“Convex Until Proven Guilty”: Dimension-Free Acceleration of Gradient Descent on Non-Convex Functions

Yair Carmon John C. Duchi Oliver Hinder Aaron Sidford [1]

Abstract

We develop and analyze a variant of Nesterov’s accelerated gradient descent (AGD) for minimization of smooth non-convex functions. We prove that one of two cases occurs: either our AGD variant converges quickly, as if the function was convex, or we produce a certificate that the function is “guilty” of being non-convex. This non-convexity certificate allows us to exploit negative curvature and obtain deterministic, dimension-free acceleration of convergence for non-convex functions. For a function f with Lipschitz continuous gradient and Hessian, we compute a point x with $\|\nabla f(x)\| \le \epsilon$ in $O(\epsilon^{-7/4}\log(1/\epsilon))$ gradient and function evaluations. Assuming additionally that the third derivative is Lipschitz, we require only $O(\epsilon^{-5/3}\log(1/\epsilon))$ evaluations.

1. Introduction

Nesterov’s seminal 1983 accelerated gradient method has inspired substantial development of first-order methods for large-scale convex optimization. In recent years, machine learning and statistics have seen a shift toward large scale *non-convex* problems, including methods for matrix completion (Koren et al., 2009), phase retrieval (Candès et al., 2015; Wang et al., 2016), dictionary learning (Mairal et al., 2008), and neural network training (LeCun et al., 2015). In practice, techniques from accelerated gradient methods—namely, momentum—can have substantial benefits for stochastic gradient methods, for example, in training neural networks (Rumelhart et al., 1986; Kingma and Ba, 2015). Yet little of the rich theory of acceleration for convex optimization is known to transfer into non-convex optimization.

Optimization becomes more difficult without convexity, as gradients no longer provide global information about the function. Even determining if a stationary point is a local minimum is (generally) NP-hard (Murty and Kabadi, 1987; Nesterov, 2000). It is, however, possible to leverage non-convexity to improve objectives in smooth optimization: moving in directions of negative curvature can guarantee function value reduction. We explore the interplay between negative curvature, smoothness, and acceleration techniques, showing how an understanding of the three simultaneously yields a method that provably accelerates convergence of gradient descent for a broad class of non-convex functions.

1.1. Problem setting

We consider the unconstrained minimization problem

$$\underset{x}{\text{minimize}}\, f(x), \tag{1}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is smooth but potentially non-convex. We assume throughout the paper that f is bounded from below, two-times differentiable, and has Lipschitz continuous gradient and Hessian. In Section 4 we strengthen our results under the additional assumption that f has Lipschitz continuous third derivatives. Following the standard first-order oracle model (Nemirovski and Yudin, 1983), we consider optimization methods that access only values and gradients of f (and not higher order derivatives), and we measure their complexity by the total number of gradient and function evaluations.

Approximating the global minimum of f to ϵ-accuracy is generally intractable, requiring time exponential in $d \log \frac{1}{\epsilon}$ (Nemirovski and Yudin, 1983, §1.6). Instead, we seek a point x that is ϵ-approximately stationary, that is,

$$\|\nabla f(x)\| \le \epsilon. \tag{2}$$

Finding stationary points is a canonical problem in nonlinear optimization (Nocedal and Wright, 2006), and while saddle points and local maxima are stationary, excepting pathological cases, descent methods that converge to a stationary point converge to a local minimum (Lee et al., 2016; Nemirovski, 1999, §3.2.2).

[1]Stanford University, Stanford, California, USA. Correspondence to: Yair Carmon <yairc@stanford.edu>, Oliver Hinder <ohinder@stanford.edu>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

If we assume f is convex, gradient descent satisfies the bound (2) after $O(\epsilon^{-1})$ gradient evaluations, and AGD improves this rate to $O(\epsilon^{-1/2}\log\frac{1}{\epsilon})$ (Nesterov, 2012). Without convexity, gradient descent is significantly worse, having worst-case complexity $\Theta(\epsilon^{-2})$ (Cartis et al., 2010). More sophisticated gradient-based methods, including nonlinear conjugate gradient (Hager and Zhang, 2006) and L-BFGS (Liu and Nocedal, 1989) provide excellent practical performance, but their global convergence guarantees are no better than $O(\epsilon^{-2})$. Our work (Carmon et al., 2016) and, independently, Agarwal et al. (2016), break this $O(\epsilon^{-2})$ barrier, obtaining the rate $O(\epsilon^{-7/4}\log\frac{d}{\epsilon})$. Before we discuss this line of work in Section 1.3, we overview our contributions.

1.2. Our contributions

"Convex until proven guilty" Underpinning our results is the observation that when we run Nesterov's accelerated gradient descent (AGD) on *any* smooth function f, one of two outcomes must follow:

(a) AGD behaves as though f was σ-strongly convex, satisfying inequality (2) in $O(\sigma^{-1/2}\log\frac{1}{\epsilon})$ iterations.

(b) There exist points u, v in the AGD trajectory that prove f is "guilty" of not being σ-strongly convex,

$$f(u) < f(v) + \nabla f(v)^T(u-v) + \frac{\sigma}{2}\|u-v\|^2 . \quad (3)$$

The intuition behind these observations is that if inequality (3) never holds during the iterations of AGD, then f "looks" strongly convex, and the convergence (a) follows. In Section 2 we make this observation precise, presenting an algorithm to monitor AGD and quickly find the witness pair u, v satisfying (3) whenever AGD progresses more slowly than it does on strongly convex functions. We believe there is potential to apply this strategy beyond AGD, extending additional convex gradient methods to non-convex settings.

An accelerated non-convex gradient method In Section 3 we propose a method that iteratively applies our monitored AGD algorithm to f augmented by a proximal regularizer. We show that both outcomes (a) and (b) above imply progress minimizing f, where in case (b) we make explicit use of the negative curvature that AGD exposes. These progress guarantees translate to an overall first-order oracle complexity of $O(\epsilon^{-7/4}\log\frac{1}{\epsilon})$, a strict improvement over the $O(\epsilon^{-2})$ rate of gradient descent. In Section 5 we report preliminary experimental results, showing a basic implementation of our method outperforms gradient descent but not nonlinear conjugate gradient.

