Catalyst for Gradient-based Nonconvex Optimization

Courtney Paquette
Lehigh University

Hongzhou Lin
Massachusetts Institute of Technology

Dmitriy Drusvyatskiy
University of Washington

Julien Mairal
Inria

Zaid Harchaoui
University of Washington

Abstract

We introduce a generic scheme to solve nonconvex optimization problems using gradient-based algorithms originally designed for minimizing convex functions. Even though these methods may originally require convexity to operate, the proposed approach allows one to use them without assuming any knowledge about the convexity of the objective. In general, the scheme is guaranteed to produce a stationary point with a worst-case efficiency typical of first-order methods, and when the objective turns out to be convex, it automatically accelerates in the sense of Nesterov and achieves near-optimal convergence rate in function values. We conclude the paper by showing promising experimental results obtained by applying our approach to incremental algorithms such as SVRG and SAGA for sparse matrix factorization and for learning neural networks.

1   Introduction

We consider optimization problems of the form

$$\min_{x \in \mathbb{R}^p} \left\{ f(x) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(x) + \psi(x) \right\}. \qquad (1)$$

Here, we set $\bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$, each function $f_i : \mathbb{R}^p \to \mathbb{R}$ is smooth, and the regularization $\psi : \mathbb{R}^p \to \bar{\mathbb{R}}$ may be nonsmooth. By considering extended-real-valued functions, this composite setting also encompasses constrained minimization by letting ψ be the indicator function of computationally tractable constraints on x.

Minimization of a regularized empirical risk objective of form (1) is central in machine learning. Whereas a significant amount of work has been devoted to this setting for convex problems, leading in particular to fast incremental algorithms [see, *e.g.*, 9, 18, 23, 32, 33, 34], the question of minimizing efficiently (1) when the functions f_i and ψ may be nonconvex is still largely open today.

Yet, nonconvex problems in machine learning are of utmost interest. For instance, the variable x may represent the parameters of a neural network, where each term $f_i(x)$ measures the fit between x and a data point indexed by i, or (1) may correspond to a nonconvex matrix factorization problem (see Section 6). Besides, even when the data-fitting functions f_i are convex, it is also typical to consider nonconvex regularization functions ψ, for example for feature selection in signal processing [14]. Motivated by these facts, we address two questions from nonconvex optimization:

1. How to apply a method for convex optimization to a nonconvex problem?

2. How to design an algorithm that does not need to know whether the objective function is convex while obtaining the optimal convergence guarantee if the function is convex?

Several works have attempted to transfer ideas from the convex world to the nonconvex one, see, *e.g.*, [12, 13]. Our paper has a similar goal and studies the extension of Nesterov's acceleration for convex problems [25] to nonconvex composite ones. For C^1-smooth and nonconvex problems, gradient descent is optimal among first-order methods in terms of information-based complexity to find an ε-stationary point [5][Thm. 2 Sec. 5]. Without additional assumptions, worst case complexity for first-order methods can not achieve better than $\mathcal{O}(\varepsilon^{-2})$ oracle queries [7, 8]. Under a stronger assumption that the objective function is C^2-smooth, state-of-the-art methods [*e.g.*, 4, 6] using Hessian-vector product only achieve marginal gain with complexity $O(\varepsilon^{-7/4} \log(1/\varepsilon))$ in more limited settings than ours.

For this reason, our work fits within a broader stream of research on methods that *do not perform worse than gradient descent in the nonconvex case* (in terms of worst-case complexity), while *automatically accelerating for minimizing convex functions*. The hope is to see acceleration in practice for nonconvex problems, by exploiting "hidden" convexity in the objective (*e.g.*, local convexity near the optimum, or convexity along the trajectory of iterates).

Table 1: Comparison of rates of convergence when applying 4WD-Catalyst to SVRG. In the convex case, we present the complexity in terms of number of iterations to obtain a point x satisfying $f(x) - f^* < \varepsilon$. In the nonconvex case, we consider instead the guarantee $\mathrm{dist}(0, \partial f(x)) < \varepsilon$. Note that the theoretical stepsize of ncvx-SVRG is much smaller than that of our algorithm and of the original SVRG. In practice, the choice of a small stepsize significantly slows down the performance (see Section 6), and ncvx-SVRG is often heuristically used with a larger stepsize in practice, which is not allowed by theory, see [30]. A mini-batch version of SVRG is also proposed there, allowing large stepsizes of $O(1/L)$, but without changing the global complexity. A similar table for SAGA [9] is provided in [28].

	Th. stepsize	Nonconvex	Convex
SVRG [34]	$O\left(\dfrac{1}{L}\right)$	not avail.	$O\left(n\dfrac{L}{\varepsilon}\right)$
ncvx-SVRG [2, 29, 30]	$O\left(\dfrac{1}{n^{2/3}L}\right)$	$O\left(\dfrac{n^{2/3}L}{\varepsilon^2}\right)$	$O\left(\sqrt{n}\dfrac{L}{\varepsilon}\right)$
4WD-Catalyst -SVRG	$O\left(\dfrac{1}{L}\right)$	$\tilde{O}\left(\dfrac{nL}{\varepsilon^2}\right)$	$\tilde{O}\left(\sqrt{\dfrac{nL}{\varepsilon}}\right)$

Our main contribution is a *generic* meta-algorithm, dubbed 4WD-Catalyst, which is able to use an optimization method $\mathcal{M}$, originally designed for convex problems, and turn it into an accelerated scheme that also applies to nonconvex objectives. 4WD-Catalyst can be seen as a **4-W**heel-**D**rive extension of Catalyst [22] to all optimization "terrains". Specifically, without knowing whether the objective function is convex or not, our algorithm may take a method $\mathcal{M}$ designed for convex optimization problems with the same structure as (1), *e.g.*, SAGA [9], SVRG [34], and apply $\mathcal{M}$ to a sequence of sub-problems such that it provides a stationary point of the nonconvex objective. Overall, the number of iterations of $\mathcal{M}$ to obtain a gradient norm smaller than ε is $\tilde{O}(\varepsilon^{-2})$ in the worst case, while automatically reducing to $\tilde{O}(\varepsilon^{-2/3})$ if the function is convex.[1] We provide the detailed proofs and the extensive experimental results in the longer version of this work [28].

