\theta_{t+1} = F(\theta_t, \eta_t) \tag{20}$$

where θ_t may store all the information maintained by the algorithm, not necessarily just a parameter value. Appendix B presents the algorithm in this case. Appendix A presents SVRG/AG, which is the particular case of this procedure applied to SVRG with an AdaGrad scheme for the update of η_t.

3 Experiments on SG and SVRG

We now present the experiments conducted to test our procedure. We first describe the experimental set up, then discuss the results.

3.1 Presentation of the experiments

We conducted ascents on synthetic data generated by three different probabilistic models: a one-dimensional Gaussian model, a Bernoulli model and a 50-dimensional linear regression model. Each model has two components: a generative distribution, and a set of distributions used to approximate the former.

One Dimensional Gaussian Model. The mean and value of the Gaussian generative distribution were set to 5 and 2 respectively. Let us note

p_θ the density of a standard Gaussian random variable. The function to maximise we used is:

$$\ell_t(\theta) = \log p_\theta(x_t) = -\frac{1}{2}(x_t - \theta)^2. \tag{21}$$

Bernoulli model. The parameter in the standard parameterisation for the Bernoulli model was set to $p = 0.3$, but we worked with a logit parameterisation $\theta = \log(p/(1-p))$ for both the generative distribution and the discriminative function. The latter is then:

$$\ell_t(\theta) = \theta \cdot x_t - \log\left(1 + e^\theta\right). \tag{22}$$

Fifty-dimensional Linear Regression model. In the last model, we compute a fixed random matrix M. We then draw samples Z from a standard 50-dimensional Gaussian distribution. We then use M to make random linear combinations $X = MZ$ of the coordinates of the Z vectors. Then we observe X and try to recover first coordinate of the sample Z. The solution θ^* is the first row of the inverse of M. Note Y the first coordinate of Z so that the regression pair is (X, Y). We want to maximise:

$$\ell_t(\theta) = -\frac{1}{2}\left(y_t - \theta \cdot x_t\right)^2, \tag{23}$$

For each model, we drew 2500 samples from the data (7500 for the 50-dimensional model), then conducted ascents on those with on the one hand the SG and SVRG algorithms, and on the other hand their LLR counterparts, SG/SG and SVRG/SG, respectively.

3.2 Description and analysis of the results

For each model, we present four different types of results. We start with the trajectories of the ascents for several initial values of η (in the 50-dimensional case, we plot the first entry of $\theta^T \cdot M$). Then we present the cumulated regrets. Next we show the evolution of the logarithm of η_t along the ascents for the LLR algorithms. Finally, we compare this to trajectories of the non-adaptive algorithms with good initial values of η. Each time, we present three figures, one for each model.

Each figure of Figures 1 to 3 is made of four graphs: the upper ones are those of SG and SVRG, the lower ones are those of SG/SG and SVRG/SG. Figures 1 to 3 present the trajectories of the ascents for several orders of magnitude of η_0, while Figures 4 to 6 present the cumulated regrets for the same η_0's. The trajectory of the running maximum likelihood (ML) is displayed in red in each plot.

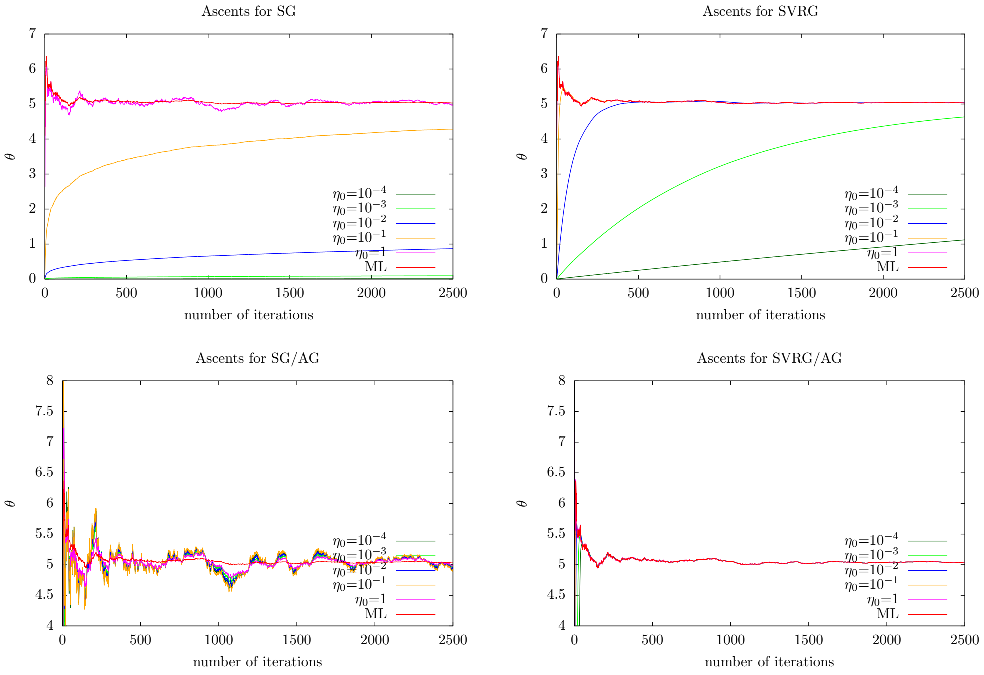


Figure 1: Trajectories of the ascents for a Gaussian model in one dimension for several algorithms and several η_0's

3.2.1 Trajectories of θ

Each figure for the ascent looks the same: there are several well distinguishable trajectories in the graphs of the standard algorithms, the upper ones, while trajectories are much closer to each other in those of the LLR algorithms, the lower ones.

Indeed, for many values of η, the standard algorithms will perform poorly. For instance, low values of η will result in dramatically low convergence towards the ML, as may be seen in some trajectories of the SG graphs. The SVRG algorithm performs noticeably better, but may start to oscillate, as in Figures 2 and 3.

These inconveniences are significantly improved by the use of LLR procedures. Indeed, in each model, almost every trajectory gets close to that of the ML in the SG/AG graphs. In the SVRG/AG graphs, the oscillations are overwhelmingly damped. Improvements for SG, though significant, are not as decisive in the linear regression model as in the other two, probably due to its greater complexity.

