Second-order step-size tuning of SGD for non-convex optimization

Camille Castera, Cédric Févotte, Jérôme Bolte, Edouard Pauwels

Second-order step-size tuning of SGD for non-convex optimization

Camille Castera*
IRIT, Université de Toulouse
CNRS, France

Jérôme Bolte†
Toulouse School of Economics
Université de Toulouse, France

Cédric Févotte†
IRIT, Université de Toulouse
CNRS, France

Edouard Pauwels†
IRIT, Université de Toulouse
CNRS, France
DEEL, IRT Saint Exupery

Abstract

In view of a direct and simple improvement of vanilla SGD, this paper presents a fine-tuning of its step-sizes in the mini-batch case. For doing so, one estimates curvature, based on a local quadratic model and using only noisy gradient approximations. One obtains a new stochastic first-order method (Step-Tuned SGD) which can be seen as a stochastic version of the classical Barzilai-Borwein method. Our theoretical results ensure almost sure convergence to the critical set and we provide convergence rates. Experiments on deep residual network training illustrate the favorable properties of our approach. For such networks we observe, during training, both a sudden drop of the loss and an improvement of test accuracy at medium stages, yielding better results than SGD, RMSprop, or ADAM.

1 Introduction

In the recent years, machine learning has generated a growing need for methods to solve non-convex optimization problems. In particular, the training of deep neural networks (DNNs) has received tremendous attention. Designing methods for this purpose is particularly difficult as one deals with both expensive function evaluations and limited storage capacities. This explains why stochastic gradient descent (SGD) remains the central algorithm in deep learning (DL). It consists in the iterative scheme,

$$\theta_{k+1} = \theta_k - \gamma_k \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k), \tag{1}$$

where $\mathcal{J}$ is the function to minimize (usually the empirical loss) parameterized by $\theta \in \mathbb{R}^P$ (the weights of the DNN), $\nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)$ is a stochastic estimation of the gradient of $\mathcal{J}$ (randomness being related to the sub-sampled mini-batch B_k), and $\gamma_k > 0$ is a step-size whose choice is critical in terms of empirical performance.

In order to improve the basic SGD method, a common practice is to use adaptive methods (Duchi et al., 2011; Tieleman and Hinton, 2012; Kingma and Ba, 2015). They act as preconditioners, reducing the importance of the choice of the sequence of step-sizes $(\gamma_k)_{k\in\mathbb{N}}$. This paper focuses instead exclusively on the step-size issue: how can we tune steps in a stochastic context by taking advantage of curvature information for non-convex landscapes?

Our starting point for step-size tuning is an infinitesimal second-order variational model along the gradient direction. The infinitesimal feature is particularly relevant in DL since small steps constitute standard practice in training due to sub-sampling noise. Second-order information is approximated with a first-order quantities using finite differences. In deterministic (full-batch) setting, our method corresponds to a non-convex version of the Barzilai-Borwein (BB) method (Barzilai and Borwein, 1988; Dai et al., 2002; Xiao et al., 2010; Biglari and Solimanpur, 2013) and is somehow a discrete non-convex adaption of the continuous gradient system in Alvarez and Cabot (2004). It is also close to earlier work Raydan (1997), with the major difference that our algorithm is supported by a variational model. This is essential to generalize the method to accommodate noisy gradients, providing a convexity test similar to those in Babaie-Kafaki and Fatemi (2013); Curtis and Guo (2016).

Our main contribution is a step-size tuning method for stochastic gradient algorithms, which is built on a strong geometrical principle: step-sizes are deduced from a carefully derived discrete approximation of a

*Corresponding author camille.castera@irit.fr
†Last three authors are listed in alphabetical order

curvature term of the expected loss. We provide general convergence guarantees to critical points and rates of convergence. Extensive computations on DL problems show that our method is particularly successful for residual networks (He et al., 2016). In that case, we observe a surprising phenomenon: in the early training stage the method shows similar performances to standard DL algorithms (SGD, ADAM, RMSprop), then at medium stage, we observe simultaneously a sudden drop of the training loss and a notable increase in test accuracy.

To summarize, our contributions are as follows:

- Exploit the vanishing step-size nature of DL training to use infinitesimal second-order optimization for fine step-size tuning.
- Use our geometrical perspective to discretize and adapt the method to noisy gradients despite strong non-linearities.
- Obtain general rigorous asymptotic theoretical guarantees (see Theorem 4.1 and Corollary 4.2).
- Show that our method has remarkable practical properties, in particular when training residual networks in DL, for which one observes an advantageous "drop down" of the loss during the training phase.

Structure of the paper. A preliminary deterministic (i.e., full-batch) algorithm is derived in Section 3.1. We then build a stochastic mini-batch variant in Section 3.2, which is our core contribution. Theoretical results are stated in Section 4 and DL experiments are conducted in Section 5.

2 Related work

Methods using second-order information for non-convex optimization have been actively studied in the last years, both for deterministic and stochastic applications, see, e.g., Royer and Wright (2018); Carmon et al. (2017); Allen-Zhu (2018); Krishnan et al. (2018); Martens and Grosse (2015); Liu and Yang (2017); Curtis and Robinson (2019).

BB-like methods are very sensitive to noisy gradient estimates. Most existing stochastic BB algorithms (Tan et al., 2016; Liang et al., 2019; Robles-Kelly and Nazari, 2019) overcome this issue with stabilization methods in the style of SVRG (Johnson and Zhang, 2013), which allows to prescribe a new step-size at every epoch only (i.e., after a full pass over the data). Doing so, one cannot capture variations of curvature within a full epoch, and one is limited to using absolute values to prevent negative step-sizes caused by non-convexity. One the contrary, our stochastic approximation method can adapt to local curvature every two iterations.

There are few techniques to analyze stochastic methods in non-convex settings. An important category is the ODE machinery used for SGD (Davis et al., 2020; Bolte and Pauwels, 2020), ADAM (Barakat and Bianchi, 2018) and INDIAN (Castera et al., 2019). In this paper, we use instead direct and more traditional arguments, such as in Li and Orabona (2019) in the context of DL.

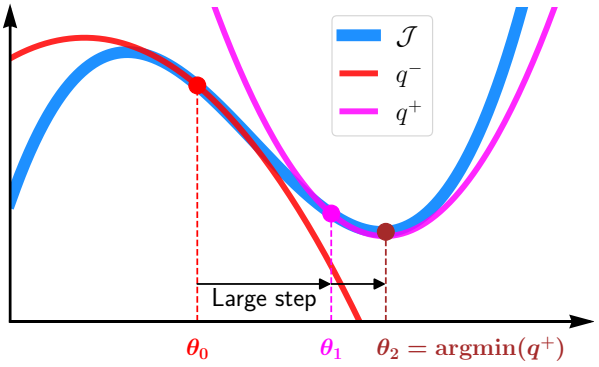

Figure 1: Illustration of negative and positive curvature steps. The function q^- represents the variational model at θ_0, with negative curvature. Concavity suggests to take a large step to reach θ_1. Then, at θ_1, the variational model q^+ has positive curvature and can be minimized to obtain θ_2.

3 Design of the algorithm

We first build a preliminary algorithm based upon a simple second-order variational model. We then adapt this algorithm to address mini-batch stochastic approximations.

3.1 Deterministic full-batch algorithm

Second-order infinitesimal step-size tuning. Assume that $\mathcal{J}:\mathbb{R}^P \mapsto \mathbb{R}$ is a twice-differentiable function. Let $\theta \in \mathbb{R}^P$. Given an update direction $d \in \mathbb{R}^P$, a natural strategy is to choose $\gamma \in \mathbb{R}$ that minimizes $\mathcal{J}(\theta+\gamma d)$. Let us approximate $\gamma \mapsto \mathcal{J}(\theta+\gamma d)$ around 0 with a Taylor expansion,

$$q_d(\gamma) \stackrel{\text{def}}{=} \mathcal{J}(\theta) + \gamma\langle \nabla\mathcal{J}(\theta), d\rangle + \frac{\gamma^2}{2}\langle \nabla^2\mathcal{J}(\theta)d, d\rangle. \quad (2)$$

If the curvature term $\langle \nabla^2\mathcal{J}(\theta)d, d\rangle$ is positive, then q_d has a unique minimizer at,

$$\gamma^\star = -\frac{\langle \nabla\mathcal{J}(\theta), d\rangle}{\langle \nabla^2\mathcal{J}(\theta)d, d\rangle}. \quad (3)$$

On the contrary when $\langle \nabla^2 \mathcal{J}(\theta) d, d \rangle \leq 0$, the infinitesimal model q_d is concave which suggests to take a large step-size. These considerations are illustrated on Figure 1.

Tuning gradient descent. In order to tune gradient descent we choose the direction $d = -\nabla \mathcal{J}(\theta)$ which gives,

$$\gamma(\theta) \stackrel{\text{def}}{=} \frac{\|\nabla \mathcal{J}(\theta)\|^2}{\langle \nabla^2 \mathcal{J}(\theta) \nabla \mathcal{J}(\theta), \nabla \mathcal{J}(\theta) \rangle}. \tag{4}$$

According to our previous considerations, an ideal iterative process $\theta_{k+1} = \theta_k - \gamma_k \nabla \mathcal{J}(\theta_k)$ would use $\gamma_k = \gamma(\theta_k)$ when $\gamma(\theta_k) > 0$. But for computational reasons and discretization purposes, we shall rather seek a step-size γ_k such that, $\gamma_k \simeq \gamma(\theta_{k-1})$ (again when $\gamma(\theta_{k-1}) > 0$). Let us assume that, for $k \geq 1$, $\theta_{k-1}, \gamma_{k-1}$ are known and let us approximate the quantity,

$$\gamma(\theta_{k-1}) = \frac{\|\nabla \mathcal{J}(\theta_{k-1})\|^2}{\langle \nabla^2 \mathcal{J}(\theta_{k-1}) \nabla \mathcal{J}(\theta_{k-1}), \nabla \mathcal{J}(\theta_{k-1}) \rangle}, \tag{5}$$

using only first-order objects. We rely on two identities,

$$\Delta\theta_k \stackrel{\text{def}}{=} \theta_k - \theta_{k-1} = -\gamma_{k-1} \nabla \mathcal{J}(\theta_{k-1}), \tag{6}$$

$$\Delta g_k \stackrel{\text{def}}{=} \nabla \mathcal{J}(\theta_k) - \nabla \mathcal{J}(\theta_{k-1}) \simeq -\gamma_{k-1} \mathcal{C}_{\mathcal{J}}(\theta_{k-1}), \tag{7}$$

where $\mathcal{C}_{\mathcal{J}}(\theta) \stackrel{\text{def}}{=} \nabla^2 \mathcal{J}(\theta) \nabla \mathcal{J}(\theta)$ and (7) is obtained by Taylor's formula. Combining the above, we are led to consider the following step-size,

$$\gamma_k = \begin{cases} \frac{\|\Delta\theta_k\|^2}{\langle \Delta\theta_k, \Delta g_k \rangle} & \text{if } \langle \Delta\theta_k, \Delta g_k \rangle > 0 \\ \nu & \text{otherwise} \end{cases}, \tag{8}$$

where $\nu > 0$ is an hyperparameter of the algorithm representing the large step-sizes to use in locally concave regions.