---

[1]In this section, the notation $\tilde{O}$ only displays the polynomial dependency with respect to ε for simplicity.

Related work. Inspired by Nesterov's fast gradient method for convex optimization [26], the first accelerated methods performing universally well for nonconvex and convex problems were introduced in [12, 13]. Specifically, the most recent approach [13] addresses composite problems such as (1) with $n = 1$, and performs no worse than gradient descent on nonconvex instances with complexity $O(\varepsilon^{-2})$ on the gradient norm. When the problem is convex, it accelerates with complexity $O(\varepsilon^{-2/3})$. Extensions to Gauss-Newton methods were also recently developed in [10]. Whether accelerated methods are superior to gradient descent on nonconvex problems remains open; however their performance escaping saddle points faster than gradient descent has been observed [15, 27].

In [21], a similar strategy is proposed, focusing instead on convergence guarantees under the so-called Kurdyka-Łojasiewicz inequality. Our scheme is in the same spirit as these previous papers, since it monotonically interlaces proximal-point steps (instead of proximal-gradient as in [13]) and extrapolation/acceleration steps. A fundamental difference is that our method is generic and can be used to accelerate a given optimization method, which is not the purpose of these previous papers.

By considering C^2-smooth nonconvex objective functions f with Lipschitz continuous gradient ∇f and Hessian $\nabla^2 f$, the authors of [4] propose an algorithm with complexity $O(\varepsilon^{-7/4} \log(1/\varepsilon))$, based on iteratively solving convex subproblems closely related to the original problem. It is not clear if the complexity of their algorithm improves in the convex setting. Note also that the algorithm proposed in [4] is inherently for C^2-smooth minimization. This implies that the scheme does not allow incorporating nonsmooth regularizers and cannot exploit finite-sum structure.

In [30], stochastic methods for minimizing (1) are proposed using variants of SVRG [16] and SAGA [9]. These schemes work for both convex and nonconvex settings and achieve convergence guarantees of $O(Ln/\varepsilon)$ (convex) and $O(n^{2/3}L/\varepsilon^2)$ (nonconvex). Although for nonconvex problems our scheme in its worst case only guarantees a rate of $\tilde{O}\left(\frac{nL}{\varepsilon^2}\right)$, we attain the optimal accelerated rate in the convex setting (See Table 1). The empirical results of [30] used a step size of order $1/L$, but their theoretical analysis without minibatch requires a much smaller stepsize, $1/(n^{2/3}L)$, whereas our analysis is able to use the $1/L$ stepsize.

A stochastic scheme for minimizing (1) under the nonconvex but *smooth* setting was recently considered in [20]. The method can be seen a nonconvex variant of the stochastically controlled stochastic gradient (SCSG) methods [19]. If the target accuracy is small, then the method performs no worse than nonconvex SVRG [30].

If the target accuracy is large, then the method achieves a rate better than SGD. The proposed scheme does not allow nonsmooth regularizers and it is unclear whether numerically the scheme performs as well as SVRG.

Finally, a method related to SVRG [16] for minimizing large sums while automatically adapting to the weak convexity constant of the objective function, is proposed in [1]. When the weak convexity constant is small (*i.e.*, the function is nearly convex), the proposed method enjoys an improved efficiency estimate. This algorithm, however, does not automatically accelerate for convex problems, in the sense that the rate is slower than $O(\varepsilon^{-3/2})$ in terms of target accuracy ε on the gradient norm.

2   Tools for Nonconvex Optimization

In this paper, we focus on a broad class of nonconvex functions known as *weakly convex* or lower C^2 functions, which covers most of the interesting cases of interest in machine learning and resemble convex functions in many aspects.

Definition 2.1 (Weak convexity)**.** A function $f\colon \mathbb{R}^p \to \overline{\mathbb{R}}$ is $\rho-$*weakly convex* if for any points x, y in $\mathbb{R}^p$ and for any λ in $[0, 1]$, the approximate secant inequality holds:

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$
$$+ \tfrac{\rho\lambda(1-\lambda)}{2}\|x - y\|^2.$$

Remark 2.2. When $\rho = 0$, the above definition reduces to the classical definition of convex functions.

Proposition 2.3. *A function f is $\rho-$weakly convex if and only if the function f_ρ is convex, where*

$$f_\rho(x) \triangleq f(x) + \tfrac{\rho}{2}\|x\|^2.$$

Corollary 2.4. *If f is twice differentiable, then f is ρ-weakly convex if and only if $\nabla^2 f(x) \succeq -\rho I$ for all x.*

Intuitively, a function is weakly convex when it is "nearly convex" up to a quadratic function. This represents a complementary notion of the strong convexity.

Proposition 2.5. *If a function f is differentiable and its gradient is Lipschitz continuous with Lipschitz parameter L, then f is L-weakly convex.*

We give the proofs of the above propositions in Sections 2 and 3 of [28]. We remark that for most of the interesting machine learning problems, the smooth part of the objective function admits Lipchitz gradients, meaning that the function is weakly convex.

Tools for nonsmooth optimization   Convergence results for nonsmooth optimization typically rely on the concept of subdifferential. However, the generalization of the subdifferential to nonconvex nonsmooth functions is not unique [3]. With the weak convexity in hand, all these constructions coincide and therefore we will abuse standard notation slightly, as set out for example in Rockafellar and Wets [31].