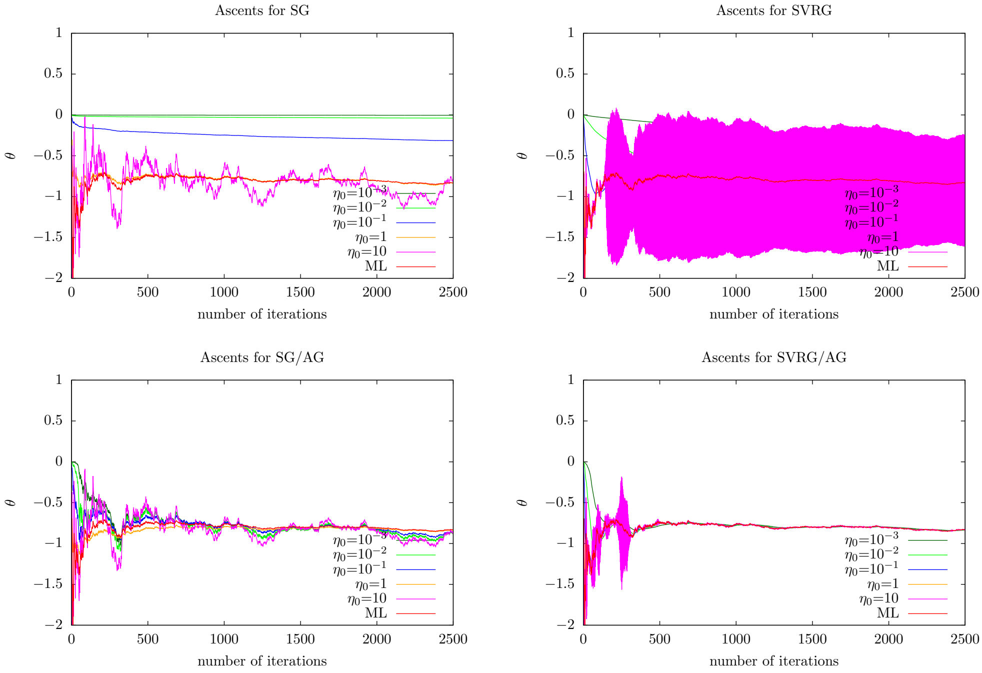


Figure 2: Trajectories of the ascents for a Bernoulli model for several algorithms and several η_0's

3.2.2 Cumulated regrets

Each curve of Figures 4 to 6 represents the difference between the cumulated regret of the algorithm used and that of the ML, for the η_0 chosen. The curves of SG and SVRG all go upwards, which means that the difference increases with time, whereas those of SG/AG and SVRG/SG tend to stagnate strikingly quickly. Actually, the trajectories for the linear regression model do not stagnate, but they are still significantly better for the LLR algorithms than for the original ones. The stagnation means that the values of the parameter found by these algorithms are very quickly as good as the Maximum Likelihood for the prediction task. Arguably, the fluctuations of the ascents around the later are therefore not a defect of the model: the cumulated regret graphs show that they are irrelevant for the minimisation at hand.

3.2.3 Evolution of the step size of the LLR algorithms during the ascents

Figures 7 to 9 show the evolution of the value of the logarithm of η_t in the LLR procedures for the three models, in regard of the trajectories of the corresponding ascents. For the Gaussian and Bernoulli models, in Figures 7

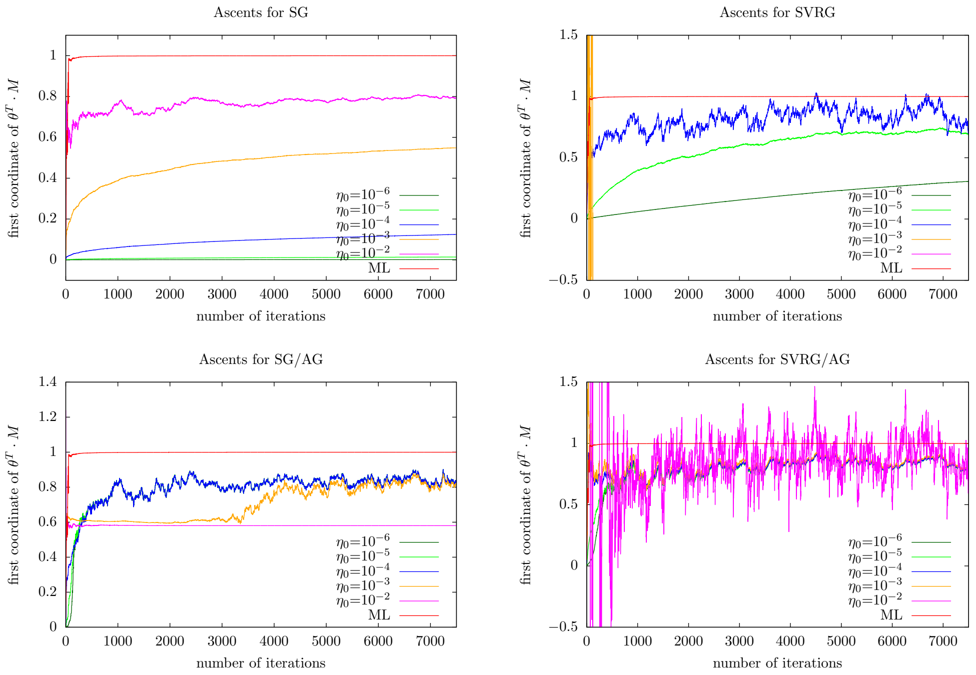


Figure 3: Trajectories of the ascents for a 50-dimensional linear regression model for several algorithms and several η_0's

and 8, $\log(\eta_t)$ tends to stagnate quite quickly. This may seem a desirable behaviour : the algorithms have reached good values for η_t, and the ascent may accordingly proceed with those. However, this analysis may seem somewhat unsatisfactory due to the $1/f(t)$ dampening term in the parameter update, which remains unaltered by our procedure. For the linear regression model, in Figure 9, the convergence takes longer in the SG/SG case, and even in the SVRG/SG one, which may be explained again by the complexity of the model.

3.2.4 LLR versus hand-crafted learning rates

Figures 10 to 12 show the trajectories of the ascents for LLR algorithms with poor initial values of the step size, compared to the trajectories of the original algorithms with hand-crafted optimal values of η. The trajectories of the original algorithms appear in red. They possess only two graphs each, where all the trajectories are pretty much undistinguishable from another. This shows that the LLR algorithms show acceptable behaviour even with poor initial values of η, proving the procedure is able to rescue very badly initialised algorithms. However, one caveat is that the LLR procedure encounters difficulties dealing with too large values of η_0, and is much more efficient at dealing with small values of η_0. We have no satisfying explana-