Improved guarantees with third-order smoothness As we show in Section 4, assuming Lipschitz continuous third derivatives instead of Lipschitz continuous Hessian allows us to increase the step size we take when exploiting negative curvature, making more function progress. Consequently, the complexity of our method improves to $O(\epsilon^{-5/3}\log\frac{1}{\epsilon})$. While the analysis of the third-order setting is more complex, the method remains essentially unchanged. In particular, we still use only first-order information, never computing higher-order derivatives.

1.3. Related work

Nesterov and Polyak (2006) show that cubic regularization of Newton's method finds a point that satisfies the stationarity condition (2) in $O(\epsilon^{-3/2})$ evaluations of the Hessian. Given sufficiently accurate arithmetic operations, a Lipschitz continuous Hessian is approximable to arbitrary precision using finite gradient differences, and obtaining a full Hessian requires $O(d)$ gradient evaluations. A direct implementation of the Nesterov-Polyak method with a first-order oracle therefore has gradient evaluation complexity $O(\epsilon^{-3/2}d)$, improving on gradient descent only if $d \ll \epsilon^{-1/2}$, which may fail in high-dimensions.

In two recent papers, we (Carmon et al., 2016) and (independently) Agarwal et al. obtain better rates for first-order methods. Agarwal et al. (2016) propose a careful implementation of the Nesterov-Polyak method, using accelerated methods for fast approximate matrix inversion. In our earlier work, we employ a combination of (regularized) accelerated gradient descent and the Lanczos method. Both find a point that satisfies the bound (2) with probability at least $1-\delta$ using $O\left(\epsilon^{-7/4}\log\frac{d}{\delta\epsilon}\right)$ gradient and Hessian-vector product evaluations.

The primary conceptual difference between our approach and those of Carmon et al. and Agarwal et al. is that we perform no eigenvector search: we automatically find directions of negative curvature whenever AGD proves f "guilty" of non-convexity. Qualitatively, this shows that explicit second orders information is unnecessary to improve upon gradient descent for stationary point computation. Quantitatively, this leads to the following improvements:

(i) Our result is *dimension-free and deterministic*, with complexity independent of the ratio d/δ, compared to the $\log\frac{d}{\delta}$ dependence of previous works. This is significant, as $\log\frac{d}{\delta}$ may be comparable to $\epsilon^{-1/4}/\log\frac{1}{\epsilon}$.

(ii) Our method uses *only gradient evaluations*, and does not require Hessian-vector products. In practice, Hessian-vector products may be difficult to implement and more expensive to compute than gradients.

(iii) Under third-order smoothness assumptions we improve our method to achieve $O(\epsilon^{-5/3} \log \frac{1}{\epsilon})$ rate. It is unclear how to extend previous approaches to obtain similar guarantees.

In distinction from the methods of Carmon et al. (2016) and Agarwal et al. (2016), our method provides no guarantees on positive definiteness of $\nabla^2 f(x)$; if initialized at a saddle point it will terminate immediately. However, as we further explain in Section D, we may combine our method with a fast eigenvector search to recover the approximate positive definiteness guarantee $\nabla^2 f(x) \succeq -\sqrt{\epsilon} I$, even improving it to $\nabla^2 f(x) \succeq -\epsilon^{2/3} I$ using third-order smoothness, but at the cost of reintroducing randomization, Hessian-vector products and a $\log \frac{d}{\delta}$ complexity term.

1.4. Preliminaries and notation

Here we introduce notation and briefly overview definitions and results we use throughout. We index sequences by subscripts, and use x_i^j as shorthand for $x_i, x_{i+1}, ..., x_j$. We use x, y, v, u, w, p, c, q and z to denote points in $\mathbb{R}^d$. Additionally, η denotes step sizes, ϵ, ε denote desired accuracy, θ denotes a scalar and $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^d$. We denote the nth derivative of a function $h : \mathbb{R} \to \mathbb{R}$ by $h^{(n)}$. We let $\log_+(t) = \max\{0, \log t\}$.

A function $f : \mathbb{R}^d \to \mathbb{R}$ has L_n-Lipschitz nth derivative if it is n times differentiable and for every x_0 and unit vector δ, the one-dimensional function $h(\theta) = f(x_0 + \theta\delta)$ satisfies

$$\left| h^{(n)}(\theta_1) - h^{(n)}(\theta_2) \right| \le L_n |\theta_1 - \theta_2|.$$

We refer to this property as nth-order smoothness, or simply smoothness for $n = 1$, where it coincides with the Lipschitz continuity of ∇f. Throughout the paper, we make extensive use of the well-known consequence of Taylor's theorem, that the Lipschitz constant of the nth-order derivative controls the error in the nth order Taylor series expansion of h, *i.e.* for $\theta, \theta_0 \in \mathbb{R}$ we have

$$\left| h(\theta) - \sum_{i=0}^{n} \frac{1}{i!} h^{(i)}(\theta_0)(\theta - \theta_0)^i \right| \le \frac{L_n}{(n+1)!} |\theta - \theta_0|^{n+1}. \tag{4}$$

A function f is σ-strongly convex if $f(u) \ge f(v) + \nabla f(v)^T (u - v) + \frac{\sigma}{2} \|u - v\|^2$ for all $v, u \in \mathbb{R}^d$.

2. Algorithm components

We begin our development by presenting the two building blocks of our result: a monitored variation of AGD (Section 2.1) and a negative curvature descent step (Section 2.2) that we use when the monitored version of AGD certifies non-convexity. In Section 3, we combine these components to obtain an accelerated method for non-convex functions.