Definition 2.6 (Subdifferential)**.** Consider a function $f\colon \mathbb{R}^p \to \overline{\mathbb{R}}$ and a point x with $f(x)$ finite. The *subdifferential* of f at x is the set

$$\partial f(x) := \{\xi \in \mathbb{R}^p : f(y) \geq f(x) + \xi^T(y - x)$$
$$+ o(\|y - x\|), \ \forall y \in \mathbb{R}^p\}.$$

Thus, a vector ξ lies in $\partial f(x)$ whenever the linear function $y \mapsto f(x) + \xi^T(y - x)$ is a lower model of f, up to first order around x. In particular, the subdifferential $\partial f(x)$ of a differentiable function f is the singleton $\{\nabla f(x)\}$; while for a convex function f it coincides with the subdifferential in the sense of convex analysis [see 31, Exercise 8.8]. Moreover, the following sum rule,

$$\partial(f + g)(x) = \partial f(x) + \nabla g(x),$$

holds for any differentiable function g.

In nonconvex optimization, standard complexity bounds are derived to guarantee

$$\mathrm{dist}\big(0, \partial f(x)\big) \leq \varepsilon \ .$$

When $\varepsilon = 0$, we are at a stationary point and first-order optimality conditions are satisfied. For functions that are nonconvex, first-order methods search for points with small subgradients, which does not necessarily imply small function values, in contrast to convex functions where the two criteria are much closer related.

3   The 4WD-Catalyst Algorithm

We present here our main algorithm called 4WD-Catalyst. The proposed approach extends the Catalyst method [22] to potentially nonconvex problems, while enjoying the two following properties:

1. When the problem is nonconvex, the algorithm automatically adapts to the unknown weak convexity constant ρ.

2. When the problem is convex, the algorithm automatically accelerates in the sense of Nesterov, providing near-optimal convergence rates for first-order methods.

Main goal.   As in the regular Catalyst algorithm of [22], the proposed scheme wraps in an outer loop a minimization algorithm $\mathcal{M}$ used in an inner loop. The goal is to leverage a method $\mathcal{M}$ that is able to
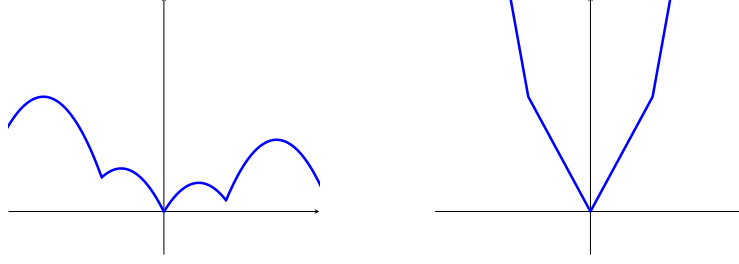
Figure 1: Example of a weakly convex function. The left figure is the original weakly convex function. By adding an appropriate quadratic to the weakly convex function (left), we get the convex function on the right.

exploit the problem structure (finite-sum, composite) in the convex case, and benefit from this feature when dealing with a new problem with unknown convexity; remarkably, $\mathcal{M}$ does not need to have any convergence guarantee for nonconvex problems to be used in 4WD-Catalyst, which is the main originality of our work.

Two-step subproblems. In each iteration, 4WD-Catalyst forms subproblems of the form

$$\min_{x} \quad f_\kappa(x;y) := f(x) + \frac{\kappa}{2}\|x-y\|^2. \quad (\mathcal{P})$$

We call y the *prox-center* and any minimizer of the subproblem a *proximal point*. The perturbed function $f_\kappa(x;y)$ satisfies the important property: $f_\kappa(\cdot;y)$ is $(\kappa-\rho)$-strongly convex for any $\kappa > \rho$. The addition of the quadratic to f makes the subproblem more "convex". That is, when f is nonconvex, a large enough κ yields convex subproblems; even when f is convex, the quadratic perturbation improves conditioning.

We now describe the k'th iteration of Algorithm 1. To this end, suppose we have available iterates x_{k-1} and v_{k-1}. At the center of our Algorithm 1 are two main sequences of iterates $(\bar{x}_k)_k$ and $(\tilde{x}_k)_k$, obtained from approximately solving two subproblems of the form $\mathcal{P}$.

1. **Proximal point step.** We first perform an inexact proximal point step with prox-center x_{k-1}:

$$\bar{x}_k \approx \underset{x}{\operatorname{argmin}}\, f_\kappa(x;x_{k-1}) \quad \text{[Proximal-point step]} \quad (2)$$

2. **Accelerated proximal point step.** Then we build the next prox-center y_k as the combination

$$y_k = \alpha_k v_{k-1} + (1-\alpha_k)x_{k-1}. \quad (3)$$

Next we use y_k as a prox-center and update the next extrapolation term:

$$\tilde{x}_k \approx \underset{x}{\operatorname{argmin}}\, f_\kappa(x;y_k) \quad \text{[Acc. prox.-point step]} \quad (4)$$

$$v_k = x_{k-1} + \frac{1}{\alpha_k}(\tilde{x}_k - x_{k-1}) \quad \text{[Extrapolation]} \quad (5)$$

where $\alpha_{k+1} \in (0,1)$ is a sequence of coefficients satisfying $(1-\alpha_{k+1})/\alpha_{k+1}^2 = 1/\alpha_k^2$. Essentially, the sequences $(\alpha_k)_k, (y_k)_k, (v_k)_k$ are built upon the extrapolation principles of [26].

