ISE

Industrial and Systems Engineering

Exploiting Negative Curvature in Deterministic and Stochastic Optimization

Frank E. Curtis

Department of Industrial and Systems Engineering
Lehigh University, Bethlehem, PA, USA

Daniel P. Robinson

Department of Applied Mathematics and Statistics
Johns Hopkins University, Baltimore, MD, USA

COR@L Technical Report 17T-003

LEHIGH UNIVERSITY

COR@L
COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Noname manuscript No.
(will be inserted by the editor)

Exploiting Negative Curvature in Deterministic and Stochastic Optimization

Frank E. Curtis · Daniel P. Robinson

October 15, 2017

Abstract This paper addresses the question of whether it can be beneficial for an optimization algorithm to follow directions of negative curvature. Although some prior work has established convergence results for algorithms that integrate both descent and negative curvature steps, there has not yet been extensive numerical evidence showing that such methods offer consistent performance improvements. In this paper, we present new frameworks for combining descent and negative curvature directions: alternating two-step approaches and dynamic step approaches. The aspect that distinguishes our approaches from ones previously proposed is that they make algorithmic decisions based on (estimated) upper-bounding models of the objective function. A consequence of this aspect is that our frameworks can, in theory, employ fixed stepsizes, which makes the methods readily translatable from deterministic to stochastic settings. For deterministic problems, we show that instances of our dynamic framework yield gains in performance compared to related methods that only follow descent steps. We also show that gains can be made in a stochastic setting in cases when a standard stochastic-gradient-type method might make slow progress.

Keywords nonconvex optimization · second-order methods · modified Newton methods · negative curvature · stochastic optimization · machine learning

Mathematics Subject Classification (2010) 49M05 · 49M15 · 49M37 · 65K05 · 90C15 · 90C26 · 90C30 · 90C53

F. E. Curtis
Department of Industrial and Systems Engineering
Lehigh University
E-mail: frank.e.curtis@gmail.com
This author was supported in part by the U.S. Department of Energy under Grant No. DE-SC0010615 and by the U.S. National Science Foundation under Grant No. CCF-1618717.

D. P. Robinson
Department of Applied Mathematics and Statistics
Johns Hopkins University
E-mail: daniel.p.robinson@gmail.com

1 Introduction

There has been a recent surge of interest in solving nonconvex optimization problems. A prime example is the dramatic increase in interest in the training of deep neural networks, a subject that admits very challenging nonconvex optimization problems. Another example is the task of clustering data that arise from the union of low dimensional subspaces. In this setting, the nonconvexity typically results from sophisticated modeling approaches that attempt to accurately capture corruptions in the data [5,13]. It is now widely accepted that the design of new methods for solving nonconvex problems (at least locally) is sorely needed.

First consider deterministic optimization problems. For solving such problems, most algorithms designed for minimizing smooth objective functions only ensure convergence to first-order stationarity, i.e., that the gradient of the objective asymptotically vanishes. This characterization is certainly accurate for line search methods, which seek to reduce the objective function by searching along descent directions. Relatively few researchers have designed line search (or other, such as trust region or regularization) algorithms that generate iterates that provably converge to second-order stationarity. The reason for this is three-fold: (*i*) such methods are more complicated and expensive, necessarily involving the computation of directions of negative curvature when they exist; (*ii*) methods designed only to achieve first-order stationarity rarely get stuck at saddle points that are first-order, but not second-order stationary [18]; and (*iii*) there has not been sufficient evidence showing benefits of integrating directions of negative curvature.

For stochastic problems, the methods most commonly invoked are variants of the stochastic gradient (SG) method. During each iteration of SG, a stochastic gradient is computed and a step opposite that direction is taken to obtain the next iterate. Even for nonconvex problems, convergence guarantees (e.g., in expectation or almost surely) for SG methods to first-order stationarity have been established under reasonable assumptions; e.g., see [2]. In fact, SG and its variants represent the current state-of-the-art for training deep neural networks. As for methods that compute and follow negative curvature directions, it is no surprise that such methods have not been used or studied extensively since practical benefits have not even been shown in deterministic optimization.

The main purpose of this paper is to revisit and provide new perspectives on the use of negative curvature directions in deterministic and stochastic optimization. Whereas previous work in deterministic settings has focused on line search and other methods, we focus on methods that attempt to construct *upper-bounding models* of the objective function. This allows them, at least in theory, to employ *fixed stepsizes* while ensuring convergence guarantees. For a few instances of such methods of interest, we provide theoretical convergence guarantees and empirical evidence showing that an optimization process can benefit by following negative curvature directions. The fact that our methods might employ fixed stepsizes is also important as it allows us to offer new strategies for stochastic optimization where, e.g., line search strategies are not often viable. Overall, the theme of this paper can be summarized as the following: *In contrast to methods designed to avoid ill-effects associated with the presence of negative curvature, our methods* exploit *directions of negative curvature to achieve gains in performance.*

1.1 Contributions

The contributions of our work are the following.