Algorithm 1 AGD-UNTIL-GUILTY($f, y_0, \varepsilon, L, \sigma$)

```
1: Set $\kappa \leftarrow L/\sigma$, $\omega \leftarrow \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$ and $x_0 \leftarrow y_0$
2: for $t = 1, 2, \ldots$ do
3:     $y_t \leftarrow x_{t-1} - \frac{1}{L}\nabla f(x_{t-1})$
4:     $x_t \leftarrow y_t + \omega(y_t - y_{t-1})$
5:     $w_t \leftarrow$ CERTIFY-PROGRESS($f, y_0, y_t, L, \sigma, \kappa$)
6:     if $w_t \ne$ NULL then                          ▷ convexity violation
7:         $(u, v) \leftarrow$ FIND-WITNESS-PAIR($f, x_0^t, y_0^t, w_t, \sigma$)
8:         return $(x_0^t, y_0^t, u, v)$
9:     if $\|\nabla f(y_t)\| \le \varepsilon$ then return $(x_0^t, y_0^t, \text{NULL})$

1: function CERTIFY-PROGRESS($f, y_0, y_t, L, \sigma, \kappa$)
2:     if $f(y_t) > f(y_0)$ then
3:         return $y_0$                                ▷ non-convex behavior
4:     Set $z_t \leftarrow y_t - \frac{1}{L}\nabla f(y_t)$
5:     Set $\psi(z_t) \leftarrow f(y_0) - f(z_t) + \frac{\sigma}{2}\|z_t - y_0\|^2$
6:     if $\|\nabla f(y_t)\|^2 > 2L\psi(z_t)e^{-t/\sqrt{\kappa}}$ then
7:         return $z_t$                                ▷ AGD has stalled
8:     else return NULL

1: function FIND-WITNESS-PAIR($f, x_0^t, y_0^t, w_t, \sigma$)
2:     for $j = 0, 1, \ldots, t-1$ do
3:         for $u = y_j, w_t$ do
4:             if eq. (8) holds with $v = x_j$ then
5:                 return $(u, x_j)$
6:     (by Corollary 1 this line is never reached)
```

2.1. AGD as a convexity monitor

The main component in our approach is Alg. 1, AGD-UNTIL-GUILTY. We take as input an L-smooth function f, conjectured to be σ-strongly convex, and optimize it with Nesterov's accelerated gradient descent method for strongly convex functions (lines 3 and 4). At every iteration, the method invokes CERTIFY-PROGRESS to test whether the optimization is progressing as it should for strongly convex functions, and in particular that the gradient norm is decreasing exponentially quickly (line 6). If the test fails, FIND-WITNESS-PAIR produces points u, v proving that f violates σ-strong convexity. Otherwise, we proceed until we find a point y such that $\|\nabla f(y)\| \le \varepsilon$.

The efficacy of our method is based on the following guarantee on the performance of AGD.

Proposition 1. *Let f be L-smooth, and let y_0^t and x_0^t be the sequence of iterates generated by* AGD-UNTIL-GUILTY*(f, y_0, L, ε, σ) for some $\varepsilon > 0$ and $0 < \sigma \le L$. Fix $w \in \mathbb{R}^d$. If for $s = 0, 1, \ldots, t-1$ we have*

$$f(u) \ge f(x_s) + \nabla f(x_s)^T (u - x_s) + \frac{\sigma}{2}\|u - x_s\|^2 \tag{5}$$

for both $u = w$ and $u = y_s$, then

$$f(y_t) - f(w) \le \left(1 - \frac{1}{\sqrt{\kappa}}\right)^t \psi(w), \tag{6}$$

where $\kappa = \frac{L}{\sigma}$ and $\psi(w) = f(y_0) - f(w) + \frac{\sigma}{2}\|w - y_0\|^2$.

Proposition 1 is essentially a restatement of established results (Nesterov, 2004; Bubeck, 2014), where we take care to phrase the requirements on f in terms of local inequalities, rather than a global strong convexity assumption. For completeness, we provide a proof of Proposition 1 in Section A.1 in the supplementary material.

With Proposition 1 in hand, we summarize the guarantees of Alg. 1 as follows.

Corollary 1. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be L-smooth, let $y_0 \in \mathbb{R}^d$, $\varepsilon > 0$ and $0 < \sigma \le L$. Let $(x_0^t, y_0^t, u, v) =$ AGD-UNTIL-GUILTY$(f,\ y_0,\ \varepsilon,\ L,\ \sigma)$. Then the number of iterations t satisfies*

$$t \le 1 + \max\left\{0, \sqrt{\frac{L}{\sigma}} \log\left(\frac{2L\psi(z_{t-1})}{\varepsilon^2}\right)\right\}, \tag{7}$$

where $\psi(z) = f(y_0) - f(z) + \frac{\sigma}{2}\|z - y_0\|^2$ is as in line 5 of CERTIFY-PROGRESS. *If $u, v \ne$ NULL (non-convexity was detected), then*

$$f(u) < f(v) + \nabla f(v)^T(u - v) + \frac{\sigma}{2}\|u - v\|^2 \tag{8}$$

where $v = x_j$ for some $0 \le j < t$ and $u = y_j$ or $u = w_t$ (defined on line 5 of AGD-UNTIL-GUILTY*). Moreover,*

$$\max\{f(y_1), \ldots, f(y_{t-1}), f(u)\} \le f(y_0). \tag{9}$$

Proof. The bound (7) is clear for $t = 1$. For $t > 1$, the algorithm has not terminated at iteration $t - 1$, and so we know that neither the condition in line 9 of AGD-UNTIL-GUILTY nor the condition in line 6 of CERTIFY-PROGRESS held at iteration $t - 1$. Thus

$$\varepsilon^2 < \|\nabla f(y_{t-1})\|^2 \le 2L\psi(z_{t-1})e^{-(t-1)/\sqrt{\kappa}},$$

which gives the bound (7) when rearranged.

Now we consider the returned vectors x_0^t, y_0^t, u, and v from AGD-UNTIL-GUILTY. Note that $u, v \ne$ NULL only if $w_t \ne$ NULL. Suppose that $w_t = y_0$, then by line 2 of CERTIFY-PROGRESS we have,

$$f(y_t) - f(w_t) > 0 = \left(1 - \frac{1}{\sqrt{\kappa}}\right)^t \psi(w_t),$$

since $\psi(w_t) = \psi(y_0) = 0$. Since this contradicts the progress bound (6), we obtain the certificate (8) by the contrapositive of Proposition 1: condition (5) must not hold for some $0 \le s < t$, implying FIND-WITNESS-PAIR will return for some $j \le s$.