Algorithm 1 4WD-Catalyst
input Fix a point $x_0 \in \operatorname{dom} f$, real numbers $\kappa_0, \kappa_{\text{cvx}} > 0$ and $T, S > 0$, and an opt. method $\mathcal{M}$.
initialization: $\alpha_1 = 1$, $v_0 = x_0$.
repeat for $k = 1, 2, \ldots$

1. Compute

$$(\bar{x}_k, \kappa_k) = \text{Auto-adapt } (x_{k-1}, \kappa_{k-1}, T).$$

2. Compute $y_k = \alpha_k v_{k-1} + (1-\alpha_k)x_{k-1}$ and apply $S \log(k+1)$ iterations of $\mathcal{M}$ to find

$$\tilde{x}_k \approx \underset{x \in \mathbb{R}^p}{\operatorname{argmin}}\, f_{\kappa_{\text{cvx}}}(x, y_k). \quad (6)$$

3. Update v_k and α_{k+1} by

$$v_k = x_{k-1} + \frac{1}{\alpha_k}(\tilde{x}_k - x_{k-1})$$

$$\text{and} \quad \alpha_{k+1} = \frac{\sqrt{\alpha_k^4 + 4\alpha_k^2} - \alpha_k^2}{2}.$$

4. Choose x_k to be any point satisfying $f(x_k) = \min\{f(\bar{x}_k), f(\tilde{x}_k)\}$.

until the stopping criterion $\operatorname{dist}(0, \partial f(\bar{x}_k)) < \varepsilon$

Picking the best. At the end of iteration k, we have two iterates, resp. $\bar{x}_k$ and $\tilde{x}_k$. Following [12], we simply choose the best of the two in terms of their objective values, that is we choose x_k such that $f(x_k) \leq \min\{f(\bar{x}_k), f(\tilde{x}_k)\}$.

The proposed scheme blends the two steps in a synergistic way, allowing us to recover the near-optimal rates of convergence in both worlds: convex and nonconvex. Intuitively, when $\bar{x}_k$ is chosen, it means that Nesterov's extrapolation step "fails" to accelerate convergence.

We present now our strategy to set the parameters of 4WD-Catalyst in order to a) automatically adapt to the unknown weak convexity constant ρ; b) enjoy near-optimal rates in both convex and nonconvex settings.

Algorithm 2 Auto-adapt (x, κ, T)

---

input $x \in \mathbb{R}^p$, method $\mathcal{M}$, $\kappa > 0$, num. iterations T.

    Repeat Run T iterations of $\mathcal{M}$ initializing from $z_0 = x$, or $z_0 = \text{prox}_{\eta\psi}(x - \eta\nabla f_0(x; y))$ with $\eta = 1/(L + \kappa)$ for composite objectives $f = f_0 + \psi$ with L-smooth function f_0, to obtain

$$z_T \approx \operatorname*{argmin}_{z \in \mathbb{R}^p} f_\kappa(z; x).$$

    If $f_\kappa(z_T; x) \leq f_\kappa(x; x)$
                 and $\text{dist}(0, \partial f_\kappa(z_T; x)) \leq \kappa \|z_T - x\|$,
    then go to output.
    else repeat with $\kappa \to 2\kappa$.
output (z_T, κ).

---

4 Parameter Choices and Adaptation

When κ is large enough, the subproblems become strongly convex; thus globally solvable. Henceforth, we will assume that $\mathcal{M}$ satisfies the following natural linear convergence assumption.

Linear convergence of $\mathcal{M}$ for strongly-convex problems. We assume that for any $\kappa > \rho$, there exist $A_\kappa \geq 0$ and τ_κ in $(0, 1)$ so that the following hold:

1. For any prox-center y in $\mathbb{R}^p$, define $f_\kappa^*(y) = \min_z f_\kappa(z, y)$. For any initial point z_0 in $\mathbb{R}^p$, the iterates $\{z_t\}_{t \geq 1}$ generated by $\mathcal{M}$ on the problem $\min_z f_\kappa(z; y)$ satisfy

$$\text{dist}^2(0, \partial f_\kappa(z_t; y)) \leq A_\kappa (1 - \tau_\kappa)^t (f_\kappa(z_0; y) - f_\kappa^*(y)). \tag{7}$$

2. The rates τ_κ and constants A_κ are increasing in κ.

When the function is strongly convex, the first condition is equivalent to the error $f_\kappa(z_k; y) - f_\kappa^*(y)$ decreasing geometrically to zero (convergence in sub-gradient norm and function values are equivalent in this case). If the method is randomized, we allow (7) to hold in expectation; see Sec. 5.1 in [28]. All algorithms of interest, (e.g., gradient descent, SVRG, SAGA) satisfy these properties.

Adaptation to weak convexity and choice of T. Recall that we add a quadratic to f to make each subproblem convex. Thus, we should set $\kappa > \rho$, if ρ were known. On the other other hand, we do not want κ too large, as that may slow down the overall algorithm. In any case, it is difficult to have an accurate estimate of ρ for machine learning problems such as neural networks. Thus, we propose a procedure described in Algorithm 2 to automatically adapt to ρ.

The idea is to fix in advance a number of iterations T, let $\mathcal{M}$ run on the subproblem for T iterations, output

the point z_T, and check if a sufficient decrease occurs. We show that if we set $T = \tilde{O}(\tau_L^{-1})$, where $\tilde{O}$ hides logarithmic dependencies in L and A_L, where L is the Lipschitz constant of the smooth part of f; then, if the subproblem were convex, the following conditions would be guaranteed:

1. Descent condition: $f_\kappa(z_T; x) \leq f_\kappa(x; x)$;
2. Adaptive stationary condition:

$$\text{dist}(0, \partial f_\kappa(z_T; x)) \leq \kappa \|z_T - x\|.$$

Thus, if either condition is not satisfied, then the subproblem is deemed not convex and we double κ and repeat. The procedure yields an estimate of ρ in a logarithmic number of increases; see [28][Lemma D.3].

The *descent condition* is a sanity check, which ensures the iterates generated by the algorithm always decrease the function value. Without it, the stationarity condition alone is insufficient because of the existence of local maxima in nonconvex problems.