- For deterministic optimization, we first provide conditions on descent and negative curvature directions that allow us to guarantee convergence to second-order stationarity with a simple two-step iteration with fixed stepsizes; see §2.1.
- Our second method for deterministic optimization uses the two-step iteration as motivation for the design of a dynamic choice for the direction and stepsize; see §2.2. In particular, the dynamic algorithm makes decisions based on which available step appears to offer a more significant objective reduction.
- We prove convergence rate guarantees for our deterministic optimization methods that provide upper bounds for the numbers of iterations required to achieve (approximate) first- and second-order stationarity; see §2.3.
- In §2.4 and §2.5, we discuss different techniques for computing the search directions in our deterministic algorithms and provide the results of numerical experiments showing the benefits of following negative curvature directions, as opposed to only descent directions.
- For stochastic optimization, we propose two methods. Our first method shows that one can maintain the convergence guarantees of a stochastic gradient method by adding an appropriately scaled negative curvature direction for a stochastic Hessian estimate; see §3.1. This approach can be seen as a refinement of that in [23], which adds noise to each SG step.
- Our second method for stochastic optimization is an adaptation of our dynamic (deterministic) method when stochastic gradient and Hessian estimates are involved; see §3.2. Although we are unable to establish a convergence theory for this approach, we do illustrate some gain in performance in neural network training; see §3.3. We view this as a first step in the design of a practical algorithm for stochastic optimization that efficiently exploits negative curvature.

Computing directions of negative curvature carries an added cost that should be taken into consideration when comparing algorithm performance. We remark along with the results of our numerical experiments how these added costs can be worthwhile and/or marginal relative to the other per-iteration costs.

1.2 Prior Related Work

For deterministic optimization problems, relatively little research has been directed towards the use of negative curvature directions. Perhaps the first exception is the work in [22] in which convergence to second-order stationary points is proved using a curvilinear search formed by descent and negative curvature directions. In a similar vein, the work in [6] offers similar convergence properties based on using a curvilinear search, although the primary focus in that work is describing how a partial Cholesky factorization of the Hessian matrix could be used to compute descent and negative curvature directions. More recently, a linear combination of descent and negative curvature directions was used in an optimization framework to establish convergence to second-order stationary solutions under loose assumptions [8,9]. For further instances of work employing negative curvature directions, see [1,10,11,21]. Importantly, none of the papers above (or any others to the best

of our knowledge) have established consistent gains in computational performance as a result of using negative curvature directions.

Another recent trend in the design of deterministic methods for nonconvex optimization is to focus on the ability of an algorithm to escape regions around a saddle point, i.e., a first-order stationary point that is not second-order stationary. A prime example of this trend is the work in [25] in which a standard type of regularized Newton method is considered. (For a more general presentation of regularized Newton methods of which the method in [25] is a special case, see [24].) While the authors do show a probabilistic convergence result for their method to (approximate) second-order stationarity, their main emphasis is on the number of iterations required to escape neighborhoods of saddle points. This result, similar to other recent papers discussing the behavior of descent methods when solving nonconvex problems (e.g., see [19]), requires that all saddle points are *non-degenerate* in the sense that the Hessian of the objective at any such point does not have any zero eigenvalues. Our convergence theory does not require such an assumption; see §2.6 for additional discussion comparing our results to others.

For stochastic optimization problems, there has been little work that focuses explicitly on the use of directions of negative curvature. For one, see [3]. Meanwhile, it has recently become clear that the nonconvex optimization problems used to train deep neural networks have a rich and interesting landscape [4,14]. Instead of using negative curvature to their advantage, modern methods ignore it, introduce random perturbations (e.g., see [7]), or employ regularized/modified Newton methods that attempt to avoid its potential ill-effects (e.g., [4,20]). Although such approaches are reasonable, one might intuitively expect better performance if directions of negative curvature are exploited instead of avoided. In this critical respect, the methods that we propose are different from prior algorithms.