Similarly, if $w_t = z_t = y_t - \frac{1}{L}\nabla f(y_t)$ then by line 6 of CERTIFY-PROGRESS we must have

$$\frac{1}{2L}\|\nabla f(y_t)\|^2 > \psi(z_t)e^{-t/\sqrt{\kappa}} \ge \left(1 - \frac{1}{\sqrt{\kappa}}\right)^t \psi(z_t).$$

Since f is L-smooth we have the standard progress guarantee (*c.f.* Nesterov (2004) §1.2.3) $f(y_t) - f(z_t) \ge \frac{1}{2L}\|\nabla f(y_t)\|^2$, again contradicting inequality (6).

To see that the bound (9) holds, note that $f(y_s) \le f(y_0)$ for $s = 0, \ldots, t-1$ since condition 2 of CERTIFY-PROGRESS did not hold. If $u = y_j$ for some $0 \le j < t$ then $f(u) \le f(y_0)$ holds trivially. Alternatively, if $u_t = w_t = z_t$ then condition 2 did not hold at time t as well, so we have $f(y_t) \le f(y_0)$ and also $f(u) = f(z_t) \le f(y_t) - \frac{1}{2L}\|\nabla f(y_t)\|^2$ as noted above; therefore $f(z_t) \le f(y_0)$. □

Before continuing, we make two remarks about implementation of Alg. 1.

(1) As stated, the algorithm requires evaluation of two function gradients per iteration (at x_t and y_t). Corollary 1 holds essentially unchanged if we execute line 9 of AGD-UNTIL-GUILTY and lines 4-6 of CERTIFY-PROGRESS only once every τ iterations, where τ is some fixed number (say 10). This reduces the number of gradient evaluations to $1 + \frac{1}{\tau}$ per iteration.

(2) Direct implementation would require $O(d \cdot t)$ memory to store the sequences y_0^t, x_0^t and $\nabla f(x_0^t)$ for later use by FIND-WITNESS-PAIR. Alternatively, FIND-WITNESS-PAIR can regenerate these sequences from their recursive definition while iterating over j, reducing the memory requirement to $O(d)$ and increasing the number of gradient and function evaluations by at most a factor of 2.

In addition, while our emphasis is on applying AGD-UNTIL-GUILTY to non-convex problems, the algorithm has implications for convex optimization. For example, we rarely know the strong convexity parameter σ of a given function f; to remedy this, O'Donoghue and Candès (2015) propose adaptive restart schemes. Instead, one may repeatedly apply AGD-UNTIL-GUILTY and use the witnesses to update σ.

2.2. Using negative curvature

The second component of our approach is exploitation of negative curvature to decrease function values; in Section 3

Algorithm 2 EXPLOIT-NC-PAIR(f, u, v, η)

```
1: $\delta \leftarrow (u-v)/\|u-v\|$
2: $u_+ \leftarrow u + \eta\delta$
3: $u_- \leftarrow u - \eta\delta$
4: **return** $\arg\min_{z\in\{u_-,u_+\}} f(z)$
```

we use AGD-UNTIL-GUILTY to generate u, v such that

$$f(u) < f(v) + \nabla f(v)^T(u-v) - \frac{\alpha}{2}\|u-v\|^2, \quad (10)$$

a nontrivial violation of convexity (where $\alpha > 0$ is a parameter we control using a proximal term). By taking an appropriately sized step from u in the direction $\pm(u-v)$, Alg. 2 can substantially lower the function value near u whenever the convexity violation (10) holds. The following basic lemma shows this essential progress guarantee.

Lemma 1. *Let $f : \mathbb{R}^d \to \mathbb{R}$ have L_2-Lipschitz Hessian. Let $\alpha > 0$ and let u and v satisfy (10). If $\|u-v\| \le \frac{\alpha}{2L_2}$, then for every $\eta \le \frac{\alpha}{L_2}$, EXPLOIT-NC-PAIR($f, u, v, \eta$) finds a point z such that*

$$f(z) \le f(u) - \frac{\alpha\eta^2}{12}. \quad (11)$$

We give the proof of Lemma 1 in Section A.2, and we outline it here. The proof is split into two parts, both using the Lipschitz continuity of $\nabla^2 f$. In the first part, we show using (10) that f has negative curvature of at least $\alpha/2$ in the direction of δ at the point u. In the second part, we consider the Taylor series expansion of f. The first order term predicts, due to its anti-symmetry, that either a step size of $-\eta$ or η in the direction δ reduces the objective. Adding our knowledge of the negative curvature from the first part yields the required progress.

3. Accelerating non-convex optimization

We now combine the accelerated convergence guarantee of Corollary 1 and the non-convex progress guarantee of Lemma 1 to form GUARDED-NON-CONVEX-AGD. The idea for the algorithm is as follows. Consider iterate $k-1$, denoted p_{k-1}. We create a proximal function $\hat{f}$ by adding the proximal term $\alpha\|x-p_{k-1}\|^2$ to f. Applying AGD-UNTIL-GUILTY to $\hat{f}$ yields the sequences $x_0, \ldots, x_t$, $y_0, \ldots, y_t$ and possibly a non-convexity witnessing pair u, v (line 3). If u, v are not available, we set $p_k = y_t$ and continue to the next iteration. Otherwise, by Corollary 1, u and v certify that $\hat{f}$ is not α strongly convex, and therefore that f has negative curvature. EXPLOIT-NC-PAIR then leverages this negative curvature, obtaining a point $b^{(2)}$. The next iterate p_k is the best out of $y_0, \ldots, y_t, u$ and $b^{(2)}$ in terms of function value.