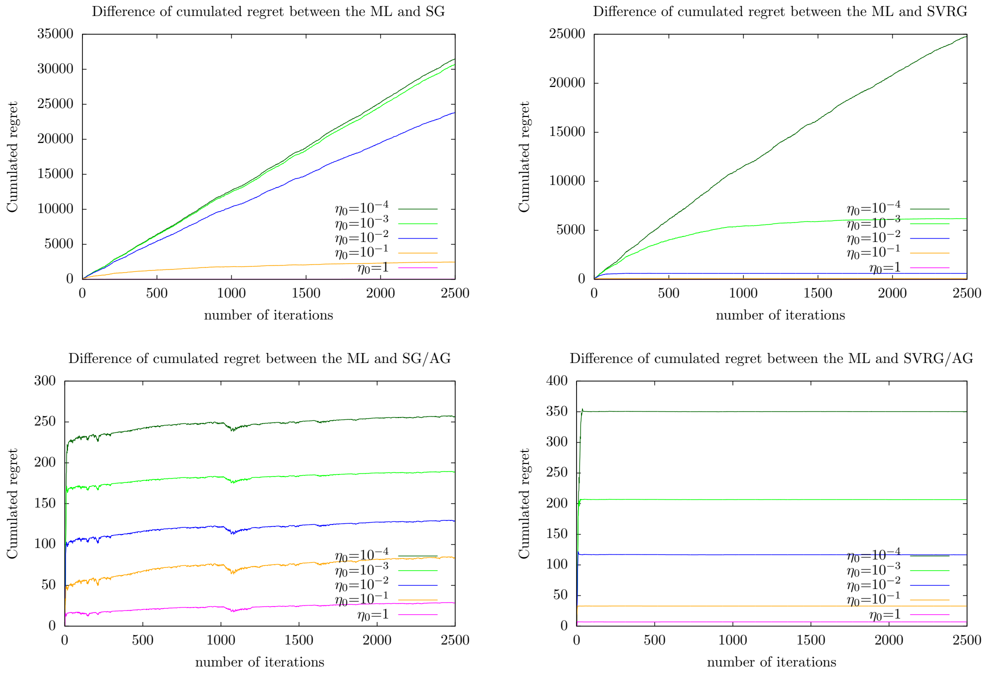


Figure 4: Difference between the cumulated regrets of the algorithm and of the ML for a Gaussian model in one dimension for several algorithms and several η_0's

tion of this phenomenon yet. We thus suggest, in practice, to underestimate rather than overestimate the initial value η_0.

3.3 η_t in a quadratic model

In a quadratic deterministic one-dimensional model, where we want to maximise:

$$f(\theta) = -\alpha \frac{x^2}{2}, \tag{24}$$

SG is numerically stable if, and only if,

$$\left| 1 - \frac{\alpha \eta}{f(t)} \right| < 1, \tag{25}$$

that is

$$\frac{\eta}{2f(t)} < \alpha^{-1}. \tag{26}$$

Each graph of Figure 13 has two curves, one for the original algorithm, the other for its LLR version. The curve of the LLR version goes down quickly, then much more slowly, while the other curve goes down slowly all the time. This shows that, for $\alpha = 10^8$, the ratio above converges quickly towards

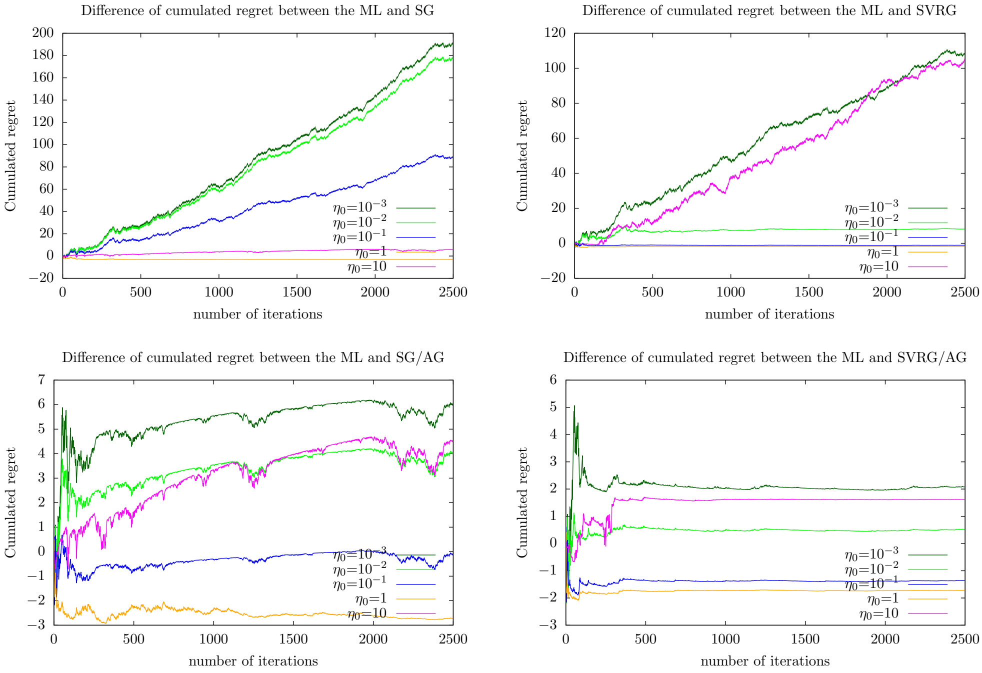


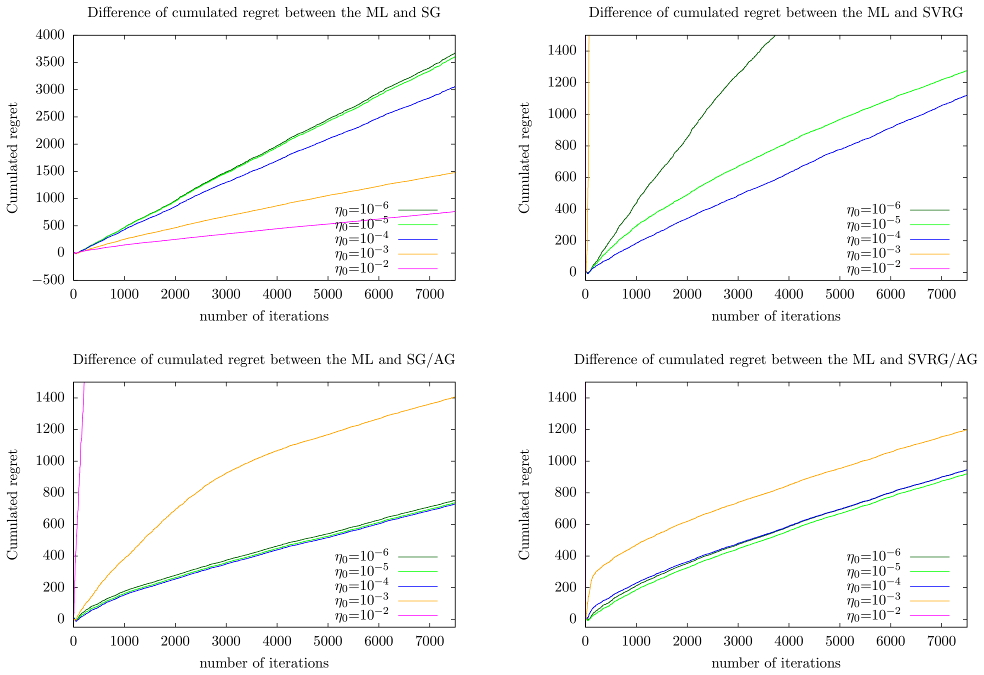
Figure 5: Difference between the cumulated regrets of the algorithm and of the ML for a Bernoulli model for several algorithms and several η_0's

α^{-1} for SG/AG and SVRG/AG, showing the ascent on η is indeed efficient. Then, the algorithm has converged, and η_t stays nearly constant, so much so that the LLR curve behaves like the other one. However, the convergence of η_t happens too slowly: θ_t takes very large values before η_t reaches this value, and even though it eventually converges to 0, such behaviour is unacceptable in practise.