The resulting full-batch non-convex optimization method is Algorithm 1, in which α is a so-called *learning rate* or *scaling factor*. This algorithm is present in the literature under subtle variants (Raydan, 1997; Dai et al., 2002; Xiao et al., 2010; Biglari and Solimanpur, 2013). It may be seen as a non-convex version of the BB method (designed for strongly convex functions). In the classical optimization literature it is often combined with line-search procedures which is impossible in our large-scale DL context. The interest of our variational viewpoint is the characterization of the underlying geometrical mechanism, which is key in designing an efficient stochastic version of Algorithm 1 in Section 3.2.

Algorithm 1 Full-batch preliminary algorithm

Input: $\alpha > 0$, $\nu > 0$
Initialize $\theta_0 \in \mathbb{R}^P$
$\theta_1 = \theta_0 - \alpha \nabla \mathcal{J}(\theta_0)$
for $k = 1, \ldots$ **do**
  $\Delta g_k = \nabla \mathcal{J}(\theta_k) - \nabla \mathcal{J}(\theta_{k-1})$
  $\Delta\theta_k = \theta_k - \theta_{k-1}$
  if $\langle \Delta g_k, \Delta\theta_k \rangle > 0$ **then**
    $\gamma_k = \frac{\|\Delta\theta_k\|^2}{\langle \Delta g_k, \Delta\theta_k \rangle}$
  else
    $\gamma_k = \nu$
  end if
  $\theta_{k+1} = \theta_k - \alpha \gamma_k \nabla \mathcal{J}(\theta_k)$
end for

Illustrative experiment. Before presenting the stochastic version, we illustrate the interest of exploiting negative curvature through the "large-step" parameter ν with a synthetic experiment inspired from Carmon et al. (2017). We apply Algorithm 1 to a non-convex regression problem of the form $\min_{\theta \in \mathbb{R}^P} \phi(A\theta - b)$ where ϕ is a non-convex real-valued function (see Section C of the Supplementary). We compare Algorithm 1 with the current methods à la BB where absolute values are used when the step-size is negative (see, e.g., Tan et al. (2016); Liang et al. (2019) in stochastic settings) and with Armijo's line-search gradient method. As shown on Figure 2, Algorithm 1 efficiently exploits local curvature and converges much faster than other methods.

3.2 Stochastic mini-batch algorithm

We wish to adapt Algorithm 1 in cases where gradients can only be approximated through mini-batch sub-sampling. This is necessary in particular for DL applications.

Mini-batch sub-sampling. We assume the following sum-structure of the loss function, for $N \in \mathbb{N}^\star$,

$$\mathcal{J}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \mathcal{J}_n(\theta), \tag{9}$$

where each $\mathcal{J}_n$ is a twice continuously-differentiable function. Given any fixed subset $\mathsf{B} \subset \{1, \ldots, N\}$, we define the following quantities for any $\theta \in \mathbb{R}^P$,

$$\mathcal{J}_{\mathsf{B}}(\theta) \stackrel{\text{def}}{=} \frac{1}{|\mathsf{B}|} \sum_{n \in \mathsf{B}} \mathcal{J}_n(\theta), \tag{10}$$

$$\nabla \mathcal{J}_{\mathsf{B}}(\theta) \stackrel{\text{def}}{=} \frac{1}{|\mathsf{B}|} \sum_{n \in \mathsf{B}} \nabla \mathcal{J}_n(\theta), \tag{11}$$

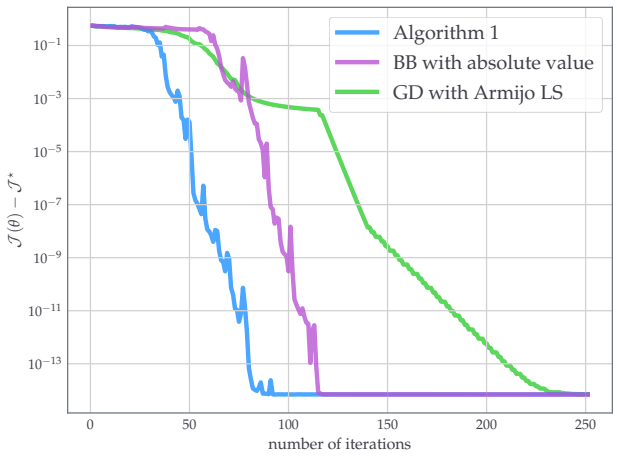

Figure 2: Values of the loss function $\mathcal{J}(\theta)$ against iterations (each corresponding to a gradient step) for the synthetic non-convex regression problem detailed in Section C of the Supplementary. The optimal value $\mathcal{J}^\star$ is unknown and is estimated by taking the best value obtained among all algorithms after 10^5 iterations.

where $|\mathsf{B}|$ denotes the number of elements of the set B. Throughout this paper we will consider independent copies of a random subset $\mathsf{S} \subset \{1, \dots, N\}$ referred to as mini-batch. The distribution of this subset is fixed throughout the paper and taken such that the expectation over the realization of S in (10) corresponds to the empirical expectation in (9). This is valid for example if S is taken uniformly at random over all possible subsets of fixed size. As a consequence, we have the identity $\mathcal{J} = \mathbb{E}_{\mathsf{S}}[\mathcal{J}_{\mathsf{S}}]$, where the expectation is taken over the random draw of S. This allows to interpret mini-batch sub-sampling as a stochastic approximation process since we also have $\nabla\mathcal{J} = \mathbb{E}_{\mathsf{S}}[\nabla\mathcal{J}_{\mathsf{S}}]$.

Our algorithm is of stochastic gradient type where stochasticity is related to the randomness of mini-batches. We start with an initialization $\theta_0 \in \mathbb{R}^P$, and a sequence of i.i.d. random mini-batches $(\mathsf{B}_k)_{k\in\mathbb{N}}$, whose common distribution is the same as S. The algorithm produces a random sequence of iterates $(\theta_k)_{k\in\mathbb{N}}$. For $k \in \mathbb{N}$, B_k is used to estimate an update direction $-\nabla\mathcal{J}_{\mathsf{B}_k}(\theta_k)$ which is used in place of $-\nabla\mathcal{J}(\theta_k)$ in the same way as gradient descent algorithm.

Second-order tuning of mini-batch SGD: Step-Tuned SGD. Our goal is to devise a step-size strategy, based on the variational ideas developed earlier and on the quantity $\mathcal{C}_{\mathcal{J}}$, in the context of mini-batch sub-sampling. First observe that for $\theta \in \mathbb{R}^P$,

$$\mathcal{C}_{\mathcal{J}}(\theta) = \nabla^2\mathcal{J}(\theta)\nabla\mathcal{J}(\theta) = \nabla\left(\frac{1}{2}\|\nabla\mathcal{J}(\theta)\|^2\right). \quad (12)$$

Algorithm 2 Step-Tuned SGD

Input: $\alpha > 0$, $\nu > 0$
Input: $\beta \in [0,1]$, $\tilde{m} > 0$, $\tilde{M} > 0$, $\delta \in (0.5, 1)$
Initialize $\theta_0 \in \mathbb{R}^P$, $G_{-1} = \mathbf{0}_P$, $\gamma_0 = 1$
Draw independent random mini-batches $(\mathsf{B}_k)_{k\in\mathbb{N}}$.
for $k = 0, 1, \dots$ **do**
  $\theta_{k+\frac{1}{2}} = \theta_k - \frac{\alpha}{(k+1)^\delta}\gamma_k\nabla\mathcal{J}_{\mathsf{B}_k}(\theta_k)$
  $\theta_{k+1} = \theta_{k+\frac{1}{2}} - \frac{\alpha}{(k+1)^\delta}\gamma_k\nabla\mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}})$
  $\Delta\theta_{\mathsf{B}_k} = \theta_{k+\frac{1}{2}} - \theta_k$
  $\Delta g_{\mathsf{B}_k} = \nabla\mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}}) - \nabla\mathcal{J}_{\mathsf{B}_k}(\theta_k)$
  $G_k = \beta G_{k-1} + (1-\beta)\Delta g_{\mathsf{B}_k}$
  $\hat{G}_k = G_k/(1-\beta^{k+1})$
  if $\langle \hat{G}_k, \Delta\theta_{\mathsf{B}_k}\rangle > 0$ **then**
    $\gamma_{k+1} = \frac{\|\Delta\theta_{\mathsf{B}_k}\|^2}{\langle \hat{G}_k, \Delta\theta_{\mathsf{B}_k}\rangle}$
  else
    $\gamma_{k+1} = \nu$
  end if
  $\gamma_{k+1} = \min(\max(\gamma_{k+1}, \tilde{m}), \tilde{M})$
end for

So rewriting $\mathcal{J}$ as an expectation,

$$\mathcal{C}_{\mathcal{J}}(\theta) = \nabla\left(\frac{1}{2}\|\mathbb{E}_{\mathsf{S}}\left[\nabla\mathcal{J}_{\mathsf{S}}(\theta)\right]\|^2\right), \quad (13)$$

where S denotes like in the previous paragraph a random subset of $\{1, \dots, N\}$, or mini-batch. This suggests the following estimator, for any subset B and $\theta \in \mathbb{R}^P$,

$$\begin{aligned}\mathcal{C}_{\mathcal{J}_{\mathsf{B}}}(\theta) &\stackrel{\text{def}}{=} \nabla\left(\frac{1}{2}\|\nabla\mathcal{J}_{\mathsf{B}}(\theta)\|^2\right)\\ &= \nabla^2\mathcal{J}_{\mathsf{B}}(\theta)\nabla\mathcal{J}_{\mathsf{B}}(\theta), \end{aligned} \quad (14)$$

to build an infinitesimal model as in (4).

Like in the deterministic case we approximate the new target (14) with a Taylor expansion of $\mathcal{J}_{\mathsf{B}}$ between two iterations. We obtain for any $\mathsf{B} \subset \{1, \dots, N\}$, $\theta \in \mathbb{R}^P$, and small $\gamma > 0$

$$-\gamma\mathcal{C}_{\mathcal{J}_{\mathsf{B}}}(\theta) \simeq \nabla\mathcal{J}_{\mathsf{B}}(\underbrace{\theta - \gamma\nabla\mathcal{J}_{\mathsf{B}}(\theta)}_{\text{next iterate}}) - \nabla\mathcal{J}_{\mathsf{B}}(\theta). \quad (15)$$

This suggests to use each mini-batch twice and compute a difference of gradients every two iterations.[1] We adopt the following convention, at iteration $k \in \mathbb{N}$, the random mini-batch B_k is used to compute a stochastic gradient, $\nabla\mathcal{J}_{\mathsf{B}_k}(\theta_k)$ and at iteration $k+\frac{1}{2}$ the same mini-batch is used to compute another stochastic gradient $\nabla\mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}})$, for a given $\theta_{k+\frac{1}{2}}$. Let us define,

$$\Delta g_{\mathsf{B}_k} \stackrel{\text{def}}{=} \nabla\mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}}) - \nabla\mathcal{J}_{\mathsf{B}_k}(\theta_k), \quad (16)$$

[1] There is also the possibility of computing additional estimates as Schraudolph et al. (2007) previously did for a stochastic BFGS algorithm, but this would double the computational cost.