Algorithm 3
GUARDED-NON-CONVEX-AGD(f, p_0, L_1, ϵ, α, η)

```
1: **for** $k = 1, 2, \ldots$ **do**
2:     Set $\hat{f}(x) := f(x) + \alpha\|x-p_{k-1}\|^2$
3:     $(x_0^t, y_0^t, u, v) \leftarrow$
           AGD-UNTIL-GUILTY($\hat{f}$, $p_{k-1}$, $\frac{\epsilon}{10}$, $L_1 + 2\alpha$, $\alpha$)
4:     **if** $u, v =$ NULL **then**
5:         $p_k \leftarrow y_t$                ▷ $\hat{f}$ effectively str. convex
6:     **else**                    ▷ non-convexity proof available
7:         $b^{(1)} \leftarrow$ FIND-BEST-ITERATE($f, y_0^t, u, v$)
8:         $b^{(2)} \leftarrow$ EXPLOIT-NC-PAIR($f, u, v, \eta$)
9:         $p_k \leftarrow \arg\min_{z\in\{b^{(1)},b^{(2)}\}} f(z)$
10:    **if** $\|\nabla f(p_k)\| \le \epsilon$ **then**
11:        **return** $p_k$

1: **function** FIND-BEST-ITERATE($f$, $y_0^t$, $u$, $v$)
2:     **return** $\arg\min_{z\in\{u,y_0,\ldots,y_t\}} f(z)$
```

The following central lemma provides a progress guarantee for each of the iterations of Alg. 3.

Lemma 2. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be L_1-smooth and have L_2-Lipschitz continuous Hessian, let $\epsilon, \alpha > 0$ and $p_0 \in \mathbb{R}^d$. Let $p_1, \ldots, p_K$ be the iterates GUARDED-NON-CONVEX-AGD(f, p_0, L_1, ϵ, α, $\frac{\alpha}{L_2}$) generates. Then for each $k \in \{1, \ldots, K-1\}$,*

$$f(p_k) \le f(p_{k-1}) - \min\left\{\frac{\epsilon^2}{5\alpha}, \frac{\alpha^3}{64L_2^2}\right\}. \quad (12)$$

We defer a detailed proof of Lemma 2 to Section B.1, and instead sketch the main arguments. Fix an iteration k of GUARDED-NON-CONVEX-AGD that is not the final one (*i.e.* $k < K$). Then, if $\hat{f}$ was effectively strongly convex we must have $\|\nabla\hat{f}(y_t)\| \le \epsilon/10$ and standard proximal point arguments show that we reduce the objective by $\epsilon^2/(5\alpha)$. Otherwise, a witness pair u, v is available for which (10) holds by Corollary 1, and $f(u) \le \hat{f}(u) \le \hat{f}(y_0) = f(p_k)$. To apply Lemma 1 it remains to show that $\|u-v\| \le \alpha/(2L_2)$. We note that, since $f(y_i) + \alpha\|y_i - y_0\|^2 = \hat{f}(y_i) \le f(y_0)$ for every $i < t$, if any iterate y_i is far from y_0, $f(y_i)$ must be substantially lower than $f(y_0)$, and therefore $b^{(1)}$ makes good progress. Formalizing the converse of this claim gives Lemma 3, which we prove Section B.2.

Lemma 3. *Let f be L_1-smooth, and $\tau \ge 0$. At any iteration of GUARDED-NON-CONVEX-AGD, if $u, v \ne$ NULL and the best iterate $b^{(1)}$ satisfies $f(b^{(1)}) \ge f(y_0) - \alpha\tau^2$ then for $1 \le i < t$,*

$$\|y_i - y_0\| \le \tau, \quad \text{and} \quad \|x_i - y_0\| \le 3\tau.$$

Consequently, $\|u-v\| \le 4\tau$.

Lemma 3 explains the role of $b^{(1)}$ produced by FIND-BEST-ITERATE: it is an "insurance policy" against $\|u-v\|$ being too large. To complete the proof of Lemma 2 we take $\tau = \frac{\alpha}{8L_2}$, so that either

$$f(b^{(1)}) \le f(y_0) - \alpha\tau^2 = f(y_0) - \frac{\alpha^3}{64L_2^2},$$

or we have $\|u - v\| \le 4\tau = \frac{\alpha}{2L_2}$ by Lemma 3, and therefore $f(b^{(2)}) \le f(y_0) - \frac{\alpha^3}{12L_2^2}$ by Lemma 1 (with $\eta = \alpha/L_2$).

Lemma 2 shows we can accelerate gradient descent in a non-convex setting. Indeed, ignoring all problem-dependent constants, setting $\alpha = \sqrt{\epsilon}$ in the bound (12) shows that we make $\Omega(\epsilon^{3/2})$ progress at every iteration of GUARDED-NON-CONVEX-AGD, and consequently the number of iterations is bounded by $O(\epsilon^{-3/2})$. Arguing that calls to AGD-UNTIL-GUILTY each require $O(\epsilon^{-1/4}\log\frac{1}{\epsilon})$ gradient computations yields the following complexity guarantee, which we prove in Section B.3.

Theorem 1. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be L_1-smooth and have L_2-Lipschitz continuous Hessian. Let $p_0 \in \mathbb{R}^d$, $\Delta_f = f(p_0) - \inf_{z\in\mathbb{R}^d} f(z)$ and $0 < \epsilon \le \min\{\Delta_f^{2/3}L_2^{1/3}, L_1^2/(64L_2)\}$. Set*

$$\alpha = 2\sqrt{L_2\epsilon} \tag{13}$$

then GUARDED-NON-CONVEX-AGD(f, p_0, L_1, ϵ, α, $\frac{\alpha}{L_2}$) finds a point p_K such that $\|\nabla f(p_K)\| \le \epsilon$ with at most

$$20 \cdot \frac{\Delta_f L_1^{1/2} L_2^{1/4}}{\epsilon^{7/4}} \log\frac{500L_1\Delta_f}{\epsilon^2} \tag{14}$$

gradient evaluations.

The conditions on ϵ simply guarantee that the clean bound (14) is non-trivial, as gradient descent yields better convergence guarantees for larger values of ϵ.

While we state Theorem 1 in terms of gradient evaluation count, a similar bound holds for function evaluations as well. Indeed, inspection of our method reveals that each iteration of Alg. 3 evaluates the function and not the gradient at at most the three points u, u_+ and u_-; both complexity measures are therefore of the same order.

4. Incorporating third-order smoothness

In this section, we show that when third-order derivatives are Lipschitz continuous, we can improve the convergence rate of Alg. 3 by modifying two of its subroutines. In Section 4.1 we introduce a modified version of EXPLOIT-NC-PAIR that can decrease function values further using third-order smoothness. In Section 4.2 we change FIND-BEST-ITERATE to provide a guarantee that $f(v)$ is never too large. We combine these two results in Section 4.3 and present our improved complexity bounds.