The *adaptive stationarity property* controls the inexactness of the subproblem in terms of subgradient norm. In a nonconvex setting, the subgradient norm is convenient, since we cannot access $f_\kappa(z_T, x) - f_\kappa^*(x)$. Furthermore, unlike the stationary condition $\text{dist}(0, \partial f_\kappa(z_T; x)) < \varepsilon$, where an accuracy ε is predefined, the adaptive stationarity condition depends on the iterate z_T. This turns out to be essential in deriving the global complexity. Sec. 4 in [28] contains more details.

Relative stationarity and predefining S. One of the main differences of our approach with the Catalyst algorithm of [22] is to use a *pre-defined* number of iterations, T and S, for solving the subproblems. We introduce κ_{cvx}, a $\mathcal{M}$ dependent smoothing parameter, and set it in the same way as the smoothing parameter in [22]. The automatic acceleration of our algorithm when the problem is convex is due to extrapolation steps in Step 2-3 of Algo. 1. We show that if we set $S = \tilde{O}\left(\tau_{\kappa_{\text{cvx}}}^{-1}\right)$, where $\tilde{O}$ hides logarithmic dependencies in L, κ_κ, and $A_{\kappa_{\text{cvx}}}$, then we can be sure that in the convex setting we have

$$\text{dist}(0, \partial f_{\kappa_{\text{cvx}}}(\tilde{x}_k; y_k)) < \frac{\kappa_{\text{cvx}}}{k + 1} \|\tilde{x}_k - y_k\|. \tag{8}$$

This relative stationarity of $\tilde{x}_k$, including the choice of κ_{cvx}, shall be crucial to guarantee that the scheme accelerates in the convex setting. An additional $k + 1$ factor appears compared to the previous adaptive stationary condition because we need higher accuracy for solving the subproblem to achieve the accelerated rate in $1/\sqrt{\varepsilon}$. Therefore, an extra $\log(k + 1)$ factor of iterations is needed; see Sec. 4 and Sec. 5 in [28].

We shall see, in Sec. 6, that our strategy consisting in predefining T and S works quite well in practice.

The worst-case theoretical bounds we derive are conservative; we observe in our experiments that one may choose T and S significantly smaller than the theory suggests and still retain the stopping criteria.

5 Global Convergence and Applications to Existing Algorithms

After presenting the main mechanisms of our algoritm, we now present its worst-case complexity, which takes into account the cost of approximately solving the subproblems (2) and (4).

Theorem 5.1 (Global complexity bounds for 4WD–Catalyst)**.** *Choose* $T = \tilde{O}(\tau_L^{-1})$ *and* $S = \tilde{O}(\tau_{\kappa_{cvx}}^{-1})$ *(see Theorem 5.6 in [28]). Then the following are true.*

1. *Algorithm 1 generates a point* x *satisfying* $\text{dist}\big(0, \partial f(x)\big) \leq \varepsilon$ *after at most*

$$\tilde{O}\left(\left(\tau_L^{-1} + \tau_{\kappa_{cvx}}^{-1}\right) \cdot \frac{L(f(x_0) - f^*)}{\varepsilon^2} \right)$$

   iterations of the method $\mathcal{M}$.
2. *If* f *is convex, Algorithm 1 generates a point* x *satisfying* $\text{dist}\big(0, \partial f(x)\big) \leq \varepsilon$ *after at most*

$$\tilde{O}\left(\left(\tau_L^{-1} + \tau_{\kappa_{cvx}}^{-1}\right) \cdot \frac{L^{1/3}\left(\kappa_{cvx}\|x^* - x_0\|^2\right)^{1/3}}{\varepsilon^{2/3}} \right)$$

   iterations of the method $\mathcal{M}$.
3. *If* f *is convex, Algorithm 1 generates a point* x *satisfying* $f(x) - f^* \leq \varepsilon$ *after at most*

$$\tilde{O}\left(\left(\tau_L^{-1} + \tau_{\kappa_{cvx}}^{-1}\right) \cdot \frac{\sqrt{\kappa_{cvx}\|x^* - x_0\|^2}}{\sqrt{\varepsilon}} \right)$$

   iterations of the method $\mathcal{M}$.
   Here $\tilde{O}$ *hides universal constants and logarithmic dependencies in* $A_{\kappa_{cvx}}$, A_L, κ_0, κ_{cvx}, ε, *and* $\|x^* - x_0\|^2$.

If $\mathcal{M}$ is a first order method, the convergence guarantee in the convex setting is *near-optimal*, up to logarithmic factors, when compared to $O(1/\sqrt{\varepsilon})$ [22, 33]. In the nonconvex setting, our approach matches, up to logarithmic factors, the best known rate for this class of functions, namely $O(1/\varepsilon^2)$ [5, 7, 8]. Moreover, our rates dependence on the dimension and Lipschitz constant equals, up to log factors, the best known dependencies in both the convex and nonconvex setting. These logarithmic factors may be the price we pay for having a generic algorithm.

Choice of κ_{cvx}. The parameter κ_{cvx} drives the convergence rate of 4WD-Catalyst in the convex setting. To determine κ_{cvx}, we compute the global complexity

of our scheme as if $\rho = 0$, hence using the same reasoning as [22]. The rule consists in maximizing the ratio $\tau_\kappa/\sqrt{\kappa}$. Then, the choice of κ_0 is independent of $\mathcal{M}$; it is an initial lower estimate for the weak convexity constant ρ. We provide a detailed derivation of all the variables for each of the considered algorithms in Section 6 of [28].

5.1 Applications

We now compare the guarantees obtained before and after applying 4WD-Catalyst to some specific optimization methods $\mathcal{M}$: full gradient, SAGA, and SVRG (see [28] for randomized coordinate descent). In the convex setting, the accuracy is stated in terms of optimization error, $f(x) - f^* \leq \varepsilon$ and in the nonconvex setting, in terms of stationarity condition $\text{dist}(0, \partial f(x)) < \varepsilon$.