4 A pathwise interpretation of the derivatives

Until now, we have tried to optimise the step size for a stochastic gradient ascent. This may be interpreted as conducting a gradient ascent on the subspace of the ascent algorithms which gathers the stochastic gradient algorithms, parametrised by $\eta \in \mathbb{R}$. However, we had to replace the $A_t(\eta)$'s by the h_t's because computing the former gave our algorithm a quadratic complexity in time. Indeed, adhesion to Equation 1 entails using $A_0(\eta_1)$ to compute $A_1(\eta_1)$, for instance. Likewise, $A_0(\eta_2)$ and $A_1(\eta_2)$ would be necessary to compute $A_2(\eta_2)$, and this scheme would repeat itself for every iteration.

We now introduce a formalism which shows the approximations we use are actually derivatives taken alongside the effective trajectory of the ascent.



Difference of cumulated regret between the ML and SG

Difference of cumulated regret between the ML and SVRG

Difference of cumulated regret between the ML and SG/AG

Difference of cumulated regret between the ML and SVRG/AG

Figure 6: Difference between the cumulated regrets of the algorithm and of the ML for a 50-dimensional linear regression model for several algorithms and several η_0's

It will also allow us to devise a new algorithm. It will, however, not account for the approximation of the Hessian.

To this avail, let us parameterise stochastic gradient algorithms by a sequence of step sizes

$$\boldsymbol{\eta} = (\eta_0, \eta_1, \ldots) \tag{27}$$

such that at iteration t, the update equation for θ_t becomes:

$$\theta_{t+1} = \theta_t + \frac{\eta_{t+1}}{f(t)} \partial_\theta \ell_t(\theta_t). \tag{28}$$

4.1 The loss as a function of step size: extension of the formalism

Consider the space $\mathcal{S}$ of infinite real sequences

$$\boldsymbol{\eta} = (\eta_0, \eta_1, \eta_2, \ldots) \tag{29}$$

We expand the T_t operators defined in Section 2 to similar ones defined on $\mathcal{S}$, with the same notation. Namely, define $T_0(\boldsymbol{\eta}) = \theta_0$ and, for $t > 0$,

$$T_t : \boldsymbol{\eta} \in \mathcal{S} \mapsto T_t(\boldsymbol{\eta}) \in \mathbb{R} \tag{30}$$

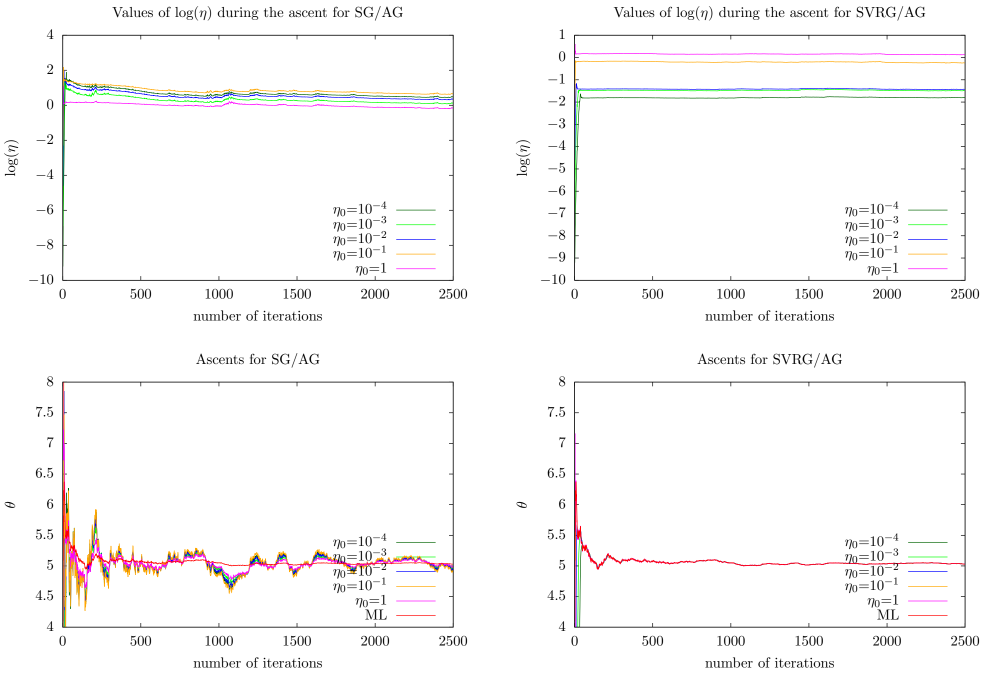


Figure 7: Evolution of $\log(\eta_t)$ in regard of the corresponding ascents for a Gaussian model in one dimension for *SG* and *SVRG* with LLR and several η_0's

where θ_t has been obtained thanks to t iterations of (28). T_t is a regular function of $\boldsymbol{\eta}$, as the computations only involve

$$\eta_0, \eta_1, \ldots, \eta_t, \tag{31}$$

and so take place in finite-dimensional spaces. This will apply in all the computations below. As before, we work on a space we call $\log(\mathcal{S})$, the image of $\mathcal{S}$ by the mapping

$$\boldsymbol{\eta} = (\eta_t)_{t\geq 0} \in \mathcal{S} \mapsto \log(\boldsymbol{\eta}) = (\log \eta_t)_{t\geq 0}, \tag{32}$$

but we do not change notation for the functions $\boldsymbol{\eta} \mapsto T_t(\boldsymbol{\eta})$, as in Section 2.

4.2 The update of the step size in the SG/SG algorithm as a gradient ascent

We now prove that in SG/SG, when the Hessian is used without approximations, the step size η_t indeed follows a gradient ascent scheme.