Algorithm 4 EXPLOIT-NC-PAIR$_3$(f, u, v, η)

```
1: $\delta \leftarrow (u - v)/\|u - v\|$
2: $\eta' \leftarrow \sqrt{\eta(\eta + \|u - v\|)} - \|u - v\|$
3: $u_+ \leftarrow u + \eta'\delta$
4: $v_- \leftarrow v - \eta\delta$
5: return $\arg\min_{z\in\{v_-,u_+\}} f(z)$
```

4.1. Making better use of negative curvature

Our first observation is that third-order smoothness allows us to take larger steps and make greater progress when exploiting negative curvature, as the next lemma formalizes.

Lemma 4. *Let $f : \mathbb{R}^d \to \mathbb{R}$ have L_3-Lipschitz third-order derivatives, $u \in \mathbb{R}^d$, and $\delta \in \mathbb{R}^d$ be a unit vector. If $\delta^T\nabla^2 f(u)\delta = -\frac{\alpha}{2} < 0$ then, for every $0 \le \eta \le \sqrt{3\alpha/L_3}$,*

$$\min\{f(u - \eta\delta), f(u + \eta\delta)\} \le f(u) - \frac{\alpha\eta^2}{8}. \tag{15}$$

Proof. For $\theta \in \mathbb{R}$, define $h(\theta) = f(u+\theta\delta)$. By assumption h''' is L_3-Lipschitz continuous, and therefore

$$h(\theta) \le h(0) + h'(0)\theta + h''(0)\frac{\theta^2}{2} + h'''(0)\frac{\theta^3}{6} + L_3\frac{\theta^4}{24}.$$

Set $A_\eta = h'(0)\eta + h'''(0)\eta^3/6$ and set $\bar{\eta} = -\text{sign}(A_\eta)\eta$. As $h'(0)\bar{\eta} + h'''(0)\bar{\eta}^3/6 = -|A_\eta| \le 0$, we have

$$h(\bar{\eta}) \le h(0) + h''(0)\frac{\eta^2}{2} + L_3\frac{\eta^4}{24} \le f(u) - \frac{\alpha\eta^2}{8},$$

the last inequality using $h(0) = f(u)$, $h''(0) = -\frac{\alpha}{2}$ and $\eta^2 \le \frac{3\alpha}{L_3}$. That $f(u + \bar{\eta}\delta) = h(\bar{\eta})$ gives the result. □

Comparing Lemma 4 to the second part of the proof of Lemma 1, we see that second-order smoothness with optimal η guarantees $\alpha^3/(12L_2^2)$ function decrease, while third-order smoothness guarantees a $3\alpha^2/(8L_3)$ decrease. Recalling Theorem 1, where α scales as a power of ϵ, this is evidently a significant improvement. Additionally, this benefit is essentially free: there is no increase in computational cost and no access to higher order derivatives. Examining the proof, we see that the result is rooted in the anti-symmetry of the odd-order terms in the Taylor expansion. This rules out extending this idea to higher orders of smoothness, as they contain symmetric fourth order terms.

Extending this insight to the setting of Lemma 1 is complicated by the fact that, at relevant scales of $\|u - v\|$, it is no longer possible to guarantee that there is negative curvature at either u or v. Nevertheless, we are able to show that a small modification of EXPLOIT-NC-PAIR achieves the required progress.

Lemma 5. *Let $f : \mathbb{R}^d \to \mathbb{R}$ have L_3-Lipschitz third-order derivatives. Let $\alpha > 0$ and let u and v satisfy* (10) *and let*

Algorithm 5 FIND-BEST-ITERATE$_3$(f, y_0^t, u, v)

```
1: Let $0 \le j < t$ be such that $v = x_j$
2: $c_j \leftarrow (y_j + y_{j-1})/2$   if $j > 0$ else $y_0$
3: $q_j \leftarrow -2y_i + 3y_{j-1}$   if $j > 0$ else $y_0$
4: return $\arg\min_{z \in \{y_0,\dots,y_t,c_j,q_j,u\}} f(z)$
```

$\eta \le \sqrt{2\alpha/L_3}$. *Then for every* $\|u - v\| \le \eta/2$, EXPLOIT-NC-PAIR$_3$*(f, u, v, η) finds a point* z *such that*

$$f(z) \le \max\left\{f(v) - \frac{\alpha}{4}\eta^2, f(u) - \frac{\alpha}{12}\eta^2\right\}. \tag{16}$$

We prove Lemma 5 in Section C.1; it is essentially a more technical version of the proof of Lemma 4, where we address the asymmetry of condition (10) by taking steps of different sizes from u and v.

4.2. Bounding the function values of the iterates using cubic interpolation

An important difference between Lemmas 1 and 5 is that the former guarantees lower objective value than $f(u)$, while the latter only improves $\max\{f(v), f(u)\}$. We invoke these lemmas for $v = x_j$ for some x_j produced by AGD-UNTIL-GUILTY, but Corollary 1 only bounds the function value at y_j and w; $f(x_j)$ might be much larger than $f(y_0)$, rendering the progress guaranteed by Lemma 5 useless. Fortunately, we are able show that whenever this happens, there must be a point on the line that connects x_j, y_j and y_{j-1} for which the function value is much lower than $f(y_0)$. We take advantage of this fact in Alg. 5, where we modify FIND-BEST-ITERATE to consider additional points, so that whenever the iterate it finds is not much better than y_0, then $f(x_j)$ is guaranteed to be close to $f(y_0)$. We formalize this claim in the following lemma, which we prove in Section C.2.