Full gradient method. First, we consider the simplest case of applying our method to the full gradient method (FG). Here, the optimal choice for κ_{cvx} is L. In the convex setting, we get the accelerated rate $O(n\sqrt{L/\varepsilon} \log(1/\varepsilon))$ which is consistent with Nesterov's accelerated variant (AFG) up to log factors. In the nonconvex case, our approach achieves no worse rate than $O(nL/\varepsilon^2 \log(1/\varepsilon))$, which is consistent with the standard gradient descent up to log factors. Under stronger assumptions, namely C^2-smoothness of the objective, the accelerated algorithm in [6] can achieve the same rate as (AFG) in the convex setting and $O(\varepsilon^{-7/4} \log(1/\varepsilon))$ in the nonconvex setting. Their approach, however, does not extend to composite setting nor to stochastic methods. The marginal loss may be the price for considering a larger class of functions.

Randomized incremental gradient. We now consider randomized incremental gradient methods such as SAGA [9] and (prox) SVRG [34]. Here, the optimal choice for κ_{cvx} is $O(L/n)$. Under the convex setting, we achieve an accelerated rate of $O(\sqrt{n}\sqrt{L/\varepsilon} \log(1/\varepsilon))$. Direct applications of SVRG and SAGA have no convergence guarantees in the nonconvex setting, but with our approach, the resulting algorithm matches the guarantees of FG up to log factors, see Table 1 for details.

6 Experiments

We investigate the performance of 4WD-Catalyst on two standard nonconvex problems in machine learning, namely on sparse matrix factorization and on training a simple two-layer neural network.

Comparison with linearly convergent methods. We report experimental results of 4WD-Catalyst when applied to the incremental algorithms SVRG [34] and SAGA [9], and consider the following variants:

- ncvx-SVRG/SAGA [2, 30] with its theoretical step-size $\eta = 1/Ln^{2/3}$.
- a minibatch variant of ncvx-SVRG/SAGA [2, 30] with batch size $b = n^{2/3}$ and stepsize $\eta = 1/L$.
- SVRG/SAGA with large stepsize $\eta = 1/L$. This is a variant of SVRG/SAGA, whose stepsize is not justified by theory for nonconvex problems, but which performs well in practice.
- 4WD-Catalyst SVRG/SAGA with its theoretical stepsize $\eta = 1/2L$.

The algorithm SVRG (resp. SAGA) was originally designed for minimizing convex objectives. The non-convex version was developed in [2, 30], using a significantly smaller stepsize $\eta = 1/Ln^{2/3}$. Following [30], we also include in the comparison a heuristic variant that uses a large stepsize $\eta = 1/L$, where no theoretical guarantee is available for nonconvex objectives. 4WD-Catalyst SVRG and 4WD-Catalyst SAGA use a similar stepsize, but the Catalyst mechanism makes this choice theoretically grounded.

Comparison with popular stochastic algorithms. We also include as baselines three popular stochastic algorithms: stochastic gradient descent (SGD), Ada-Grad [11], and Adam [17].

- SGD with constant stepsize.
- AdaGrad [11] with stepsize $\eta = 0.1$ or 0.01.
- Adam [17] with stepsize $\alpha = 0.01$ or 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

The stepsize (learning rate) of these algorithms are manually tuned to output the best performance. Note that none of them, SGD, AdaGrad [11] or Adam [17] enjoys linear convergence when the problem is strongly convex. Therefore, we do not apply 4WD-Catalyst to these algorithms. SGD is used in both experiments, whereas AdaGrad and Adam are used only on the neural network experiments since it is unclear how to apply them to a nonsmooth objective.

Parameter settings. We start from an initial estimate of the Lipschitz constant L and use the theoretically recommended parameters $\kappa_0 = \kappa_{\text{cvx}} = 2L/n$ in 4WD-Catalyst. We set the number of inner iterations $T = S = n$ in all experiments which means making at most one pass over the data to solve each sub-problem. Moreover, the $\log(k)$ dependency dictated by the theory is dropped while solving the subproblem in (6). These choices turn out to be justified *a posteriori*, as both SVRG and SAGA have a much better convergence rate in practice than the theoretical rate derived from a worst-case analysis. Indeed, in all experiments, one pass over the data to solve each sub-problem was found to be enough to guarantee sufficient descent. We focus in the main text on the results for SVRG and relegate

results for SAGA and details about experiments to Section 7 of [28].

Sparse matrix factorization a.k.a. dictionary learning. Dictionary learning consists of representing a dataset $X = [x_1, \cdots, x_n] \in \mathbb{R}^{m \times n}$ as a product $X \approx DA$, where D in $\mathbb{R}^{m \times p}$ is called a dictionary, and A in $\mathbb{R}^{p \times n}$ is a sparse matrix. The classical nonconvex formulation [see 24] can be reformulated as the equivalent finite-sum problem $\min_{D \in \mathcal{C}} \frac{1}{n} \sum_{i=1}^n f_i(D)$ with

$$f_i(D) := \min_{\alpha \in \mathbb{R}^p} \frac{1}{2} \|x_i - D\alpha\|_2^2 + \psi(\alpha). \qquad (9)$$

ψ is a sparsity-inducing regularization and $\mathcal{C}$ is chosen as the set of matrices whose columns are in the ℓ_2-ball; see Sec. 7 in [28]. We consider elastic-net regularization $\psi(\alpha) = \frac{\mu}{2}\|\alpha\|^2 + \lambda\|\alpha\|_1$ of [35], which has a sparsity-inducing effect, and report the corresponding results in Figure 2. We learn a dictionary in $\mathbb{R}^{m \times p}$ with $p = 256$ elements on a set of whitened normalized image patches of size $m = 8 \times 8$. Parameters are set to be as in [24]— that is, a small value $\mu = 1e-5$, and $\lambda = 0.25$, leading to sparse matrices A (on average ≈ 4 non-zero coefficients per column of A).