Proposition 1. *Let*

$$(\theta_t)_{t\geq 0}\,, \qquad \boldsymbol{\eta} = (\eta_t)_{t\geq 0} \tag{33}$$

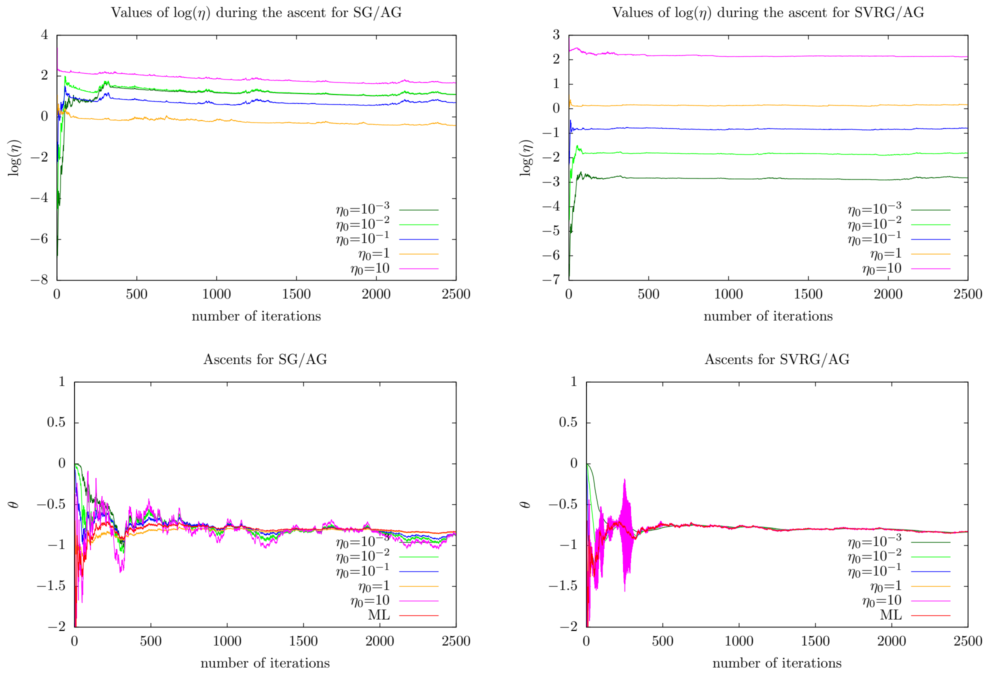


Figure 8: Evolution of $\log(\eta_t)$ in regard of the corresponding ascents for a Bernoulli model in one dimension for SG and SVRG with LLR and several η_0's

be the sequences of parameters and step-sizes obtained with the SG/SG algorithm, where the Hessian is not approximated: this is Algorithm 2 where the update on h_t is replaced with

$$h_{t+1} = h_t + \frac{\eta}{\mu_t}\partial_\theta \ell_t(\theta_t) + \frac{\eta}{\mu_t}\partial^2_\theta \ell_t(\theta_t) \cdot h_t. \tag{34}$$

Define e in the tangent plane of $\log(\mathcal{S})$ at $\log(\boldsymbol{\eta})$ by

$$e_t = 1, \qquad t \geq 0. \tag{35}$$

Then, for all $t \geq 0$,

$$\log \eta_{t+1} = \log \eta_t + \frac{1}{\mu_t}\frac{\partial}{\partial e}\ell_t(T_t(\boldsymbol{\eta})). \tag{36}$$

The proof lies in Appendix C.2.1.

4.3 A new algorithm, using a notion of "memory" borrowed from [SZL13]

We would now like to compute the change in η implied by a small modification of all previous coordinates η_s for s less than the current time t, but to

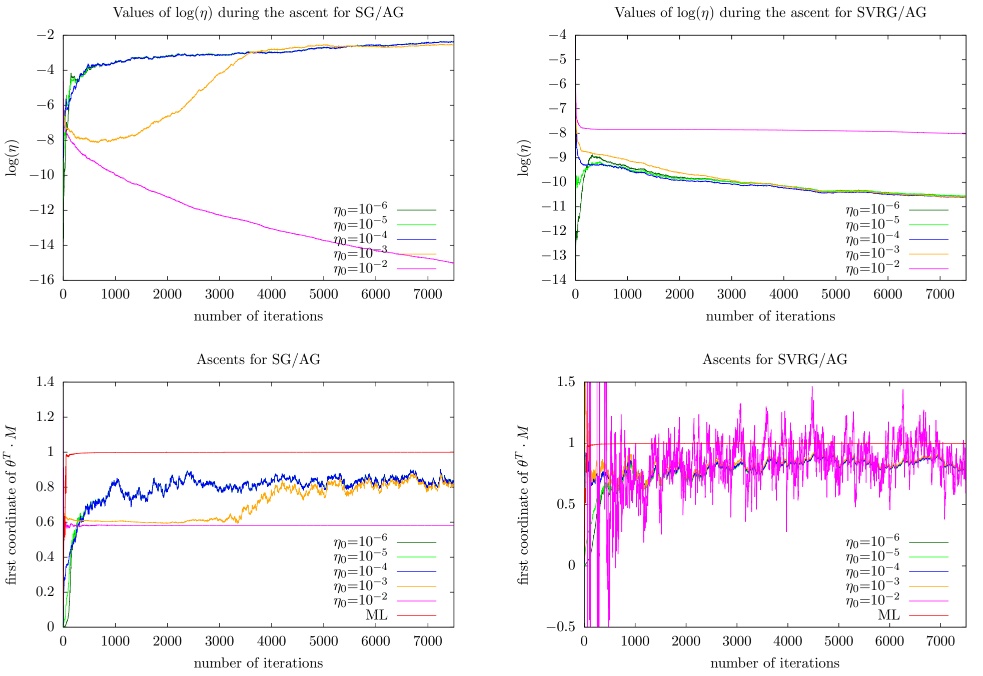


Figure 9: Evolution of $\log(\eta_t)$ in regard of the corresponding ascents for a 50-dimensional linear regression model for SG and SVRG with LLR and several η_0's

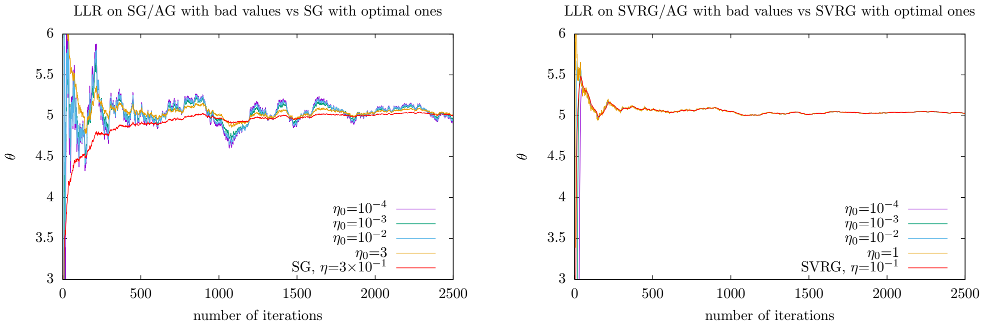


Figure 10: Trajectories of the ascents for a Gaussian model in one dimension for LLR algorithms with poor η_0's and original algorithms with empirically optimal η's

compute the modification differently according to whether the coordinate s is "outdated" or not. To do it, we use the quantity τ_t defined in Section 4.2 of [SZL13] as the "number of samples in recent memory". We want to discard the old η's and keep the recent ones. Therefore, at each time t, we