2 Deterministic Optimization

Consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} \; f(x) \tag{1}$$

where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable and bounded below by a scalar $f_{\inf} \in \mathbb{R}$. We define the gradient function $g := \nabla f$ and Hessian function $H := \nabla^2 f$. We assume that both of these functions are Lipschitz continuous on the path defined by the iterates computed in an algorithm, the gradient function g with Lipschitz constant $L \in (0, \infty)$ and the Hessian function H with Lipschitz constant $\sigma \in (0, \infty)$. Given an invertible matrix $M \in \mathbb{R}^{n \times n}$, its Euclidean norm condition number is written as $\kappa(M) = \|M\|_2 \|M^{-1}\|_2$. Given a scalar $\lambda \in \mathbb{R}$, we define $(\lambda)_- := \min\{0, \lambda\}$.

In the remainder of this section, we present a two-step method that is guaranteed to converge toward second-order stationarity, as well as a dynamic approach that chooses between two types of steps at each point based on lower bounds on objective function decrease. Both algorithms are presented in a generic manner that offers flexibility in the ways the steps are computed.

2.1 Two-Step Method

Our first method alternates between negative curvature and descent steps using fixed stepsizes for each. (Either can be taken first; arbitrarily, we state our algorithm and analysis assuming that one starts with a negative curvature step.) At a given iterate $x_k \in \mathbb{R}^n$, let λ_k denote the left-most eigenvalue of $H(x_k)$. If $\lambda_k \geq 0$ (i.e., $H(x_k) \succeq 0$), the algorithm sets $d_k \leftarrow 0$; otherwise, d_k is computed so that

$$d_k^T H(x_k) d_k \leq \gamma \lambda_k \|d_k\|_2^2 < 0, \tag{2a}$$

$$g(x_k)^T d_k \leq 0, \tag{2b}$$

$$\text{and} \quad \|d_k\|_2 \leq \theta |\lambda_k|, \tag{2c}$$

for some $\gamma \in (0, 1]$ and $\theta \in (0, \infty)$. A step in this direction is then taken to obtain $\hat{x}_k \leftarrow x_k + \beta d_k$ for some $\beta > 0$. At this point, if $g(\hat{x}_k) = 0$, then the algorithm sets $\hat{s}_k \leftarrow 0$; otherwise, $\hat{s}_k$ is computed satisfying

$$\frac{-g(\hat{x}_k)^T \hat{s}_k}{\|\hat{s}_k\|_2 \|g(\hat{x}_k)\|_2} \geq \delta \quad \text{and} \quad \zeta \leq \frac{\|\hat{s}_k\|_2}{\|g(\hat{x}_k)\|_2} \leq \eta \tag{3}$$

for some $(\delta, \zeta) \in (0, 1] \times (0, 1]$ and $\eta \in [1, \infty)$. The iteration ends by taking a step along this direction to obtain $x_{k+1} \leftarrow \hat{x}_k + \alpha \hat{s}_k \equiv x_k + \alpha \hat{s}_k + \beta d_k$ for some $\alpha > 0$. It is worthwhile to remark that (2) and (3) are satisfiable; e.g., if $\hat{s}_k = -g(x_k)$ and d_k is an eigenvector corresponding to the leftmost eigenvalue λ_k scaled so that $g(x_k)^T d_k \leq 0$ and $\|d_k\|_2 = |\lambda_k|$, then (2) and (3) are satisfied with $\gamma = \theta = \delta = \zeta = \eta = 1$. For further remarks on step computation techniques, see §2.4.

Algorithm 1 Two-Step Method

Require: $\alpha \in (0, (2\delta\zeta)/(L\eta^2))$ and $\beta \in (0, (3\gamma)/(\sigma\theta))$
1: **for** $k \in \mathbb{N}$ **do**
2: **if** $\lambda_k \geq 0$ **then** set $d_k \leftarrow 0$ **else** set d_k satisfying (2)
3: set $\hat{x}_k \leftarrow x_k + \beta d_k$
4: **if** $g(\hat{x}_k) = 0$ **then** set $\hat{s}_k \leftarrow 0$ **else** set $\hat{s}_k$ satisfying (3)
5: **if** $d_k = \hat{s}_k = 0$ **then return** x_k
6: set $x_{k+1} \leftarrow \hat{x}_k + \alpha_k \hat{s}_k \equiv x_k + \alpha \hat{s}_k + \beta d_k$
7: **end for**

We now show that Algorithm 1 converges toward second-order stationarity. Critical for this analysis are bounds on the constants (α, β) that are stated in the algorithm. Also, for the analysis, it is convenient to define

$$\mathcal{D} := \{k \in \mathbb{N} : d_k \neq 0\} \equiv \{k \in \mathbb{N} : \lambda_k < 0\}$$

along with the indicator $\mathcal{I}_{\mathcal{D}}(k)$, which evaluates as 1 if $k \in \mathcal{D}$ and as 0 otherwise.