Neural networks. We consider simple binary classification problems for learning neural networks. Assume that we are given a training set $\{a_i, b_i\}_{i=1}^n$, where the variables b_i in $\{-1, +1\}$ represent class labels, and a_i in $\mathbb{R}^p$ are feature vectors. The estimator of a label class is now given by a two-layer neural network $\hat{b} = \text{sign}(w_2^\top \sigma(W_1^\top a))$, where W_1 in $\mathbb{R}^{p \times d}$ represents the weights of a hidden layer with d neurons, w_2 in $\mathbb{R}^d$ carries the weight of the network's second layer, and $\sigma(u) = \log(1 + e^u)$ is a non-linear function, applied point-wise to its arguments. We use the logistic loss to fit the estimators to the true labels and report experimental results on the two datasets alpha and covtype. The weights of the network are randomly initialized and we fix the number of hidden neurons to $d = 100$.

Computational cost. For SGD, AdaGrad, Adam, and all the ncvx-SVRG variants, one iteration corresponds to one pass over the data in the plots. For 4WD-Catalyst-SVRG, it solves two sub-problems per iteration, which doubles the cost per iteration comparing to the other algorithms. It is worth remarking that every time acceleration occurs in our experiments, $\tilde{x}_k$ is almost always preferred to $\bar{x}_k$ in step 4 of 4WD-Catalyst, suggesting that half of the computations may be reduced when running 4WD-Catalyst-SVRG.

Experimental conclusions. In the matrix factorization experiments in Fig. 2, 4WD-Catalyst-SVRG was always competitive, with a similar performance to the heuristic SVRG-$\eta = 1/L$ in two cases out of three, while being significantly better as soon as the
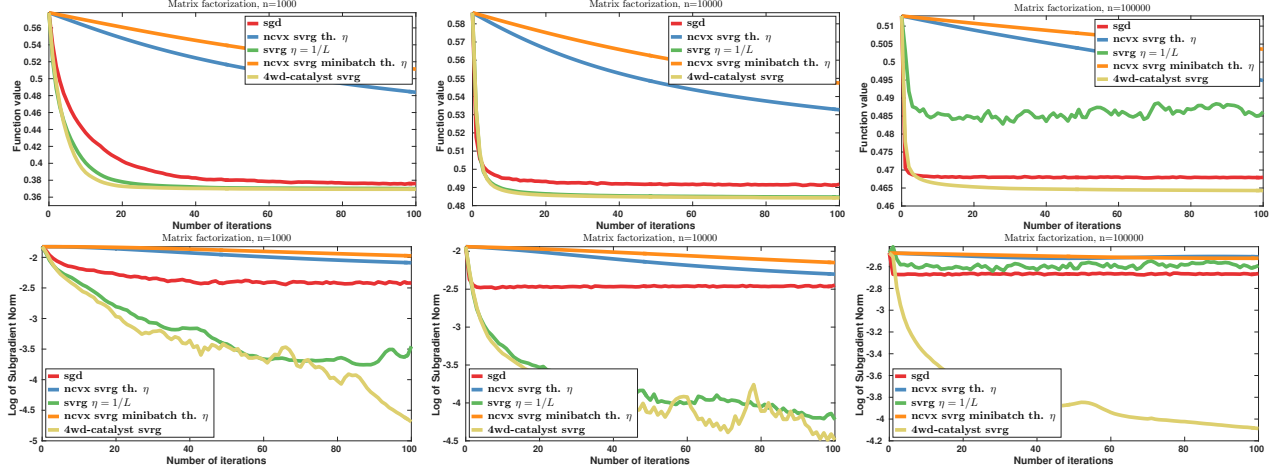
Figure 2: Dictionary learning experiments. We plot the function value (top) and the subgradient norm (bottom). From left to right, we vary the size of the dataset from $n = 1\,000$ to $n = 100\,000$
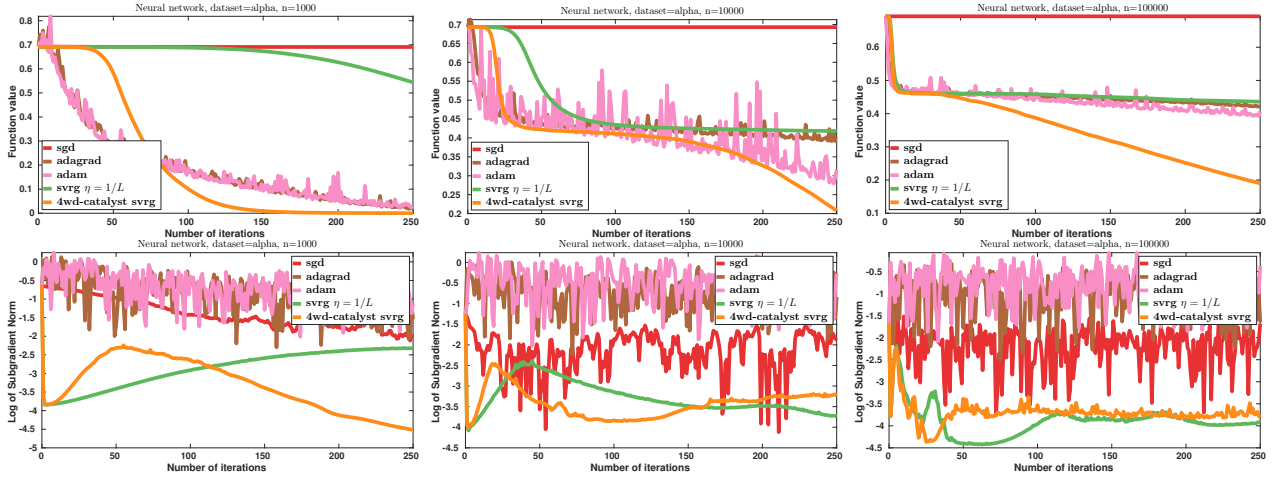