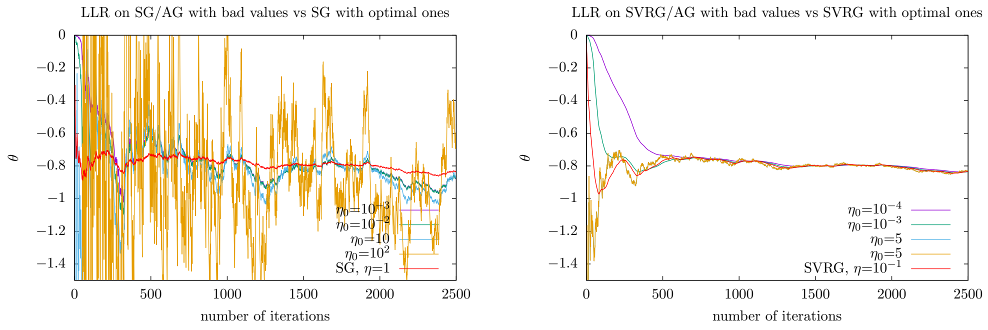


Figure 11: Trajectories of the ascents for a Bernoulli model for LLR algorithms with poor η_0's and original algorithms with empirically optimal η's

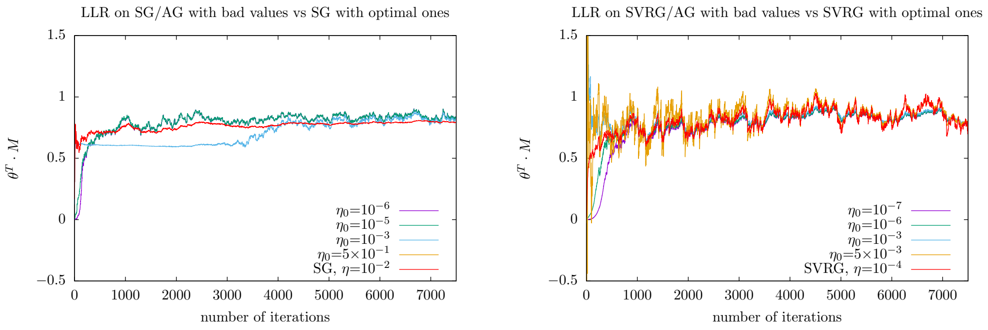


Figure 12: Trajectories of the ascents for a 50-dimensional linear regression model for LLR algorithms with poor η_0's and original algorithms with empirically optimal η's

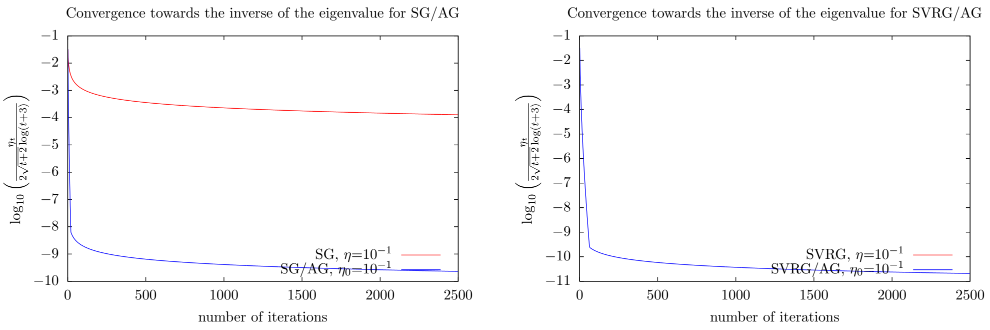


Figure 13: Evolution of $\log_{10}\left(\frac{\eta_t}{2\sqrt{t+2}\log(t+3)}\right)$ for a quadratic deterministic one-dimensional model for SG, SVRG and their LLR versions

compute

$$\gamma_t = \exp(-1/\tau_t). \tag{37}$$

Choose $\boldsymbol{\eta} \in \log(\mathcal{S})$, and consider the vector in the tangent plane to $\log(\mathcal{S})$ at $\boldsymbol{\eta}$:

$$e_t^j = \begin{cases} \prod_{k=j}^{t} \gamma_j, & j \leq t \\ 0, & j \geq t+1. \end{cases} \tag{38}$$

To run an algorithm using the e_t's instead of e as before, all we need to compute again is the formula for the update of the derivative below:

$$\mathcal{H}_t := \frac{\partial}{\partial e_t} T_t(\boldsymbol{\eta}). \tag{39}$$

$\mathcal{H}_t$ may indeed be computed, thanks to the following result.

Proposition 2. *The update equation of $\mathcal{H}_t$ is:*

$$\mathcal{H}_{t+1} = \gamma_{t+1}\mathcal{H}_t + \gamma_{t+1}\frac{\eta_{t+1}}{f(t)}\partial_\theta \ell_t(T_t(\eta^t)) + \gamma_{t+1}\frac{\eta_{t+1}}{f(t)}\partial_T^2 \ell_t(T_t(\boldsymbol{\eta})) \cdot \mathcal{H}_t. \tag{40}$$

The proof lies in Section C.2.2.

Acknowledgements

The first author would like to thank Gaetan Marceau-Caron for his advice on programming, and Jérémy Bensadon for crucial help with LaTeX.

References

[Ama98] Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Comput.*, 10:251–276, February 1998.

[Bot10] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[DBLJ14] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. 2014.

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[JZ13] Rie Johnson and Tong Zhang. Accelerating Stochastic Gradient Descent using Predictive Variance Reduction. 2013.

[KC78] Harold J. Kushner and Dean S. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*, volume 26 of *Applied Mathematical Sciences*. Springer-Verlag, New York Heidelberg Berlin, 1978.

[MDA15] Douglas Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hypermarameter optimization through reversible learning. 2015.

[RM51] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[SLRB13] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. Technical Report 00860051, HAL, 2013.

[SZL13] Tom Schaul, Sixin Zhang, and Yann LeCun. No More Pesky Learning Rates. pages 343–351. JMLR, 2013.

A LLR applied to the Stochastic Variance Reduced Gradient

The Stochastic Variance Reduced Gradient (SVRG) was introduced by Johnson and Zhang in [JZ13]. We define here a version intended for online use.

Algorithm 4 (SVRG online)**.** *We maintain $\theta_t, \theta^b \in \Theta$ (current parameter and base parameter) and $s_t^b \in T_{\simeq\theta_t}\Theta$ (sum of the gradients of the ℓ_s computed at θ_s up to time t).*

θ is set to θ_0 and θ^b along s^b to 0.