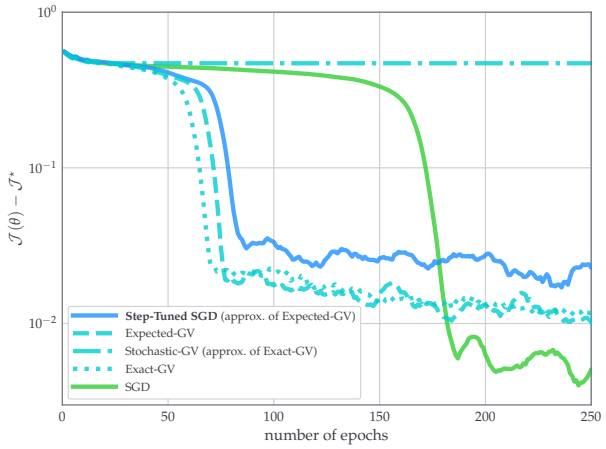

Figure 3: Values of the loss function against epochs for non-convex regression: heuristic methods (dashed lines) of Section 3.3 are compared with Step-Tuned SGD (plain blue). SGD serves as a reference to evidence the drop down effect.

thereby Δg_{B_k} forms an approximation of $-\gamma_k \mathcal{C}_{\mathcal{J}_{\mathsf{B}_k}}(\theta_k)$ that we can use to compute the next step-size γ_{k+1}. We define the difference between two iterates accordingly,

$$\Delta\theta_{\mathsf{B}_k} \overset{\text{def}}{=} \theta_{k+\frac{1}{2}} - \theta_k. \tag{17}$$

Finally, we stabilize the approximation of the target quantity in (14) by using an exponential moving average of the previously computed $(\Delta g_{\mathsf{B}_j})_{j\leq k}$. More precisely, we recursively compute G_k defined by,

$$G_k = \beta G_{k-1} + (1-\beta)\Delta g_{\mathsf{B}_k}. \tag{18}$$

We finally introduce $\hat{G}_k = G_k/(1-\beta^{k+1})$ to debias the estimator G_k such that the sum of the weights in the average equals 1. This mostly impacts the first iterations as β^{k+1} vanishes quickly; a similar process is used in ADAM (Kingma and Ba, 2015).

Altogether we obtain our main method: Algorithm 2, which we name *Step-Tuned SGD*, as it aims to tune the step-size every two iterations and not at every epoch like most stochastic BB methods. Note that the main idea behind Step-Tuned SGD remains the same than in the deterministic setting: we exploit the curvature properties of $\mathcal{J}_{\mathsf{B}_k}$ through the quantities $\langle \hat{G}_k, \Delta\theta_{\mathsf{B}_k}\rangle$ to devise our method. Compared to Algorithm 1, the iteration index is shifted by 1 so that the estimated step-size γ_{k+1} only depends on mini-batches B_0 up to B_k and is therefore conditionally independent of B_{k+1}. This conditional dependency structure is crucial to obtain the convergence guarantees given in in Section 4.

3.3 Heuristic construction of Step-Tuned SGD

In this section we present the main elements which led us to the step-tuned method of Algorithm 2 and discuss its hyper-parameters. Throughout this paragraph, the term *gradient variation* (GV) denotes the local variations of the gradient; it is simply the difference of consecutive gradients along a sequence. Our heuristic discussion blends discretization arguments and experimental considerations. We use the non-convex regression experiment of Section 3.1 as a test for our intuition and algorithms. A complete description of the methods below is given in Section D, we only sketch the main ideas.

First heuristic experiment with exact GVs. Assume that along any ordered collection $\theta_1, \dots, \theta_k \in \mathbb{R}^P$, one is able to evaluate the GVs of $\mathcal{J}$, that is, terms of the form $\nabla\mathcal{J}(\theta_i) - \nabla\mathcal{J}(\theta_{i-1})$. Recall that we denote $\Delta\theta_i = \theta_i - \theta_{i-1}$, the difference between two consecutive iterates, for all $i \geq 1$. In the deterministic (i.e., noiseless) setting, Algorithm 1 is based on these GVs, indeed,

$$\theta_{k+1} = \theta_k - \frac{\alpha\|\Delta\theta_k\|^2}{\langle \nabla\mathcal{J}(\theta_k) - \nabla\mathcal{J}(\theta_{k-1}), \Delta\theta_k\rangle}\nabla\mathcal{J}(\theta_k), \tag{19}$$

whenever the denominator is positive. Given our sequence of independent random mini-batches $(\mathsf{B}_k)_{k\in\mathbb{N}}$, a heuristic stochastic approximation version of this recursion could be as follows,

$$\theta_{k+1} = \theta_k - \frac{\alpha_k\|\Delta\theta_k\|^2}{\langle \nabla\mathcal{J}(\theta_k) - \nabla\mathcal{J}(\theta_{k-1}), \Delta\theta_k\rangle}\nabla\mathcal{J}_{\mathsf{B}_k}(\theta_k), \tag{Exact-GV}$$

where the difference between (19) and Exact-GV lies in the randomness of the search direction and the dependency of the scaling factor α_k which aims to moderate the effect of noise (generally $\alpha_k \to 0$). As shown in Figure 3 the recursion Exact-GV is much faster than SGD especially for the first ~ 150 epochs which is often the main concern for DL applications.

For large sums, the gradient-variation in Exact-GV is too computationally expensive. One should therefore adapt (Exact-GV) to the mini-batch context. A direct adaption would simply consists in the algorithm,

$$\theta_{k+1} = \theta_k - \frac{\alpha_k\|\Delta\theta_k\|^2}{\langle \nabla\mathcal{J}_{\mathsf{B}_k}(\theta_k) - \nabla\mathcal{J}_{\mathsf{B}_{k-1}}(\theta_{k-1}), \Delta\theta_k\rangle}\nabla\mathcal{J}_{\mathsf{B}_k}(\theta_k), \tag{Stochastic-GV}$$

where mini-batches are used both to obtain a search direction and to approximate the GV. For this naive approach, we observe a dramatic loss of performance, as illustrated in Figure 3. This reveals the necessity to use accurate stochastic approximation of GVs.

Second heuristic experiment using expected gradient variations. Towards a more stable approximation of the GVs, we consider the following recursion,

$$\theta_{k+1} = \theta_k - \frac{\alpha_k \|\Delta\theta_k\| \|\nabla \mathcal{J}_{\mathsf{B}_{k-1}}(\theta_{k-1})\|}{\langle -\mathbb{E}_{\mathsf{S}}[\mathcal{C}_{\mathcal{J}_{\mathsf{S}}}(\theta_{k-1})], \Delta\theta_k \rangle} \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k), \tag{Expected-GV}$$

where $\mathcal{C}_{\mathcal{J}_{\mathsf{B}}}$ is defined in (14) for any $\mathsf{B} \subset \{1, \dots, N\}$ and the expectation taken is over the independent draw of $\mathsf{S} \subset \{1, \dots, N\}$, conditioned on the other random variables. The main difference with Exact-GV is the use of expected GVs instead of exact GVs, the minus sign ensures a coherent interpretation in term of GVs. The numerator in Expected-GV is also modified to ensure homogeneity of the steps with the other variations of the algorithm. Indeed $\mathcal{C}_{\mathcal{J}_{\mathsf{B}}}(\theta_k)$ approximates a difference of gradients modulo a step-size, see (15). As illustrated in Figure 3, the recursion Expected-GV provides performances comparable to Exact-GV, and in particular for both algorithms, we also recover the loss drop which was observed in the deterministic setting.

Algorithm 2 is nothing less than an approximate version of Expected-GV which combines a double use of mini-batches with a moving average. Indeed, from (15), considering the expectation over the random draw of S, for any $\theta \in \mathbb{R}^P$ and small $\gamma > 0$, we have,

$$-\gamma \mathbb{E}_{\mathsf{S}}[\mathcal{C}_{\mathcal{J}_{\mathsf{S}}}(\theta)] \simeq \mathbb{E}_{\mathsf{S}}\left[\nabla \mathcal{J}_{\mathsf{S}}\left(\theta - \gamma \nabla \mathcal{J}_{\mathsf{S}}(\theta)\right) - \nabla \mathcal{J}_{\mathsf{S}}(\theta)\right]. \tag{20}$$

The purpose of the term $\hat{G}_k$ in Algorithm 2 is precisely to mimick this last quantity. The experimental results of Algorithm 2 are very similar to those of Expected-GV, see Figure 3.

Let us conclude by saying that the above considerations on gradient variations (GVs) led us to propose Algorithm 2 as a possible stochastic version of Algorithm 1. The similarity between the performances of the two methods and the underlying geometric aspects (see Section 3.2) were also major motivations.

Parameters of the algorithm. Algorithm 2 contains more hyper-parameters than in the deterministic case, but we recommend to keep the default values for most of them.[2] Like in most optimizers (SGD, Adam, RMSprop, etc.), only the parameter $\alpha > 0$ has to be carefully tuned to get the most of Algorithm 2. Note that we enforce $\gamma_k \in [\tilde{m}, \tilde{M}]$. The bounds stabilize the algorithm and also play an important role for the convergence as we will show in Section 4. Note that we also enforce the step-size to decrease using a decay of the form $1/k^\delta$ where δ is usually close to 0.5. This standard procedure goes back to Robbins and Monro (1951) and is again necessary to obtain the convergence results presented next.

[2]Default values: $(\nu, \beta, \tilde{m}, \tilde{M}, \delta) = (2, 0.9, 0.5, 2, 0.501)$

4 Theoretical results

We study the convergence of Step-Tuned SGD for smooth non-convex stochastic optimization which encompasses in particular smooth DL problems.

Main result. We recall that $\mathcal{J}$ is a finite sum of twice continuously-differentiable functions $(\mathcal{J}_n)_{n=1,\dots,N}$. Hence, the gradient of $\mathcal{J}$ and the gradients of each $\mathcal{J}_n$ are locally Lipschitz-continuous. A function g is locally Lipschitz-continuous on $\mathbb{R}^P$ if for any $\theta \in \mathbb{R}^P$, there exists a neighborhood V of θ and a constant $L \in \mathbb{R}_+$ such that for all $\psi_1, \psi_2 \in \mathrm{V}$,

$$\|g(\psi_1) - g(\psi_2)\| \leq L \|\psi_1 - \psi_2\|. \tag{21}$$

We assume that $\mathcal{J}$ is lower-bounded on $\mathbb{R}^P$. The main theoretical result of this paper follows.

Theorem 4.1. *Let $\theta_0 \in \mathbb{R}^P$, and let $(\theta_k)_{k\in\mathbb{N}}$ be a sequence generated by Step-Tuned SGD initialized at θ_0. Assume that there exists $C_1 > 0$ such that almost surely $\sup_{k\in\mathbb{N}} \|\theta_k\| < C_1$. Then the sequence of values $(\mathcal{J}(\theta_k))_{k\in\mathbb{N}}$ converges almost surely and $\|\nabla \mathcal{J}(\theta_k)\|^2$ converges to 0 almost surely. In addition, for $k \in \mathbb{N}^\star$,*

$$\min_{j\in\{0,\dots,k-1\}} \mathbb{E}\left[\|\nabla \mathcal{J}(\theta_j)\|^2\right] = O\left(\frac{1}{k^{1-\delta}}\right).$$

The results above state in particular that a realization of the algorithm reaches a point where the gradient is arbitrarily small with probability one. Note that the rate depends on the parameter $\delta \in (0.5, 1)$ which can be chosen by the user and corresponds to the decay schedule $1/(k+1)^\delta$. In most cases, one will want to slowly decay the step-size so $\delta \simeq 0.5$ and the rate is close to $1/\sqrt{k+1}$.