Figure 3: Neural network experiments. Same experimental setup as in Fig. 2. From left to right, we vary the size of the dataset's subset from $n = 1\,000$ to $n = 100\,000$.

amount of data n was large enough. As expected, the variants of SVRG with theoretical stepsizes have slow convergence, but exhibit a stable behavior compared to SVRG-$\eta = 1/L$. This confirms the ability of 4WD-Catalyst-SVRG to adapt to nonconvex terrains. Similar conclusions hold when applying 4WD-Catalyst to SAGA; see Sec. 7 in [28].

In the neural network experiments, we observe that 4WD-Catalyst-SVRG converges much faster overall in terms of objective values than other algorithms. Yet Adam and AdaGrad often perform well during the first iterations, they oscillate a lot, which is a behavior commonly observed. In contrast, 4WD-Catalyst-SVRG always decreases and keeps decreasing while other algorithms tend to stabilize, hence achieving significantly lower objective values.

More interestingly, as the algorithm proceeds, the subgradient norm may increase at some point and then

decrease, while the function value keeps decreasing. This suggests that the extrapolation step, or the Auto-adapt procedure, is helpful to escape bad stationary points, e.g., saddle-points. We leave the study of this particular phenomenon as a potential direction for future work.

Acknowledgements

References

[1] Z. Allen-Zhu. Natasha: Faster non-convex stochastic optimization via strongly non-convex parameter. In *International conference on machine learning (ICML)*, 2017.

[2] Z. Allen-Zhu and E. Hazan. Variance reduction for faster non-convex optimization. In *International conference on machine learning (ICML)*, 2016.

[3] J. M. Borwein and A. S. Lewis. *Convex analysis and nonlinear optimization: theory and examples.* Springer Verlag, 2006.

[4] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford. Accelerated methods for non-convex optimization. *preprint arXiv:1611.00756*, 2016.

[5] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford. Lower bounds for finding stationary points I. *preprint arXiv:1710.11606*, 2017.

[6] Y. Carmon, O. Hinder, J. C. Duchi, and A. Sidford. "convex until proven guilty": Dimension-free acceleration of gradient descent on non-convex functions. In *International conference on machine learning (ICML)*, 2017.

[7] C. Cartis, N. I. M. Gould, and P. L. Toint. On the complexity of steepest descent, newton's and regularized newton's methods for nonconvex unconstrained optimization problems. *SIAM Journal on Optimization*, 20(6):2833–2852, 2010.

[8] C. Cartis, N.I.M. Gould, and P. L. Toint. On the complexity of finding first-order critical points in constrained nonlinear optimization. *Mathematical Programming, Series A*, 144:93–106, 2014.

[9] A. J. Defazio, F. Bach, and S. Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[10] D. Drusvyatskiy and C. Paquette. Efficiency of minimizing compositions of convex functions and smooth maps. *preprint arXiv:1605.00125*, 2016.

[11] J. C. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research (JMLR)*, 12:2121–2159, 2011.

[12] S. Ghadimi and G. Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1-2, Ser. A):59–99, 2016.

[13] S. Ghadimi, G. Lan, and H. Zhang. Generalized uniformly optimal methods for nonlinear programming. *preprint arXiv:1508.07384*, 2015.

[14] T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical learning with sparsity: the Lasso and generalizations.* CRC Press, 2015.

[15] C. Jin, P. Netrapalli, and M. I. Jordan. Accelerated gradient descent escapes saddle points faster than gradient descent. *preprint arXiv:1711.10456*, 2017.

[16] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[18] G. Lan and Y. Zhou. An optimal randomized incremental gradient method. *Mathematical Programming, Series A*, pages 1–38, 2017.

[19] L. Lei and M. I. Jordan. Less than a single pass: stochastically controlled stochastic gradient method. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[20] L. Lei, C. Ju, J. Chen, and M. I. Jordan. Nonconvex finite-sum optimization via SCSG methods. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[21] H. Li and Z. Lin. Accelerated proximal gradient methods for nonconvex programming. In *Advances in Neural Information Processing Systems (NIPS)*. 2015.

[22] H. Lin, J. Mairal, and Z. Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[23] J. Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.

[24] J. Mairal, F. Bach, and J. Ponce. Sparse modeling for image and vision processing. *Foundations and Trends in Computer Graphics and Vision*, 8(2-3):85–283, 2014.

[25] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

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

[27] M. O'Neill and S. J. Wright. Behavior of accelerated gradient methods near critical points of nonconvex problems. *preprint arXiv:1706.07993*, 2017.

[28] C. Paquette, H. Lin, D. Drusvyatskiy, J. Mairal, and Z. Harchaoui. Catalyst acceleration for gradient-based non-convex optimization. *preprint arXiv:1703.10993*, 2017.

[29] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola. Stochastic variance reduction for non-convex optimization. In *International conference on machine learning (ICML)*, 2016.

[30] S. J. Reddi, S. Sra, B. Poczos, and A. J. Smola. Proximal stochastic methods for nonsmooth non-convex finite-sum optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[31] R. T. Rockafellar and R. J.-B. Wets. *Variational analysis*, volume 317 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1998.

[32] M. Schmidt, N. Le Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1):83–112, 2017.

[33] B. E. Woodworth and N. Srebro. Tight complexity bounds for optimizing composite objectives. In *Advances in Neural Information Processing Systems (NIPS)*. 2016.

[34] L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

[35] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.