The update equations read:

$$\begin{cases} s_{t+1}^b = s_t^b + \partial_\theta \ell_t(\theta^b) \\ \theta_{t+1} = \theta_t + \eta \left(\partial_\theta \ell_t(\theta_t) - \partial_\theta \ell_t(\theta^b) + \dfrac{s_{t+1}^b}{t+1} \right). \end{cases} \tag{41}$$

We now present the LLR version, obtained by updating the η of SVRG thanks to an SG ascent. We call this algorithm "SVRG/SG".

Algorithm 5 (SVRG/AG)**.** *We maintain $\theta_t, \theta^b \in \Theta$ (current parameter and base parameter), $\eta_t \in \mathbb{R}$ (current step size), $s_t^b \in T_{\simeq\theta_t}\Theta$ (sum of the gradients of the ℓ_s computed at θ_s up to time t), $h_t \in T_{\theta_t}\Theta$ (approximation of the derivative of T_t with respect to $\log(\eta)$ at η_t) and the real numbers n_t (average of the squared norms of the λ_s defined below) and d_t (renormalising factor for the computation of n_t).*

θ is set to θ_0, the other variables are set to 0.

At each time t, we compute $\mu_t \in \mathbb{R}$ (a rate used in several updates), and $\lambda_t \in \mathbb{R}$ (the approximate derivative of $\ell_t \circ \theta_t$ with respect to $\log(\eta)$ at η_t).

The update equations read:

$$\begin{cases} \mu_t = \sqrt{t+2}\log(t+3) \\ \lambda_t = \partial_\theta \ell_t(\theta_t) \cdot h_t \\ d_{t+1} = \left(1 - \dfrac{1}{\mu_t}\right) d_t + \dfrac{1}{\mu_t} \\ n_{t+1}^2 = \left(\left(1 - \dfrac{1}{\mu_t}\right) n_t^2 + \dfrac{1}{\mu_t}\lambda_t^2\right) d_{t+1}^{-1} \\ \eta_{t+1} = \eta_t \exp\left(\dfrac{1}{\mu_t}\dfrac{\lambda_t}{n_{t+1}}\right) \\ s_{t+1}^b = s_t^b + \partial_\theta \ell_t(\theta^b) \\ h_{t+1} = h_t + \eta_{t+1}\left(\partial_\theta \ell_t(\theta_t + h_t) - \partial_\theta \ell_t(\theta^b) + \dfrac{s_{t+1}^b}{t+1}\right) \\ \theta_{t+1} = \theta_t + \eta_{t+1}\left(\partial_\theta \ell_t(\theta_t) - \partial_\theta \ell_t(\theta^b) + \dfrac{s_{t+1}^b}{t+1}\right). \end{cases} \tag{42}$$

B LLR applied to a general stochastic gradient algorithm

Let Θ and H be two spaces. Θ is the space of parameters, H is that of hyperparameters. In this section, a parameter potentially means a tuple of parameters in the sense of other sections. For instance, in SVRG/SG online, we would call a parameter the couple

$$\left(\theta_t, \theta^b\right). \tag{43}$$

Likewise, in the same algorithm, we would call a hyperparameter the couple

$$\left(\eta_t, h_t\right). \tag{44}$$

Let

$$\begin{aligned} F: \quad & \Theta \times H \to \Theta \\ & (\theta, \eta) \mapsto F(\theta, \eta). \end{aligned} \tag{45}$$

be differentiable with respect to both variables. We consider the algorithm:

$$\theta_{t+1} = F(\theta_t, \eta_t). \tag{46}$$

Let us present its LLR version. We call it GEN/SG, GEN standing for "general".

Algorithm 6 (GEN/SG)**.** *We maintain $\theta_t \in \Theta$ (current parameter), $\eta_t \in H$ (current hyperparameter), $h_t \in T_{\theta_t}\Theta$ (approximation of the derivative of T_t in the direction of $e \in T_{\eta_t}H$).*

θ and η are set to user-defined values.

The update equations read:

$$\begin{cases} \eta_{t+1} = \eta_t + \alpha \partial_\theta \ell_t(\theta_t) \cdot h_t \\ h_{t+1} = \partial_\theta F(\theta_t, \eta_t) \cdot h_t + \partial_\eta F(\theta_t, \eta_t) \cdot \dfrac{\partial}{\partial e}\eta_t \\ \theta_{t+1} = F\left(\theta_t, \eta_{t+1}\right). \end{cases} \tag{47}$$

C Computations

C.1 Computations for Section 2: proof of Fact 1

Proof. θ_0 is fixed, so $A_0(\eta) = 0$. Let $t \geq 0$. We differentiate (5) with respect to $\log(\eta)$, to obtain:

$$\frac{\partial}{\partial \log \eta} T_{t+1}(\eta) = \frac{\partial}{\partial \log \eta} T_t(\eta) + \frac{\eta}{f(t)}\, \partial_\theta \ell_t(\theta_t) + \frac{\eta}{f(t)} \partial_\theta^2 \ell_t(\theta_t(\eta)) \cdot \frac{\partial}{\partial \log \eta} T_t(\eta), \tag{48}$$

which concludes the proof. □

C.2 Computations for Section 4

C.2.1 Computations for Section 4.2: proof of Proposition 1

To prove Proposition 1, we use the following three lemmas. The first two are technical, and are used in the proof of the third one, which provides an update formula for the derivative appearing in the statement of the proposition. We may have proceeded without these, as in the proof of Fact 1, but they allow the approach to be more generic.

Lemma 2. *Let*

$$\begin{array}{rlcl} F_t : & \Theta \times \mathbb{R} & \to \Theta & \\ & (\theta, \eta) & \mapsto & F_t(\theta, \eta) = \theta + \frac{\eta}{f(t)} \partial_\theta \ell_t(\theta). \end{array} \tag{49}$$

Then,

$$\frac{\partial}{\partial \theta} F_t(\theta, \eta) = \mathrm{Id} + \frac{\eta}{f(t)} \partial_\theta^2 \ell_t(\theta) \tag{50}$$

and

$$\frac{\partial}{\partial \eta} F_t(\theta, \eta) = \frac{1}{f(t)} \partial_\theta \ell_t(\theta). \tag{51}$$

Id *is the identity on the tangent plane to* Θ *in* θ*.*

Lemma 3. *Let*

$$\begin{array}{rlcl} V_t : & \mathcal{S} & \to \Theta \times \mathbb{R} & \\ & \boldsymbol{\eta} & \mapsto & V_t(\boldsymbol{\eta}) = (\theta_t(\eta), \eta_{t+1}). \end{array} \tag{52}$$

Consider $\log(\boldsymbol{\eta}) \in \log(\mathcal{S})$*, and any vector* e *tangent to* $\log(\mathcal{S})$ *at this point. Then the directional derivative of*

$$\begin{array}{rlcl} F_t \circ V_t : & \mathcal{S} & \to \Theta & \\ & \boldsymbol{\eta} & \mapsto & F_t(V_t(\boldsymbol{\eta})) = T_t(\boldsymbol{\eta}) + \frac{\eta_{t+1}}{f(t)} \partial_\theta \ell_t(T_t(\boldsymbol{\eta})) \end{array} \tag{53}$$

at the point $\log(\boldsymbol{\eta})$ *and in the direction* e *is*

$$\frac{\partial}{\partial e} F_t \circ V_t(\boldsymbol{\eta}) = \frac{\partial}{\partial e} T_t(\eta) + \frac{\partial}{\partial e} \eta_{t+1} \frac{1}{f(t)} \partial_\theta \ell_t(T_t(\boldsymbol{\eta})) + \frac{\eta_{t+1}}{f(t)} \partial_\theta^2 \ell_t(T_t(\boldsymbol{\eta})) \cdot \frac{\partial}{\partial e} T_t(\eta). \tag{54}$$

We may then prove the following lemma.