An alternative to the boundedness assumption. In Theorem 4.1 we make the assumption that almost surely the iterates $(\theta_k)_{k\in\mathbb{N}}$ are uniformly bounded. While this is usual for non-convex problems tackled with stochastic algorithms (Davis et al., 2020; Duchi and Ruan, 2018; Castera et al., 2019)) this may be hard to check or enforce in practice. One can alternatively

leverage additional regularity assumptions on the loss function as Li and Orabona (2019) did for example for the scalar variant of ADAGRAD. This is more restrictive than the locally Lipschitz-continuous property of the gradient that we used but for completeness we provide below an alternative version of Theorem 4.1 for such a framework.

Corollary 4.2. *Let $\theta_0 \in \mathbb{R}^P$, and let $(\theta_k)_{k\in\mathbb{N}}$ be a sequence generated by Step-Tuned SGD initialized at θ_0. Assume that each $\mathcal{J}_n$ and $\nabla\mathcal{J}_n$ are Lipschitz-continuous on $\mathbb{R}^P$, $n = 1, \dots, N$. Then the same conclusions as in Theorem 4.1 apply.*

Proof sketch of Theorem 4.1. The proof of our main theorem is fully detailed in Section B of the Supplementary. Here we present the key elements of this proof.

- The proof relies on the descent lemma: for any compact subset $\mathsf{C} \subset \mathbb{R}^P$ there exists $L > 0$ such that for any $\theta \in \mathsf{C}$ and $d \in \mathbb{R}^P$ such that $\theta + d \in \mathsf{C}$,

$$\mathcal{J}(\theta + d) \leq \mathcal{J}(\theta) + \langle \nabla\mathcal{J}(\theta), d\rangle + \frac{L}{2}\|d\|^2. \quad (22)$$

- Let $(\theta_k)_{k\in\mathbb{N}}$ be a realization of the algorithm. Using the boundedness assumption, almost surely the iterates belong to a compact subset C on which $\nabla\mathcal{J}$ and the gradients estimates $\nabla\mathcal{J}_{\mathsf{B}_k}$ are uniformly bounded. So at any iteration $k \in \mathbb{N}$, we may use the descent lemma (22) on the update direction $d = -\gamma_k\nabla\mathcal{J}(\theta_k)$ to bound the difference $\mathcal{J}(\theta_{k+1}) - \mathcal{J}(\theta_k)$.

- As stated in Section 3.2, conditioning on $\mathsf{B}_0, \dots, \mathsf{B}_{k-1}$ the step-size γ_k is constructed to be independent of the current mini-batch B_k. Using this and the descent lemma, we show that there exist $M_1, M_2 > 0$ such that, for all $k \in \mathbb{N}^\star$,

$$\begin{aligned}&\mathbb{E}\left[\mathcal{J}(\theta_{k+1}) \mid \mathsf{B}_0, \dots \mathsf{B}_{k-1}\right]\\ &\leq \mathbb{E}\left[\mathcal{J}(\theta_k)\right] - \frac{M_1}{(k+1)^\delta}\|\nabla\mathcal{J}(\theta_k)\|^2 + \frac{M_2}{(k+1)^{2\delta}}.\end{aligned} \quad (23)$$

- The convergence of $(\mathcal{J}(\theta_k))_{k\in\mathbb{N}}$ follows from (23) using a martingale argument and the fact that $\sum_{k=0}^{+\infty}\frac{1}{(k+1)^{2\delta}} < \infty$.

- The convergence of $\|\nabla\mathcal{J}(\theta_k)\|^2$ is obtained by using the tower property of conditional expectations on (23) and summing to deduce that,

$$\sum_{k=0}^{+\infty}\frac{1}{(k+1)^\delta}\|\nabla\mathcal{J}(\theta_k)\|^2 < +\infty. \quad (24)$$

Since $\sum_{k=0}^{+\infty}\frac{1}{(k+1)^\delta} = +\infty$ The conclusion then follows using analysis arguments similar to those in Li and Orabona (2019).

5 Application to Deep Learning

We finally evaluate the performance of Step-Tuned SGD by training DNNs. We consider four different problems presented next and fully-specified in Section A.1 of the Supplementary. The results with Problems (a) and (b) are presented here while the results with Problems (c) and (d) are relegated to the Supplementary. We compare Step-Tuned SGD with two of the most popular non-momentum methods, SGD and RMSprop Tieleman and Hinton (2012), and we also consider the momentum method ADAM (Kingma and Ba, 2015) which is a very popular DL optimizer. Our method is detailed below.

5.1 Setting of the experiments

- We consider image classification problems (CIFAR-10 and CIFAR-100 (Krizhevsky, 2009)) and the training of an auto-encoder on MNIST (LeCun et al., 2010).

- The networks are slightly modified versions of Lenet (LeCun et al., 1998), ResNet-20 (He et al., 2016) and the auto-encoder of Hinton and Salakhutdinov (2006).

- As specified in Table 1 of the Supplementary, we used either smooth (ELU, SiLU) or nonsmooth (ReLU) activations (ResNets were used with ReLU). Though our theoretical analysis only applies to smooth activations, we did not in practice observe a significant qualitative difference between ReLU or its smooth versions.

- For image classification tasks, the dissimilarity measure is the cross-entropy, and for the auto-encoder, it is the mean-squared error. In each problem we also add a ℓ^2-regularization parameter (a.k.a. weight decay) of the form $\frac{\lambda}{2}\|\theta\|_2^2$.

- For each algorithm, we selected the learning rate parameter α from the set $\{10^{-4}, \dots, 10^0\}$. The value is selected as the one yielding minimum training loss after 10% of the total number of epochs. For example, if we intend to train the network during 100 epochs, the grid-search is carried on the first 10 epochs. For Step-Tuned SGD, the parameter ν was selected with the same criterion from the set $\{1, 2, 5\}$. All other parameters of the algorithms are left to their default values.

- The step-size decay schedule of SGD and Step-Tuned SGD has the form $1/q^\delta$ where q is the current epoch index and $\delta = 0.501$. It slightly differs from what is given in Algorithm 2 as we apply the decay at each epoch instead of each iteration. This slower schedule still satisfies the conditions of Section 4, so convergence is still granted. RMSprop and ADAM rely on their own adaptive procedure and are used without step-size decay schedule.
- The experiments were run on a Nvidia GTX 1080 TI GPU, with an Intel Xeon 2640 V4 CPU. The code was written in *Python* 3.6.9 and *PyTorch* 1.4 (Paszke et al., 2019).

5.2 Results

We perform two types of experiments, a comparative one to assess the quality of Step-Tuned SGD against concurrent optimization algorithms, and another one to study the effect of changing the way mini-batches are used.

Comparison with standard methods. The results with ResNets trained on CIFAR-10 and CIFAR-100 are displayed on Figure 4. The results for the other two experiments are postponed to Section A.2 of the Supplementary. In Figure 4, we display the evolution of the values of the loss function and of the test accuracy during the training phase. We observe a recurrent behavior: during early training Step-Tuned SGD behaves similarly than other methods, then there is a sudden drop of the loss (combined with an improvement in terms of test accuracy which we discuss below). As a result, it achieves best training performance among all algorithms. This behavior is in accordance with our preliminary observations in Figure 3. A similar behavior has been be reported in the literature when using SGD with a manually enforced reduction of the learning rate after a prescribed number of epochs, see, e.g., He et al. (2016). Our algorithm provides a similar qualitative behavior but in an automatic way.

In addition to efficient training performance (in terms of loss values), Step-Tuned SGD generalizes well (as measured by test accuracy). Figure 4 even shows a correlation between test accuracy and training loss. Conditions or explanations for when this happens are not fully understood to this day. Yet, SGD is often said to behave well with respect to this matter (Wilson et al., 2017) and hence it is satisfactory to observe that Step-Tuned SGD seems to inherit this property.

To conclude, in most cases the adaptive step-sizes of Step-Tuned SGD represent a significant improvement compared to SGD. It also seems to be a good alternative to adaptive methods like RMSprop or ADAM especially on residual networks.

Effect of the new mini-batch sub-sampling. Step-Tuned SGD departs from the usual process of drawing a new mini-batch after each gradient update. Indeed, we use each mini-batch twice in order to properly approach the curvature of the sliding loss, but also to maintain a computing time similar to standard algorithms.

We need to make sure that using the same mini-batch twice is not the source of the observed advantage of Step-Tuned SGD. We therefore performed additional experiments where all competing methods are used with the mini-batch drawing procedure of Step-Tuned SGD (each mini-batch being used to compute two consecutive gradient steps). The results are presented in Figure 5 and in Section A.3 of the Supplementary. We observe that this variant actually reduces the performance of concurrent methods especially in term of training error. Thus, on these problems, changing the mini-batch strategy is not the reason for the success of our method.

6 Conclusion

We presented a new method to tune SGD step-sizes for stochastic non-convex optimization within a first-order computational framework. Using empirical and geometric considerations, we addressed the difficulty of preserving favorable behaviors of deterministic algorithms while dealing with mini-batches. In particular, we tackled the problem of adapting the step-sizes to the local landscape of non-convex loss functions. For a computational cost similar to SGD, our method uses a step-size schedule changing every two iterations unlike other stochastic methods à la Barzilai-Borwein. Our algorithm comes with asymptotic convergence results and convergence rates. The method shows efficiency in DL with a typical sudden drop of the error rate at medium stages, especially on ResNets.

Acknowledgements

The authors acknowledge the support of the European Research Council (ERC FACTORY-CoG-6681839), the Agence Nationale de la Recherche (ANR 3IA-ANITI, ANR-17-EURE-0010 CHESS, ANR-19-CE23-0017 MASDOL) and the Air Force Office of Scientific Research (FA9550-18-1-0226).

Part of the numerical experiments were done using the OSIRIM platform of IRIT, supported by the CNRS, the FEDER, Région Occitanie and the French

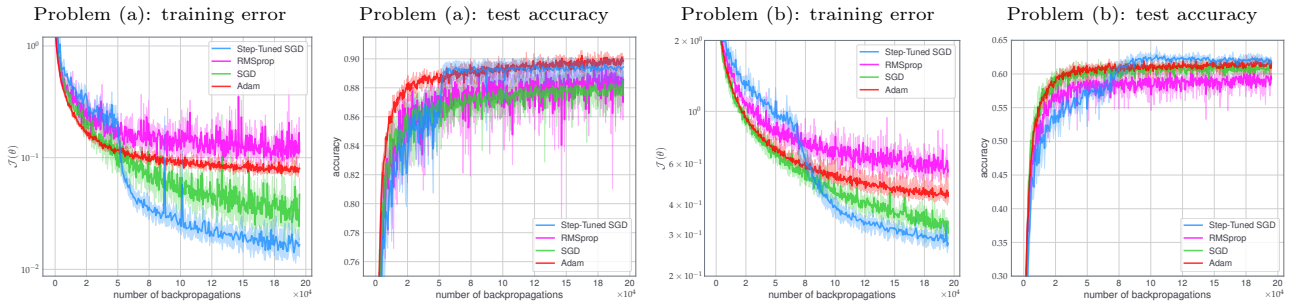

Figure 4: Classification of CIFAR-10 (left) and CIFAR-100 (right) with ResNet-20, corresponding to Problems (a) and (b) described in Section A.1 of the Supplementary. Continuous lines: average values from 3 random initializations. Limits of shadow area: best and worst runs (in training loss). For fair comparison values are plotted against the number of gradient estimates computed (using back-propagation).