Lemma 6. *Let f be L_1-smooth and have L_3-Lipschitz continuous third-order derivatives, and let* $\tau \le \sqrt{\alpha/(16L_3)}$ *with* $\tau, \alpha, L_1, L_3 > 0$. *Consider* GUARDED-NON-CONVEX-AGD *with* FIND-BEST-ITERATE *replaced by* FIND-BEST-ITERATE$_3$. *At any iteration, if* $u, v \ne$ NULL *and the best iterate* $b^{(1)}$ *satisfies* $f(b^{(1)}) \ge f(y_0) - \alpha\tau^2$ *then,*

$$f(v) \le f(y_0) + 14\alpha\tau^2.$$

We now explain the idea behind the proof of Lemma 6. Let $0 \le j < t$ be such that $v = x_j$ (such j always exists by Corollary 1). If $j = 0$ then $x_j = y_0$ and the result is trivial, so we assume $j \ge 1$. Let $f_r : \mathbb{R} \to \mathbb{R}$ be the restriction of f to the line containing y_{j-1} and y_j (and also q_j, c_j and x_j). Suppose now that f_r is a cubic polynomial. Then, it is completely determined by its values at any 4 points, and $f(x_j) = -C_1 f(q_j) + C_2 f(y_{j-1}) - C_3 f(c_j) + C_4 f(y_j)$ for $C_j \ge 0$ independent of f. By substituting the bounds $f(y_{j-1}) \vee f(y_j) \le f(y_0)$ and $f(q_j) \wedge f(c_j) \ge f(b^{(1)}) \ge f(y_0) - \alpha\tau^2$, we obtain an upper bound on $f(x_j)$ when f_r is cubic. To generalize this upper bound to f_r with Lipschitz third-order derivative, we can simply add to it the approximation error of an appropriate third-order Taylor series expansion, which is bounded by a term proportional to $L_3\tau^4 \le \alpha\tau^2/16$.

4.3. An improved rate of convergence

With our algorithmic and analytic upgrades established, we are ready to state the enhanced performance guarantees for GUARDED-NON-CONVEX-AGD, where from here on we assume that EXPLOIT-NC-PAIR$_3$ and FIND-BEST-ITERATE$_3$ subsume EXPLOIT-NC-PAIR and FIND-BEST-ITERATE, respectively.

Lemma 7. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be L_1-smooth and have L_3-Lipschitz continuous third-order derivatives, let $\epsilon, \alpha > 0$ and $p_0 \in \mathbb{R}^d$. If p_0^K is the sequence of iterates produced by* GUARDED-NON-CONVEX-AGD*(f, p_0, L_1, ϵ, α, $\sqrt{\frac{2\alpha}{L_3}}$), then for every $1 \le k < K$,*

$$f(p_k) \le f(p_{k-1}) - \min\left\{\frac{\epsilon^2}{5\alpha}, \frac{\alpha^2}{32L_3}\right\}. \tag{17}$$

The proof of Lemma 7 is essentially identical to the proof of Lemma 2, where we replace Lemma 1 with Lemmas 5 and 6 and set $\tau = \sqrt{\alpha/(32L_3)}$. For completeness, we give a full proof in Section C.3. The gradient evaluation complexity guarantee for third-order smoothness then follows precisely as in our proof of Theorem 1; see Sec. C.4 for a proof of the following

Theorem 2. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be L_1-smooth and have L_3-Lipschitz continuous third-order derivatives. Let $p_0 \in \mathbb{R}^d$, $\Delta_f = f(p_0) - \inf_{z\in\mathbb{R}^d} f(z)$ and $0 < \epsilon^{2/3} \le \min\{\Delta_f^{1/2} L_3^{1/6}, L_1/(8L_3^{1/3})\}$. If we set*

$$\alpha = 2L_3^{1/3}\epsilon^{2/3}, \tag{18}$$

GUARDED-NON-CONVEX-AGD*(f, p_0, L_1, ϵ, α, $\sqrt{\frac{2\alpha}{L_3}}$) finds a point p_K such that $\|\nabla f(p_K)\| \le \epsilon$ and requires at most*

$$20 \cdot \frac{\Delta_f L_1^{1/2} L_3^{1/6}}{\epsilon^{5/3}} \log\left(\frac{500L_1\Delta_f}{\epsilon^2}\right) \tag{19}$$

gradient evaluations.

We remark that Lemma 2 and Theorem 1 remain valid after the modifications described in this section. Thus, Alg. 3 transitions between smoothness regimes by simply varying the scaling of α and η with ϵ.

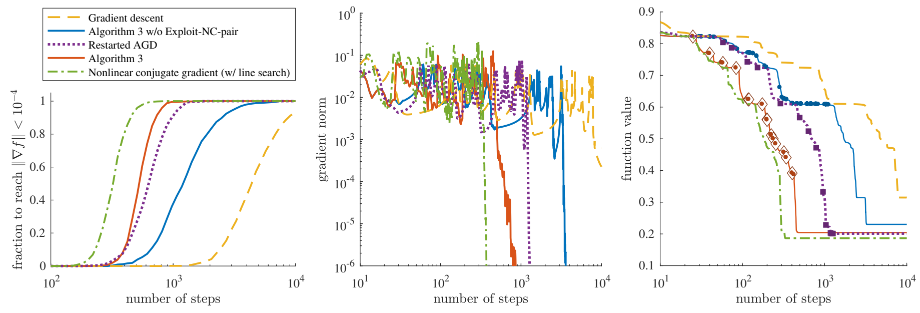

Figure 1. Performance on a non-convex regression problem. *Left:* cumulative distribution of number of steps required to achieve gradient norm $< 10^{-4}$. *Center:* gradient norm trace for a representative instance. *Right:* function value trace for the same instance. For Alg. 3, the dots correspond to negative curvature detection and the diamonds correspond to negative curvature exploitation (*i.e.* when $f(b^{(2)}) < f(b^{(1)})$). For RAGD, the squares indicate restarts due to non-monotonicity.

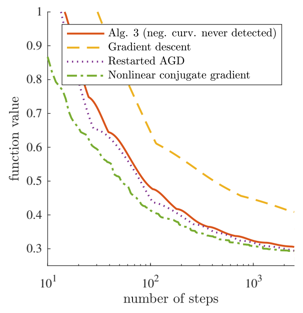

Figure 2. Performance on neural network training.