Lemma 4. *Define*

$$\mathcal{H}_t = \frac{\partial}{\partial e} T_t(\boldsymbol{\eta}). \tag{55}$$

Then for all $t \geq 0$,

$$\mathcal{H}_{t+1} = \mathcal{H}_t + \frac{\eta_{t+1}}{f(t)} \partial_\theta \ell_t(T_t(\boldsymbol{\eta})) + \frac{\eta_{t+1}}{f(t)} \partial_\theta^2 \ell_t(T_t(\boldsymbol{\eta})) \cdot \mathcal{H}_t. \tag{56}$$

Proof. The update equation of $T_t(\boldsymbol{\eta})$, (28), is such that:

$$T_{t+1}(\boldsymbol{\eta}) = T_t(\boldsymbol{\eta}) + \frac{\eta_{t+1}}{f(t)} \partial_\theta \ell_t(T_t(\boldsymbol{\eta})) = F_t \circ V_t(\boldsymbol{\eta}). \tag{57}$$

From the above and Lemma 3,

$$\frac{\partial}{\partial e} T_{t+1}(\eta) = \frac{\partial}{\partial e} T_t(\boldsymbol{\eta}) + \frac{\partial}{\partial e} \eta_{t+1} \frac{1}{f(t)} \partial_\theta \ell_t(T_t(\boldsymbol{\eta})) + \frac{\eta_{t+1}}{f(t)} \partial_\theta^2 \ell_t(T_t(\boldsymbol{\eta})) \cdot \frac{\partial}{\partial e} T_t(\boldsymbol{\eta}). \tag{58}$$

Now,

$$\frac{\partial}{\partial e} \eta_{t+1} = \eta_{t+1}, \tag{59}$$

which concludes the proof. □

Finally, we prove Proposition 1.

Proof of Proposition 1. It is sufficient to prove that, for all $t \geq 0$,

$$\frac{\partial}{\partial e} \ell_t(T_t(\boldsymbol{\eta})) = \partial_\theta \ell_t(\theta_t) \cdot h_t, \tag{60}$$

that is,

$$\partial_\theta \ell_t(T_t(\boldsymbol{\eta})) \cdot \mathcal{H}_t = \partial_\theta \ell_t(\theta_t) \cdot h_t. \tag{61}$$

Therefore, it is sufficient to prove that, for all $t \geq 0$, $T_t(\boldsymbol{\eta}) = \theta_t$ and $\mathcal{H}_t = h_t$. $T_0(\eta) = \theta_0$ by construction and, since θ_0 does not depend on $\boldsymbol{\eta}$, $\mathcal{H}_0 = 0 = h_0$. Assuming the results hold up to iteration t, it is straighforward that $T_{t+1}(\boldsymbol{\eta}) = \theta_{t+1}$, since for all $s \leq t$, $T_s(\boldsymbol{\eta}) = \theta_s$. Therefore, thanks to Lemma 4, $\mathcal{H}_t$ and h_t have the same update, so that $\mathcal{H}_{t+1} = h_{t+1}$, which concludes the proof. □

C.2.2 Computations for Section 4.3: proof of Proposition 2

Proof. Thanks to (58) in Lemma 3,

$$\begin{aligned} \frac{\partial}{\partial e_{t+1}} T_{t+1}(\eta) = \frac{\partial}{\partial e_{t+1}} T_t(\boldsymbol{\eta}) &+ \frac{\partial}{\partial e_{t+1}} \eta_{t+1} \frac{1}{f(t)} \partial_\theta \ell_t(T_t(\boldsymbol{\eta})) \\ &+ \frac{\eta_{t+1}}{f(t)} \partial_\theta^2 \ell_t(T_t(\boldsymbol{\eta})) \cdot \frac{\partial}{\partial e_{t+1}} T_t(\boldsymbol{\eta}), \end{aligned} \tag{62}$$

that is:

$$\mathcal{H}_{t+1} = \frac{\partial}{\partial e_{t+1}} T_t(\boldsymbol{\eta}) + \frac{\partial}{\partial e_{t+1}} \eta_{t+1} \frac{1}{f(t)} \partial_\theta \ell_t(T_t(\boldsymbol{\eta})) + \frac{\eta_{t+1}}{f(t)} \partial_\theta^2 \ell_t(T_t(\boldsymbol{\eta})) \cdot \frac{\partial}{\partial e_{t+1}} T_t(\boldsymbol{\eta}). \tag{63}$$

We first prove:

$$\frac{\partial}{\partial e_{t+1}} T_t(\boldsymbol{\eta}) = \gamma_{t+1} \frac{\partial}{\partial e_t} T_t(\eta). \tag{64}$$

Define $(f_j)_{j \geq 0}$ the canonical basis of the tangent plane to $\log(\mathcal{S})$ at $\boldsymbol{\eta}$. Then,

$$e_{t+1} = \gamma_{t+1}(e_t + f_{t+1}). \tag{65}$$

Therefore,

$$\begin{aligned} \frac{\partial}{\partial e_{t+1}} T_t(\boldsymbol{\eta}) &= \frac{\partial}{\partial e_{t+1}} T_t(\boldsymbol{\eta}) \\ &= \gamma_{t+1} \frac{\partial}{\partial e_t} T_t(\eta^t) + \frac{\partial}{\partial f_{t+1}} T_t(\eta^t) \\ &= \gamma_{t+1} \frac{\partial}{\partial e_t} T_t(\eta^t) \end{aligned} \tag{66}$$

because the last term is 0. Therefore,

$$\frac{\partial}{\partial e_{t+1}} T_t(\boldsymbol{\eta}) = \gamma_{t+1} \mathcal{H}_t. \tag{67}$$

Then, thanks to (58),

$$\frac{\partial}{\partial e_{t+1}} \eta_{t+1} = \gamma_{t+1} \eta_{t+1}, \tag{68}$$

which is true since

$$\frac{\partial}{\partial e_{t+1}} \eta_{t+1} = \gamma_{t+1} \frac{\partial}{\partial f_{t+1}} \eta_{t+1} = \gamma_{t+1} \eta_{t+1}, \tag{69}$$

and concludes the proof. □