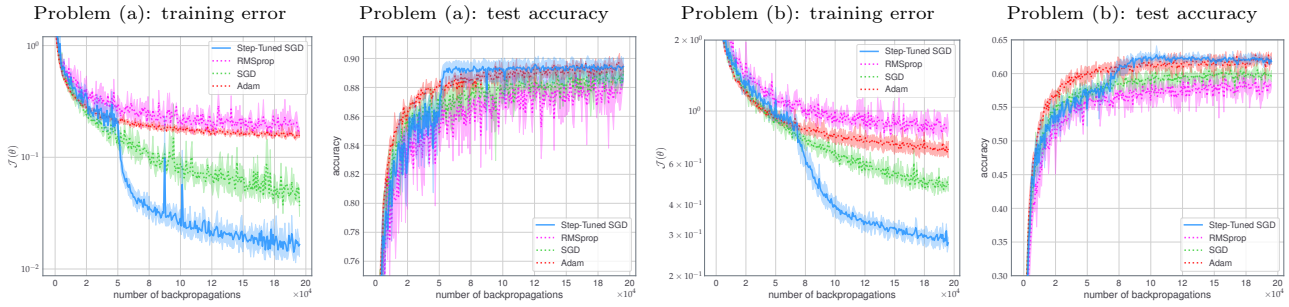

Figure 5: Experiment on the effect of using the same mini-batch twice. For each algorithm, a mini-batch is drawn and used twice consecutively to perform two updates as in Algorithm 2.

government (`http://osirim.irit.fr/site/en`). We thank the development teams of the following libraries that were used in the experiments: Python (Rossum, 1995), Numpy (Walt et al., 2011), Matplotlib (Hunter, 2007), PyTorch (Paszke et al., 2019), and the PyTorch implementation of ResNets from Idelbayev (2018).

We thank Emmanuel Soubies and Sixin Zhang for useful discussions.

References

Alber, Y. I., Iusem, A. N., and Solodov, M. V. (1998). On the projected subgradient method for nonsmooth convex optimization in a hilbert space. *Mathematical Programming*, 81(1):23–35.

Allen-Zhu, Z. (2018). Natasha 2: Faster non-convex optimization than SGD. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2675–2686.

Alvarez, F. and Cabot, A. (2004). Steepest descent with curvature dynamical system. *Journal of Optimization Theory and Applications*, 120(2):247–273.

Babaie-Kafaki, S. and Fatemi, M. (2013). A modified two-point stepsize gradient algorithm for unconstrained minimization. *Optimization Methods and Software*, 28(5):1040–1050.

Barakat, A. and Bianchi, P. (2018). Convergence of the ADAM algorithm from a dynamical system viewpoint. *arXiv preprint:1810.02263*.

Barzilai, J. and Borwein, J. M. (1988). Two-point step size gradient methods. *IMA journal of Numerical Analysis*, 8(1):141–148.

Bertsekas, D. P., Hager, W., and Mangasarian, O. (1998). *Nonlinear programming*. Athena Scientific Belmont, MA.

Biglari, F. and Solimanpur, M. (2013). Scaling on the spectral gradient method. *Journal of Optimization Theory and Applications*, 158:626–635.

Bolte, J. and Pauwels, E. (2020). A mathematical model for automatic differentiation in machine learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Carmon, Y., Duchi, J. C., Hinder, O., and Sidford, A. (2017). Convex until proven guilty: Dimension-free acceleration of gradient descent on non-convex functions. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 654–663.

Castera, C., Bolte, J., Févotte, C., and Pauwels, E. (2019). An inertial Newton algorithm for deep learning. *arXiv preprint:1905.12278*.

Curtis, F. E. and Guo, W. (2016). Handling nonpositive curvature in a limited memory steepest descent method. *IMA Journal of Numerical Analysis*, 36(2):717–742.

Curtis, F. E. and Robinson, D. P. (2019). Exploiting negative curvature in deterministic and stochastic optimization. *Mathematical Programming*, 176(1-2):69–94.

Dai, Y., Yuan, J., and Yuan, Y.-X. (2002). Modified two-point stepsize gradient methods for unconstrained optimization. *Computational Optimization and Applications*, 22(1).

Davis, D., Drusvyatskiy, D., Kakade, S., and Lee, J. D. (2020). Stochastic subgradient method converges on tame functions. *Foundations of Computational mathematics*, 20(1):119–154.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

Duchi, J. C. and Ruan, F. (2018). Stochastic methods for composite and weakly convex optimization problems. *SIAM Journal on Optimization*, 28(4):3229–3259.

Durrett, R. (2019). *Probability: theory and examples, 5th edition*, volume 49. Cambridge university press.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95.

Idelbayev, Y. (2018). Proper ResNet implementation for CIFAR10/CIFAR100 in PyTorch. `https://github.com/akamaster/pytorch_resnet_cifar10`.

Johnson, R. and Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems (NIPS)*, pages 315–323.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Krishnan, S., Xiao, Y., and Saurous, R. A. (2018). Neumann optimizer: A practical optimization algorithm for deep neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report, Canadian Institute for Advanced Research.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y., Cortes, C., and Burges, C. (2010). MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*.

Li, X. and Orabona, F. (2019). On the convergence of stochastic gradient descent with adaptive stepsizes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 983–992.

Liang, J., Xu, Y., Bao, C., Quan, Y., and Ji, H. (2019). Barzilai–Borwein-based adaptive learning rate for deep learning. *Pattern Recognition Letters*, 128:197 – 203.

Liu, M. and Yang, T. (2017). On noisy negative curvature descent: Competing with gradient descent for faster non-convex optimization. *arXiv preprint:1709.08571*.

Martens, J. and Grosse, R. (2015). Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2408–2417.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NIPS)*, pages 8026–8037.

Raydan, M. (1997). The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. *SIAM Journal on Optimization*, 7(1):26–33.

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

Robles-Kelly, A. and Nazari, A. (2019). Incorporating the Barzilai-Borwein adaptive step size into subgradient methods for deep network training. In *2019 Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–6.

Rossum, G. (1995). *Python reference manual*. CWI (Centre for Mathematics and Computer Science).

Royer, C. W. and Wright, S. J. (2018). Complexity analysis of second-order line-search algorithms for smooth nonconvex optimization. *SIAM Journal on Optimization*, 28(2):1448–1477.

Schraudolph, N. N., Yu, J., and Günter, S. (2007). A stochastic Quasi-Newton method for online convex optimization. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Tan, C., Ma, S., Dai, Y.-H., and Qian, Y. (2016). Barzilai-Borwein step size for stochastic gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*, pages 685–693.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Walt, S. v. d., Colbert, S. C., and Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. (2017). The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4148–4158.

Xiao, Y., Wang, Q., and Wang, D. (2010). Notes on the Dai–Yuan–Yuan modified spectral gradient method. *Journal of Computational and Applied Mathematics*, 234(10):2986 – 2992.

A Deep learning: Details and additional experiments

A.1 Summary of the DL experiments

In addition to the method described in Section 5.1, we provide in Table 1 a summary of each problem considered.

Table 1: Setting of the four different deep learning experiments.

	Problem (a)	**Problem (b)**
Type	Image classification	Image classification
Dataset	CIFAR-10	CIFAR-100
Network	ResNet-20 (Residual, Convolutional)	ResNet-20 (Residual, Convolutional)
Batch-size	128	128
Activation functions	ReLU	ReLU
Dissimilarity measure	Cross-entropy	Cross-entropy
Regularization	$\lambda = 10^{-4}$	$\lambda = 10^{-4}$
Grid-search	50 epochs	50 epochs
Stop-criterion	500 epochs	500 epochs
	Problem (c)	**Problem (d)**
Type	Image classification	Auto-encoder
Dataset	CIFAR-10	MNIST
Network	Lenet (Classical, Convolutional)	Dense
Batch-size	128	128
Activation functions	ELU	SiLU
Dissimilarity measure	Cross-entropy	Mean-squared error
Regularization	$\lambda = 10^{-4}$	$\lambda = 10^{-4}$
Grid-search	30 epochs	50 epochs
Stop-criterion	300 epochs	500 epochs

A.2 Additional comparative experiments

We compared different algorithms for solving Problems (a) and (b) in Section 5, below we provide the same type of experiments for Problems (c) and (d). The results are presented in Figure 6.

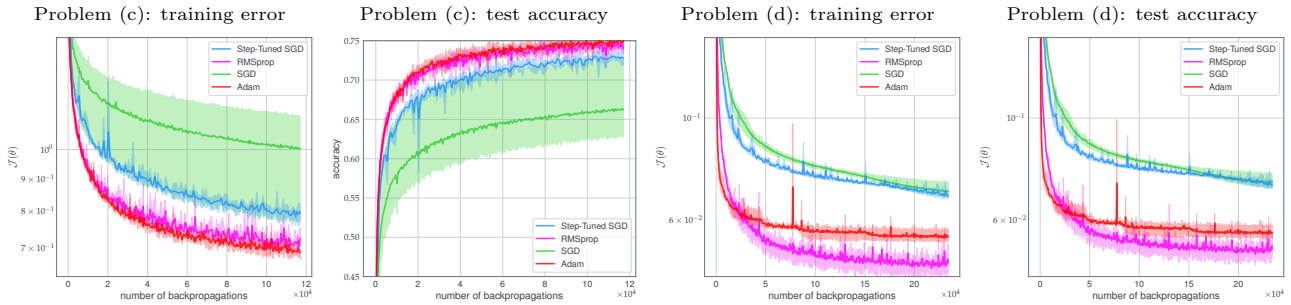

Figure 6: Classification of CIFAR-10 with an adaptation of LeNet (left) and training of an auto-encoder on MNIST (right). This corresponds to Problems (c) and (d) of Table 1. Continuous lines: average values from 3 random initializations. Limits of shadow area: best and worst runs (in training loss). For fair comparison values are plotted against the number of gradient estimates computed (using back-propagation).

On these problems we do not observe the same drop in training loss as we observed when training the ResNets. Yet, in every cases Step-Tuned SGD represents an improvement to vanilla SGD for a negligible additional computational cost. RMSprop and ADAM are more efficient for training the auto-encoder. This may come from the fact that these methods use a coordinate-wise step-size, acting as a preconditioners, which may be particularly useful to train networks with a complex structure such as auto-encoders.

A.3 Additional experiments on the effect of using twice each mini-batch.

We present two additional experiments on the effect of using each mini-batch twice, similarly to what we presented in Figure 5. The results for Problems (c) and (d) are reported on Figure 7.