5. Preliminary experiments

The primary purpose of this paper is to demonstrate the feasibility of acceleration for non-convex problems using only first-order information. Given the long history of development of careful schemes for non-linear optimization, it is unrealistic to expect a simple implementation of the momentum-based Algorithm 3 to outperform state-of-the-art methods such as non-linear conjugate gradients and L-BFGS. It is important, however, to understand the degree of non-convexity in problems we encounter in practice, and to investigate the efficacy of the negative curvature detection-and-exploitation scheme we propose.

Toward this end, we present two experiments: (1) fitting a non-linear regression model and (2) training a small neural network. In these experiments we compare a basic implementation of Alg. 3 with a number baseline optimization methods: gradient descent (GD), non-linear conjugate gradients (NCG) (Hager and Zhang, 2006), Accelerated Gradient Descent (AGD) with adaptive restart (O'Donoghue and Candès, 2015) (RAGD), and a crippled version of Alg. 3 without negative curvature exploitation (C-Alg. 3). We compare the algorithms on the number of gradient steps, but note that the number of oracle queries per step varies between methods. We provide implementation details in Section E.1.

For our first experiment, we study robust linear regression with the smooth biweight loss (Beaton and Tukey, 1974), $f(x) = \frac{1}{m}\sum_{i=1}^{m} \phi(a_i^T x - b_i)$ where $\phi(\theta) := \theta^2/(1+\theta^2)$. For 1,000 independent experiments, we randomly generate problem data to create a highly non-convex problem (see Section E.2). In Figure 1 we plot aggregate convergence time statistics, as well as gradient norm and function value trajectories for a single representative problem instance. The figure shows that gradient descent and C-Alg. 3 (which does not exploit curvature) converge more slowly than the other methods. When C-Alg. 3 stalls it is detecting negative curvature, which implies the stalling occurs around saddle points. When negative curvature exploitation is enabled, Alg. 3 is faster than RAGD, but slower than NCG. In this highly non-convex problem, different methods often converge to local minima with (sometimes significantly) different function values. However, each method found the "best" local minimum in a similar fraction of the generated instances, so there does not appear to be a significant difference in the ability of the methods to find "good" local minima in this problem ensemble.

For the second experiment we fit a neural network model[1] comprising three fully-connected hidden layers containing 20, 10 and 5 units, respectively, on the MNIST handwritten digits dataset (LeCun et al., 1998) (see Section E.3). Figure 2 shows a substantial performance gap between gradient descent and the other methods, including Alg. 3. However, this is not due to negative curvature exploitation; in fact, Alg. 3 never detects negative curvature in this problem, implying AGD never stalls. Moreover, RAGD never restarts. This suggests that the loss function f is "effectively convex" in large portions of the training trajectory, consistent with the empirical observations of Goodfellow et al. (2015); a phenomenon that may merit further investigation.

We conclude that our approach can augment AGD in the presence of negative curvature, but that more work is necessary to make it competitive with established methods such as non-linear conjugate gradients. For example, adaptive schemes for setting α, η and L_1 must be developed. However, the success of our method may depend on whether AGD stalls at all in real applications of non-convex optimization.

[1]Our approach in its current form is inapplicable to training neural networks of modern scale, as it requires computation of exact gradients.

Acknowledgment

OH was supported by the PACCAR INC fellowship. YC and JCD were partially supported by the SAIL-Toyota Center for AI Research and NSF-CAREER award 1553086. YC was partially supported by the Stanford Graduate Fellowship and the Numerical Technologies Fellowship.

References

N. Agarwal, Z. Allen-Zhu, B. Bullins, E. Hazan, and T. Ma. Finding approximate local minima for non-convex optimization in linear time. *arXiv preprint arXiv:1611.01146*, 2016.

A. E. Beaton and J. W. Tukey. The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data. *Technometrics*, 16(2):147–185, 1974.

A. Beck and M. Teboulle. Gradient-based algorithms with applications to signal recovery. *Convex optimization in signal processing and communications*, pages 42–88, 2009.

S. Bubeck. Convex optimization: Algorithms and complexity. *arXiv preprint arXiv:1405.4980*, 2014.

E. J. Candès, X. Li, and M. Soltanolkotabi. Phase retrieval via Wirtinger flow: Theory and algorithms. *IEEE Transactions on Information Theory*, 61(4):1985–2007, 2015.

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

C. Cartis, N. I. 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.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

I. J. Goodfellow, O. Vinyals, and A. M. Saxe. Qualitatively characterizing neural network optimization problems. In *International Conference on Learning Representations*, 2015.

W. W. Hager and H. Zhang. A survey of nonlinear conjugate gradient methods. *Pacific Journal of Optimization*, 2(1):35–58, 2006.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

Y. LeCun, C. Cortes, and C. J. Burges. The MNIST database of handwritten digits, 1998.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht. Gradient descent only converges to minimizers. In *29th Annual Conference on Learning Theory (COLT)*, pages 1246—1257, 2016.

D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In *Advances in Neural Information Processing Systems 21*, 2008.

K. Murty and S. Kabadi. Some NP-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39:117–129, 1987.

A. Nemirovski. Optimization II: Standard numerical methods for nonlinear continuous optimization. Technion – Israel Institute of Technology, 1999. URL http://www2.isye.gatech.edu/~nemirovs/Lect_OptII.pdf.

A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley, 1983.

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.

Y. Nesterov. Squared functional systems and optimization problems. In *High Performance Optimization*, volume 33 of *Applied Optimization*, pages 405–440. Springer, 2000.

Y. Nesterov. *Introductory Lectures on Convex Optimization*. Kluwer Academic Publishers, 2004.

Y. Nesterov. How to make the gradients small. *Optima 88*, 2012.

Y. Nesterov and B. T. Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2006.

B. O’Donoghue and E. Candès. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, 15(3):715–732, 2015.

E. Polak and G. Ribière. Note sur la convergence de directions conjugées. *Rev. Fr. Inform. Rech. Oper. v16*, pages 35–43, 1969.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing – Explorations in the Microstructure of Cognition*, chapter 8, pages 318–362. MIT Press, 1986.

G. Wang, G. B. Giannakis, and Y. C. Eldar. Solving systems of random quadratic equations via truncated amplitude flow. *arXiv:1605.08285 [stat.ML]*, 2016.