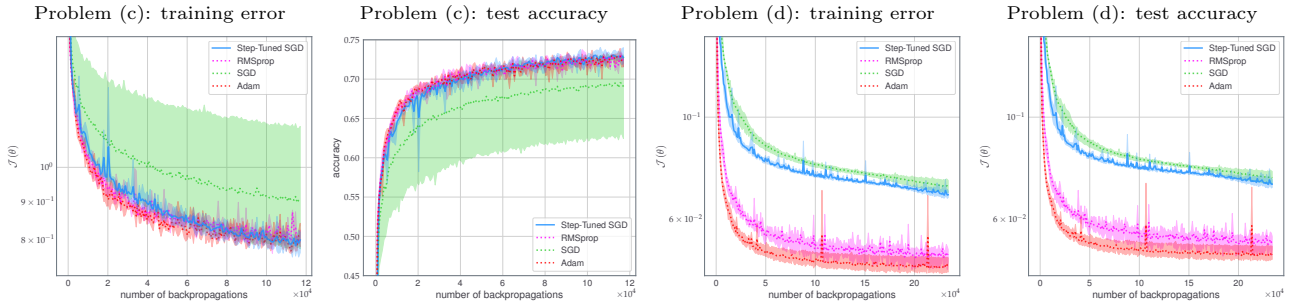

Figure 7: Experiment on the effect of using the same mini-batch twice. For each algorithm, a mini-batch is drawn and used twice consecutively to perform two updates as in Algorithm 2. Classification of CIFAR-10 with an adaptation of LeNet (left) and training of an auto-encoder on MNIST (right).

Like for Problems (a) and (b) we observe that changing the way of using the mini-batches is not the source of the satisfactory results of our method. Actually, under this usage of the mini-batches, ADAM and RMSprop achieve similar results as Step-Tuned SGD on LeNet (unlike the comparative experiments of Figure 6). Finally, these experiments consolidate the hypothesis that coordinate-wise step-sizes are the reason for the gap of performance between ADAM, RMSprop and Step-Tuned SGD on the training of the auto-encoder.

B Proof of the theoretical results

We state a lemma that we will use to prove Theorem 4.1.

B.1 Preliminary lemma

The result is the following.

Lemma B.1 (Alber et al. (1998, Proposition 2))**.** *Let $(u_k)_{k\in\mathbb{N}}$ and $(v_k)_{k\in\mathbb{N}}$ two non-negative real sequences. Assume that $\sum_{k=0}^{+\infty} u_k v_k < +\infty$, and $\sum_{k=0}^{+\infty} v_k = +\infty$. If there exist a constant $C > 0$ such that $\forall k \in \mathbb{N}, |u_{k+1} - u_k| \leq C v_k$, then $u_k \xrightarrow[k\to+\infty]{} 0$.*

B.2 Proof of the main theorem

We can now prove Theorem 4.1.

Proof of Theorem 4.1. We first clarify the random process induced by the draw of the mini-batches. Algorithm 2 takes a sequence of mini-batches as input. This sequence is represented by the random variables $(\mathsf{B}_k)_{k\in\mathbb{N}}$ as described in Section 3.2. Each of these random variables is independent of the others. In particular, for $k \in \mathbb{N}^\star$, B_k is independent of the previous mini-batches $\mathsf{B}_0, \dots, \mathsf{B}_{k-1}$. For convenience, we will denote $\underline{\mathsf{B}}_k = \{\mathsf{B}_0, \dots, \mathsf{B}_k\}$, the mini-batches up to iteration k. Due to the randomness of the mini-batches, the algorithm is a random process as well. As such, θ_k is a random variable with a deterministic dependence on $\underline{\mathsf{B}}_{k-1}$ and is independent of B_k. However, $\theta_{k+\frac{1}{2}}$ and B_k are not independent. Similarly, we constructed γ_k such that it is a random variable with a deterministic dependence on $\underline{\mathsf{B}}_{k-1}$, which is independent of B_k. This dependency structure will be crucial to derive and bound conditional expectations. Finally, we highlight the following important identity, for any $k \geq 1$,

$$\mathbb{E}\left[\nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k) \middle| \underline{\mathsf{B}}_{k-1}\right] = \nabla \mathcal{J}(\theta_k). \tag{25}$$

Indeed, the iterate θ_k is a deterministic function of $\underline{\mathsf{B}}_{k-1}$, so taking the expectation over B_k, which is independent of $\underline{\mathsf{B}}_{k-1}$, we recover the full gradient of $\mathcal{J}$ as the distribution of B_k is the same as that of S in Section 3.2. Notice in addition that the same does not hold for $\theta_{k+\frac{1}{2}}$ (as it depends on B_k).

We now provide estimates that will be used extensively in the rest of the proof. The gradient of the loss function $\nabla\mathcal{J}$ is locally Lipschitz-continuous as $\mathcal{J}$ is twice continuously differentiable. By assumption, there exists a compact convex set $\mathsf{C} \subset \mathbb{R}^P$, such that with probability 1, the sequence of iterates $(\theta_k)_{k\in\mathbb{N}}$ belongs to C. Therefore, by local Lipschicity, the restriction of $\nabla\mathcal{J}$ to C is Lipschitz-continuous on C. Similarly, each $\nabla\mathcal{J}_n$ is also Lipschitz-continuous on C. We denote by $L > 0$ a Lipschitz constant common to each $\nabla\mathcal{J}_n$, $n = 1, \dots, N$. Notice that the Lipschicity is preserved by averaging, in other words,

$$\forall \mathsf{B} \subseteq \{1, \dots, N\}, \forall \psi_1, \psi_2 \in \mathsf{C}, \quad \|\mathcal{J}_{\mathsf{B}}(\psi_1) - \mathcal{J}_{\mathsf{B}}(\psi_2)\| \leq L\|\psi_1 - \psi_2\|. \tag{26}$$

In addition, using the continuity of the $\nabla\mathcal{J}_n$'s, there exists a constant $C_2 > 0$, such that,

$$\forall \mathsf{B} \subseteq \{1, \dots, N\}, \forall \psi \in \mathsf{C}, \quad \|\nabla\mathcal{J}_{\mathsf{B}}(\psi)\| \leq C_2. \tag{27}$$

Finally, for a function $g : \mathbb{R}^P \to \mathbb{R}$ with L-Lipschitz continuous gradient, we recall the following inequality called descent lemma (see for example Bertsekas et al. (1998, Proposition A.24)). For any $\theta \in \mathbb{R}^P$ and any $d \in \mathbb{R}^P$,

$$g(\theta + d) \leq g(\theta) + \langle \nabla g(\theta), d\rangle + \frac{L}{2}\|d\|^2. \tag{28}$$

In our case since we only have the L-Lipschitz continuity of $\nabla\mathcal{J}$ on C which is convex, we have a similar bound for $\nabla\mathcal{J}$ on C: for any $\theta \in \mathsf{C}$ and any $d \in \mathbb{R}^P$ such that $\theta + d \in \mathsf{C}$,

$$\mathcal{J}(\theta + d) \leq \mathcal{J}(\theta) + \langle \nabla\mathcal{J}(\theta), d\rangle + \frac{L}{2}\|d\|^2. \tag{29}$$

Let $\theta_0 \in \mathbb{R}^P$ and let $(\theta_k)_{k\in\mathbb{N}}$ a sequence generated by Algorithm 2 initialized at θ_0. By assumption this sequence belongs to C almost surely. To simplify we denote $\eta_k = \alpha\gamma_k(k+1)^{-\delta}$. Fix an iteration iteration $k \in \mathbb{N}$, we can use (29) with $\theta = \theta_k$ and $d = -\eta_k \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)$, almost surely (with respect to the boundedness assumption),

$$\mathcal{J}(\theta_{k+\frac{1}{2}}) \leq \mathcal{J}(\theta_k) - \eta_k \langle \nabla \mathcal{J}(\theta_k), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\rangle + \frac{\eta_k^2}{2} L \|\nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\|^2. \tag{30}$$

Similarly with $\theta = \theta_{k+\frac{1}{2}}$ and $d = -\eta_k \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}})$, almost surely,

$$\mathcal{J}(\theta_{k+1}) \leq \mathcal{J}(\theta_{k+\frac{1}{2}}) - \eta_k \langle \nabla \mathcal{J}(\theta_{k+\frac{1}{2}}), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}})\rangle + \frac{\eta_k^2}{2} L \|\nabla \mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}})\|^2. \tag{31}$$

We combine (30) and (31), almost surely,

$$\begin{aligned}\mathcal{J}(\theta_{k+1}) \leq \mathcal{J}(\theta_k) &- \eta_k \left(\langle \nabla \mathcal{J}(\theta_k), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\rangle + \langle \nabla \mathcal{J}(\theta_{k+\frac{1}{2}}), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}})\rangle \right) \\ &+ \frac{\eta_k^2}{2} L \left(\|\nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\|^2 + \|\nabla \mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}})\|^2 \right).\end{aligned} \tag{32}$$

Using the boundedness assumption and (27), almost surely,

$$\|\nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\|^2 \leq C_2 \quad \text{and} \quad \|\nabla \mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}})\|^2 \leq C_2. \tag{33}$$

So almost surely,

$$\begin{aligned}\mathcal{J}(\theta_{k+1}) \leq \mathcal{J}(\theta_k) &- \eta_k \left(\langle \nabla \mathcal{J}(\theta_k), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\rangle + \langle \nabla \mathcal{J}(\theta_{k+\frac{1}{2}}), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}})\rangle \right) \\ &+ \eta_k^2 L C_2.\end{aligned} \tag{34}$$

Then, we take the conditional expectation of (34) over B_k conditionally on $\underline{\mathsf{B}}_{k-1}$ (the mini-batches used up to iteration $k-1$), we have,

$$\begin{aligned}\mathbb{E}\left[\mathcal{J}(\theta_{k+1})\middle|\underline{\mathsf{B}}_{k-1}\right] \leq \mathbb{E}\left[\mathcal{J}(\theta_k)\middle|\underline{\mathsf{B}}_{k-1}\right] &+ \mathbb{E}\left[\eta_k^2 L C_2\middle|\underline{\mathsf{B}}_{k-1}\right] \\ &- \mathbb{E}\left[\eta_k \left(\langle \nabla \mathcal{J}(\theta_k), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\rangle + \langle \nabla \mathcal{J}(\theta_{k+\frac{1}{2}}), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}})\rangle \right)\middle|\underline{\mathsf{B}}_{k-1}\right].\end{aligned} \tag{35}$$

As explained at the beginning of the proof, θ_k is a deterministic function of $\underline{\mathsf{B}}_{k-1}$, thus, $\mathbb{E}\left[\mathcal{J}(\theta_k)\middle|\underline{\mathsf{B}}_{k-1}\right] = \mathcal{J}(\theta_k)$. Similarly, by construction η_k is independent of the current mini-batch B_k, it is a deterministic function of $\underline{\mathsf{B}}_{k-1}$. Hence, (35) reads,

$$\begin{aligned}\mathbb{E}\left[\mathcal{J}(\theta_{k+1})\middle|\underline{\mathsf{B}}_{k-1}\right] \leq &\mathcal{J}(\theta_k) + \eta_k^2 L C_2 - \eta_k \langle \nabla \mathcal{J}(\theta_k), \mathbb{E}\left[\nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\middle|\underline{\mathsf{B}}_{k-1}\right]\rangle \\ &- \eta_k \mathbb{E}\left[\langle \nabla \mathcal{J}(\theta_{k+\frac{1}{2}}), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}})\rangle\middle|\underline{\mathsf{B}}_{k-1}\right].\end{aligned} \tag{36}$$

Then, we use the fact that $\mathbb{E}\left[\nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\middle|\underline{\mathsf{B}}_{k-1}\right] = \nabla \mathcal{J}(\theta_k)$. Overall, we obtain,

$$\begin{aligned}\mathbb{E}\left[\mathcal{J}(\theta_{k+1})\middle|\underline{\mathsf{B}}_{k-1}\right] \leq &\mathcal{J}(\theta_k) + \eta_k^2 L C_2 - \eta_k \|\nabla \mathcal{J}(\theta_k)\|^2 \\ &- \eta_k \mathbb{E}\left[\langle \nabla \mathcal{J}(\theta_{k+\frac{1}{2}}), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}})\rangle\middle|\underline{\mathsf{B}}_{k-1}\right].\end{aligned} \tag{37}$$

We will now bound the last term of (37). First we write,

$$\begin{aligned}&- \langle \nabla \mathcal{J}(\theta_{k+\frac{1}{2}}), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}})\rangle \\ &= -\langle \nabla \mathcal{J}(\theta_{k+\frac{1}{2}}), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}}) - \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\rangle - \langle \nabla \mathcal{J}(\theta_{k+\frac{1}{2}}), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\rangle.\end{aligned} \tag{38}$$

Using the Cauchy-Schwarz inequality, as well as (26) and (27), almost surely,

$$\begin{aligned}|\langle \nabla \mathcal{J}(\theta_{k+\frac{1}{2}}), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}}) - \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\rangle| &\leq \|\nabla \mathcal{J}(\theta_{k+\frac{1}{2}})\| \|\nabla \mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}}) - \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\| \\ &\leq \|\nabla \mathcal{J}(\theta_{k+\frac{1}{2}})\| L \|\theta_{k+\frac{1}{2}} - \theta_k\| \\ &\leq \|\nabla \mathcal{J}(\theta_{k+\frac{1}{2}})\| L \| - \eta_k \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\| \\ &\leq L C_2^2 \eta_k.\end{aligned} \tag{39}$$

Hence,

$$-\langle \nabla \mathcal{J}(\theta_{k+\frac{1}{2}}), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}})\rangle \leq LC_2^2\eta_k - \langle \nabla \mathcal{J}(\theta_{k+\frac{1}{2}}), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\rangle. \tag{40}$$

We performs similar computations on the last term of (40), almost surely

$$\begin{aligned}
&-\langle \nabla \mathcal{J}(\theta_{k+\frac{1}{2}}), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\rangle \\
&= -\langle \nabla \mathcal{J}(\theta_{k+\frac{1}{2}}) - \nabla \mathcal{J}(\theta_k), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\rangle - \langle \nabla \mathcal{J}(\theta_k), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\rangle \\
&\leq \|\nabla \mathcal{J}(\theta_{k+\frac{1}{2}}) - \nabla \mathcal{J}(\theta_k)\|\|\nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\| - \langle \nabla \mathcal{J}(\theta_k), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\rangle \\
&\leq LC_2\|\theta_{k+\frac{1}{2}} - \theta_k\| - \langle \nabla \mathcal{J}(\theta_k), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\rangle \\
&\leq LC_2^2\eta_k - \langle \nabla \mathcal{J}(\theta_k), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\rangle
\end{aligned} \tag{41}$$

Finally we obtain by combining (38), (40) and (41), almost surely,

$$-\langle \nabla \mathcal{J}(\theta_{k+\frac{1}{2}}), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}})\rangle \leq 2LC_2^2\eta_k - \langle \nabla \mathcal{J}(\theta_k), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\rangle. \tag{42}$$

Going back to the last term of (37), we have, taking the conditional expectation of (42), almost surely

$$\begin{aligned}
&-\eta_k \mathbb{E}\left[\langle \nabla \mathcal{J}(\theta_{k+\frac{1}{2}}), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}})\rangle \middle| \underline{\mathsf{B}}_{k-1}\right] \\
&\quad \leq 2LC_2^2\eta_k^2 - \eta_k \mathbb{E}\left[\langle \nabla \mathcal{J}(\theta_k), \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)\rangle \middle| \underline{\mathsf{B}}_{k-1}\right] \\
&\quad \leq 2LC_2^2\eta_k^2 - \eta_k \langle \nabla \mathcal{J}(\theta_k), \mathbb{E}\left[\nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k) \middle| \underline{\mathsf{B}}_{k-1}\right]\rangle \\
&\quad = 2LC_2^2\eta_k^2 - \eta_k \|\nabla \mathcal{J}(\theta_k)\|^2.
\end{aligned} \tag{43}$$

In the end we obtain, for an arbitrary iteration $k \in \mathbb{N}$, almost surely

$$\mathbb{E}\left[\mathcal{J}(\theta_{k+1}) \middle| \underline{\mathsf{B}}_{k-1}\right] \leq \mathcal{J}(\theta_k) - 2\eta_k \|\nabla \mathcal{J}(\theta_k)\|^2 + \eta_k^2 L(C_2 + 2C_2^2). \tag{44}$$

To simplify we assume that $\tilde{M} > \nu$ (otherwise set $\tilde{M} = \max(\tilde{M}, \nu)$). We use the fact that, $\eta_k \in [\frac{\alpha\tilde{m}}{(k+1)^\delta}, \frac{\alpha\tilde{M}}{(k+1)^\delta}]$, to obtain almost surely,

$$\mathbb{E}\left[\mathcal{J}(\theta_{k+1}) \middle| \underline{\mathsf{B}}_{k-1}\right] \leq \mathcal{J}(\theta_k) - 2\frac{\alpha\tilde{m}}{(k+1)^\delta} \|\nabla \mathcal{J}(\theta_k)\|^2 + \frac{\alpha^2\tilde{M}^2}{(k+1)^{2\delta}} L(C_2 + 2C_2^2). \tag{45}$$

To prove that the sequence $(\mathcal{J}(\theta_k))_{k\in\mathbb{N}}$ converges almost surely, remark that in particular for $k \in \mathbb{N}$, we have,

$$\mathbb{E}\left[\mathcal{J}(\theta_{k+1}) \middle| \underline{\mathsf{B}}_{k-1}\right] \leq \mathcal{J}(\theta_k) + \frac{\alpha^2\tilde{M}^2}{(k+1)^{2\delta}} L(C_2 + 2C_2^2). \tag{46}$$

For each $k \in \mathbb{N}$, set

$$v_k = \frac{\alpha^2\tilde{M}^2}{(k+1)^{2\delta}} L(C_2 + 2C_2^2), \tag{47}$$

this sequence is summable. From (46) we have,

$$\mathbb{E}\left[\mathcal{J}(\theta_{k+1}) \middle| \underline{\mathsf{B}}_{k-1}\right] + \sum_{i=k+1}^{+\infty} v_i \leq \mathcal{J}(\theta_k) + \sum_{i=k}^{+\infty} v_i. \tag{48}$$

This shows that the sequence of random variable $\mathcal{J}(\theta_k) + \sum_{i=k}^{+\infty} v_i$ forms a supermartingale with respect to the filtration induced by the increasing family of random variables $(\underline{\mathsf{B}}_{k-1})_{k\in\mathbb{N}}$ (see chapter 4 of Durrett (2019)). Since it is bounded below, by Theorem 4.2.12 of Durrett (2019) we have that $\mathcal{J}(\theta_k) + \sum_{i=k}^{+\infty} v_i$ admits a limit almost surely as $k \to \infty$. Then, $\sum_{i=k}^{+\infty} v_i$ tends to 0, so $\mathcal{J}(\theta_k)$ converges almost surely.

We now prove that almost surely, $\|\nabla \mathcal{J}(\theta_k))\|^2$ converges to 0 in expectation. To do so, we go back to (45). We have, almost surely,

$$2\frac{\alpha\tilde{m}}{(k+1)^\delta} \|\nabla \mathcal{J}(\theta_k)\|^2 \leq \mathcal{J}(\theta_k) - \mathbb{E}\left[\mathcal{J}(\theta_{k+1}) \middle| \underline{\mathsf{B}}_{k-1}\right] + \frac{\alpha^2\tilde{M}^2}{(k+1)^{2\delta}} L(C_2 + 2C_2^2). \tag{49}$$

We now consider the expectation of (49) (with respect to the random variables $(\mathsf{B}_k)_{k\in\mathbb{N}}$). The tower property of the conditional expectation gives $\mathbb{E}_{\underline{\mathsf{B}}_{k-1}}[\mathbb{E}[\mathcal{J}(\theta_{k+1})|\underline{\mathsf{B}}_{k-1}]] = \mathbb{E}\left[\mathcal{J}(\theta_{k+1})\right]$ so we obtain, for all $k \in \mathbb{N}$,

$$2\frac{\alpha\tilde{m}}{(k+1)^\delta}\mathbb{E}\left[\|\nabla\mathcal{J}(\theta_k)\|^2\right] \leq \mathbb{E}\left[\mathcal{J}(\theta_k)\right] - \mathbb{E}\left[\mathcal{J}(\theta_{k+1})\right] + \frac{\alpha^2\tilde{M}^2}{(k+1)^{2\delta}}L(C_2+2C_2^2). \tag{50}$$

Then for $K \geq 1$, we sum from 0 to $K-1$,

$$\begin{aligned}\sum_{k=0}^{K-1} 2\frac{\alpha\tilde{m}}{(k+1)^\delta}\mathbb{E}\left[\|\nabla\mathcal{J}(\theta_k)\|^2\right] \leq &\sum_{k=0}^{K-1}\mathbb{E}\left[\mathcal{J}(\theta_k)\right] - \sum_{k=0}^{K-1}\mathbb{E}\left[\mathcal{J}(\theta_{k+1})\right] \\ &+ \sum_{k=0}^{K-1}\frac{\alpha^2\tilde{M}^2}{(k+1)^{2\delta}}L(C_2+2C_2^2).\end{aligned} \tag{51}$$

This can be simplified into,

$$\begin{aligned}\sum_{k=0}^{K-1} 2\frac{\alpha\tilde{m}}{(k+1)^\delta}\mathbb{E}\left[\|\nabla\mathcal{J}(\theta_k)\|^2\right] &\leq \mathcal{J}(\theta_0) - \mathbb{E}\left[\mathcal{J}(\theta_K)\right] + \sum_{k=0}^{K-1}\frac{\alpha^2\tilde{M}^2}{(k+1)^{2\delta}}L(C_2+2C_2^2). \\ &\leq \mathcal{J}(\theta_0) - \inf_{\psi\in\mathbb{R}^P}\mathcal{J}(\psi) + \sum_{k=0}^{K-1}\frac{\alpha^2\tilde{M}^2}{(k+1)^{2\delta}}L(C_2+2C_2^2),\end{aligned} \tag{52}$$

from there we can deduce that the right-hand side is bounded. Indeed, by assumption, $\inf_{\psi\in\mathbb{R}^P}\mathcal{J}(\psi) > -\infty$, since in addition, $\delta \in]1/2, 1[$, $\sum \frac{1}{(k+1)^{2\delta}} < +\infty$ we have,

$$\sum_{k=0}^{+\infty}\frac{\alpha^2\tilde{M}^2}{(k+1)^{2\delta}}L(C_2+2C_2^2) < +\infty. \tag{53}$$

Overall, from (52) we deduce that,

$$\sum_{k=0}^{+\infty} 2\frac{\alpha\tilde{m}}{(k+1)^\delta}\mathbb{E}\left[\|\nabla\mathcal{J}(\theta_k)\|^2\right] < +\infty. \tag{54}$$

In (54) each $\|\nabla\mathcal{J}(\theta_k)\|^2$, for $k \in \mathbb{N}^\star$ can be considered as a function of $(\mathsf{B}_j)_{j\in\mathbb{N}}$ although it actually depends only on $(\mathsf{B}_k)_{j<k}$. This way, we can use Beppo-Levi's theorem on the sequence $\left(\sum_{k=0}^{K} 2\frac{\alpha\tilde{m}}{(k+1)^\delta}\|\nabla\mathcal{J}(\theta_k)\|^2\right)_{K\in\mathbb{N}}$ to interchange the expectation and the sum, so,

$$\mathbb{E}\left[\sum_{k=0}^{+\infty} 2\frac{\alpha\tilde{m}}{(k+1)^\delta}\|\nabla\mathcal{J}(\theta_k)\|^2\right] < +\infty. \tag{55}$$

This implies in particular that the random variable,

$$\sum_{k=0}^{+\infty}\frac{1}{(k+1)^\delta}\|\nabla\mathcal{J}(\theta_k)\|^2, \tag{56}$$

is finite almost surely and has finite expectation. Indeed, the sum in (56) must be finite with probability one for (55) to hold. Since $\sum_{k=0}^{+\infty}\frac{1}{(k+1)^\delta} = +\infty$, this implies at least that almost surely,

$$\liminf_{k\to\infty}\|\nabla\mathcal{J}(\theta_k)\|^2 = 0. \tag{57}$$

To prove that in addition $\lim_{k\to\infty}\|\nabla\mathcal{J}(\theta_k)\|^2 = 0$, we will use Lemma B.1 with $u_k = \|\nabla\mathcal{J}(\theta_k)\|^2$ and $v_k = \frac{1}{(k+1)^\delta}$. So we need to prove that there exists $C_3 > 0$ such that $|u_{k+1} - u_k| \leq C_3 v_k$. To do so, we use the L-Lipschitz

continuity of the gradients on C, triangle inequalities and (27). It holds, almost surely, for all $k \in \mathbb{N}$

$$\begin{aligned}
&\left|\|\nabla\mathcal{J}(\theta_{k+1})\|^2 - \|\nabla\mathcal{J}(\theta_k)\|^2\right| \\
=&\ (\ \|\nabla\mathcal{J}(\theta_{k+1})\| + \|\nabla\mathcal{J}(\theta_k)\|\ ) \ \times\ |\ \|\ \nabla\mathcal{J}(\theta_{k+1})\| - \|\nabla\mathcal{J}(\theta_k)\ \|\ | \\
\leq& 2C_2\ |\|\nabla\mathcal{J}(\theta_{k+1})\| - \|\nabla\mathcal{J}(\theta_k)\||| \\
\leq& 2C_2 \|\nabla\mathcal{J}(\theta_{k+1}) - \nabla\mathcal{J}(\theta_k)\| \\
\leq& 2C_2 L \|\theta_{k+1} - \theta_k\| \\
\leq& 2C_2 L \left\| -\eta_k \nabla\mathcal{J}_{\mathsf{B}_k}(\theta_k) - \eta_k \nabla\mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}}) \right\| \\
\leq& 2C_2 L \frac{\alpha\tilde{M}}{(k+1)^\delta} \|\nabla\mathcal{J}_{\mathsf{B}_k}(\theta_k) + \nabla\mathcal{J}_{\mathsf{B}_k}(\theta_{k+\frac{1}{2}})\| \\
\leq& 4C_2^2 L \frac{\alpha\tilde{M}}{(k+1)^\delta}.
\end{aligned} \tag{58}$$

So taking $C_3 = 4C_2^2 L\alpha\tilde{M}$, by Lemma B.1, almost surely, $\lim_{k\to+\infty} \|\nabla\mathcal{J}(\theta_k)\|^2 = 0$.

Finally, we prove the rate of Theorem 4.1. From (54), there exists $C_4 > 0$ such that for any $K \in \mathbb{N}$, it holds,

$$\begin{aligned}
C_4 \geq \sum_{k=0}^{K} \frac{1}{(k+1)^\delta} \mathbb{E}\left[\|\nabla\mathcal{J}(\theta_k)\|^2\right] \geq \min_{k\in\{1,\dots,K\}} \mathbb{E}\left[\|\nabla\mathcal{J}(\theta_k)\|^2\right] \sum_{k=0}^{K} \frac{1}{(k+1)^\delta} \\
\geq (K+1)^{1-\delta} \min_{k\in\{1,\dots,K\}} \|\nabla\mathcal{J}(\theta_k)\|^2,
\end{aligned} \tag{59}$$

and we obtain the rate. □

B.3 Proof of the corollary

Before proving the corollary we recall the following result.

Lemma B.2. *Let $g : \mathbb{R}^P \to \mathbb{R}$ an L-Lipschitz and differentiable function. Then ∇g is uniformly bounded on $\mathbb{R}^P$.*

We can now prove the corollary.

proof of Corollary 4.2. The proof is very similar to the one of Theorem 4.1. Denote L the Lipschitz constant of $\nabla\mathcal{J}$. Then, the descent lemma (30) holds surely and we obtain (52). Then, since for all $n \in \{1,\dots,N\}$, each $\mathcal{J}_n$ is Lipschitz, so is $\mathcal{J}$. Furthermore, globally Lipschitz functions have uniformly bounded gradients so $\nabla\mathcal{J}$ has bounded gradient. Similarly, at iteration $k \in \mathbb{N}$, $\mathbb{E}\left[\|\nabla\mathcal{J}_{\mathsf{B}_k}(\theta_k)\|\right]$ is also uniformly bounded. These arguments allows to follow the lines of the proof of Theorem 4.1 and in particular to obtain (27) and (29), and the same conclusions follow by repeating the same arguments. □

C Details on the synthetic experiments

We detail the non-convex regression problem that we presented in Figure 2 and 3. Given a matrix $A \in \mathbb{R}^{N\times P}$ and a vector $b \in \mathbb{R}^N$, denote A_n the n-*th* line of A. The problem is to minimize a loss function of the form,

$$\theta \in \mathbb{R}^P \mapsto \mathcal{J}(\theta) = \frac{1}{N} \sum_{n}^{N} \phi(A_n^T\theta - b_n), \tag{60}$$

where the non-convexity comes from the dissimilarity measure $\phi(t) = t^2/(1+t^2)$. For more details on the initialization of A and b we refer to Carmon et al. (2017) where this problem is initially proposed. In the experiments of Figure 3, the mini-batch approximation was made by selecting a subset of the lines of A, which amounts to compute only a few terms of the full sum in (60). We used $N = 500$, $P = 30$ and mini-batches of size 50.

In the deterministic setting we ran each algorithm during 250 iterations and selected the hyper-parameters of each algorithm such that they achieved $|\mathcal{J}(\theta) - \mathcal{J}^\star| < 10^{-1}$ as fast as possible. In the mini-batch experiments we ran each algorithm during 250 epochs and selected the hyper-parameters that yielded the smallest value of $\mathcal{J}(\theta)$ after 100 epochs.

D Description of auxiliary algorithms

We precise the heuristic algorithms used in Figure 3 and discussed in Section 3.3.

Algorithm 3 Stochastic-GV SGD

Input: $\alpha > 0$, $\nu > 0$
Input: $\tilde{m} > 0$, $\tilde{M} > 0$, $\delta \in (0.5, 1)$
Initialize $\theta_0 \in \mathbb{R}^P$, $\gamma_0 = 1$
Draw mini-batches $(\mathsf{B}_k)_{k\in\mathbb{N}}$ independently and uniformly at random with replacement.
$\theta_1 = \theta_0 - \alpha\gamma_0 \nabla \mathcal{J}_{\mathsf{B}_0}(\theta_0)$
for $k = 1, \dots$ **do**
  $\Delta\theta_k = \theta_k - \theta_{k-1}$
  $\Delta g_k^{\text{naive}} = \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k) - \nabla \mathcal{J}_{\mathsf{B}_{k-1}}(\theta_{k-1})$
  if $\langle \Delta g_k^{\text{naive}}, \Delta\theta_{\mathsf{B}_k} \rangle > 0$ **then**
    $\gamma_k = \frac{\|\Delta\theta_k\|^2}{\langle \Delta g_k^{\text{naive}}, \Delta\theta_k \rangle}$
  else
    $\gamma_k = \nu$
  end if
  $\gamma_k = \min(\max(\gamma_k, \tilde{m}), \tilde{M})$
  $\theta_{k+1} = \theta_k - \frac{\alpha}{(k+1)^\delta} \gamma_k \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)$
end for

Algorithm 4 Exact-GV SGD

Input: $\alpha > 0$, $\nu > 0$
Input: $\tilde{m} > 0$, $\tilde{M} > 0$, $\delta \in (0.5, 1)$
Initialize $\theta_0 \in \mathbb{R}^P$, $\gamma_0 = 1$
Draw mini-batches $(\mathsf{B}_k)_{k\in\mathbb{N}}$ independently and uniformly at random with replacement.
$\theta_1 = \theta_0 - \alpha\gamma_0 \nabla \mathcal{J}_{\mathsf{B}_0}(\theta_0)$
for $k = 1, \dots$ **do**
  $\Delta\theta_k = \theta_k - \theta_{k-1}$
  $G_k = \nabla \mathcal{J}(\theta_k) - \nabla \mathcal{J}(\theta_{k-1})$
  if $\langle G_k, \Delta\theta_{\mathsf{B}_k} \rangle > 0$ **then**
    $\gamma_k = \frac{\|\Delta\theta_k\|^2}{\langle G_k, \Delta\theta_k \rangle}$
  else
    $\gamma_k = \nu$
  end if
  $\gamma_k = \min(\max(\gamma_k, \tilde{m}), \tilde{M})$
  $\theta_{k+1} = \theta_k - \frac{\alpha}{(k+1)^\delta} \gamma_k \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)$
end for

Algorithm 5 Expected-GV SGD

Input: $\alpha > 0$, $\nu > 0$
Input: $\tilde{m} > 0$, $\tilde{M} > 0$, $\delta \in (0.5, 1)$
Initialize $\theta_0 \in \mathbb{R}^P$, $\gamma_0 = 1$
Draw mini-batches $(\mathsf{B}_k)_{k \in \mathbb{N}}$ independently and uniformly at random with replacement.
$\theta_1 = \theta_0 - \alpha\gamma_0 \nabla \mathcal{J}_{\mathsf{B}_0}(\theta_0)$
for $k = 1, \ldots$ **do**
  $\Delta\theta_k = \theta_k - \theta_{k-1}$
  $G_k = -\frac{\alpha}{(k-1)^\delta}\gamma_{k-1}\mathbb{E}\left[\mathcal{C}_{\mathcal{J}_{\mathsf{B}_{k-1}}}(\theta_{k-1})\right]$
  if $\langle G_k, \Delta\theta_{\mathsf{B}_k}\rangle > 0$ **then**
    $\gamma_k = \frac{\|\Delta\theta_k\|^2}{\langle G_k, \Delta\theta_k\rangle}$
  else
    $\gamma_k = \nu$
  end if
  $\gamma_k = \min(\max(\gamma_k, \tilde{m}), \tilde{M})$
  $\theta_{k+1} = \theta_k - \frac{\alpha}{(k+1)^\delta}\gamma_k \nabla \mathcal{J}_{\mathsf{B}_k}(\theta_k)$
end for