Theorem 1 *If Algorithm 1 terminates finitely in iteration $k \in \mathbb{N}$, then $g(x_k) = 0$ and $\lambda_k \geq 0$, i.e., x_k is second-order stationary. Otherwise, the computed iterates satisfy*

$$\lim_{k \to \infty} \|g(x_k)\|_2 = 0 \quad \textit{and} \quad \liminf_{k \to \infty} \lambda_k \geq 0. \tag{4}$$

Proof Algorithm 1 terminates finitely only if, for some $k \in \mathbb{N}$, $d_k = \hat{s}_k = 0$. This can only occur if $\lambda_k \geq 0$ and, since then $d_k = 0$ yields $\hat{x}_k = x_k$, if $g(x_k) = 0$. These represent the desired conclusions for this case. Otherwise, if Algorithm 1 does not terminate finitely, consider arbitrary $k \in \mathbb{N}$. If $k \notin \mathcal{D}$, then $d_k = 0$ and $\hat{x}_k = x_k$, meaning that $f(\hat{x}_k) = f(x_k)$. Otherwise, if $k \in \mathcal{D}$, then $d_k \neq 0$ and $\lambda_k < 0$, and, by the step computation conditions in (2), it follows that

$$\begin{aligned} f(\hat{x}_k) &\leq f(x_k) + g(x_k)^T(\beta d_k) + \tfrac{1}{2}(\beta d_k)^T H(x_k)(\beta d_k) + \tfrac{1}{6}\sigma\|\beta d_k\|_2^3 \\ &\leq f(x_k) + \tfrac{1}{2}\beta^2\gamma\lambda_k\|d_k\|_2^2 + \tfrac{1}{6}\sigma\beta^3\theta^3|\lambda_k|^3 \\ &= f(x_k) - \tfrac{1}{2}\beta^2\theta^2\big(\gamma - \tfrac{1}{3}\sigma\beta\theta\big)|\lambda_k|^3 \\ &= f(x_k) - c_1(\beta)|\lambda_k|^3, \end{aligned}$$

where $c_1(\beta) := \frac{1}{2}\beta^2\theta^2\big(\gamma - \frac{1}{3}\sigma\beta\theta\big) \in (0, \infty)$. Similarly, it follows from (3) that

$$\begin{aligned} f(x_{k+1}) &\leq f(\hat{x}_k) + g(\hat{x}_k)^T(\alpha\hat{s}_k) + \tfrac{1}{2}L\|\alpha\hat{s}_k\|_2^2 \\ &\leq f(\hat{x}_k) - \alpha\delta\zeta\|g(\hat{x}_k)\|_2^2 + \tfrac{1}{2}L\alpha^2\eta^2\|g(\hat{x}_k)\|_2^2 \\ &= f(\hat{x}_k) - \alpha(\delta\zeta - \tfrac{1}{2}L\alpha\eta^2)\|g(\hat{x}_k)\|_2^2 \\ &= f(\hat{x}_k) - c_2(\alpha)\|g(\hat{x}_k)\|_2^2, \end{aligned}$$

where $c_2(\alpha) := \alpha(\delta\zeta - \frac{1}{2}L\alpha\eta^2) \in (0, \infty)$. Overall,

$$f(x_{k+1}) \leq f(x_k) - \mathcal{I}_{\mathcal{D}}(k)c_1(\beta)|\lambda_k|^3 - c_2(\alpha)\|g(\hat{x}_k)\|_2^2. \tag{5}$$

Now observe that, for any $\ell \in \mathbb{N}$, it follows that

$$\begin{aligned} f(x_0) - f(x_{\ell+1}) &= \sum_{k=0}^{\ell}(f(x_k) - f(x_{k+1})) \\ &\geq \sum_{k=0}^{\ell}\Big(\mathcal{I}_{\mathcal{D}}(k)c_1(\beta)|\lambda_k|^3 + c_2(\alpha)\|g(\hat{x}_k)\|_2^2\Big). \end{aligned}$$

This inequality and f being bounded below show that

$$\sum_{k=0,k\in\mathcal{D}}^{\infty}|\lambda_k|^3 \equiv \sum_{k=0}^{\infty}|(\lambda_k)_-|^3 < \infty \tag{6a}$$

$$\text{and} \quad \sum_{k=0}^{\infty}\|g(\hat{x}_k)\|_2^2 < \infty. \tag{6b}$$

The latter bound yields

$$\lim_{k\to\infty}\|g(\hat{x}_k)\|_2 = 0, \tag{7}$$

while the former bound and (2c) yield

$$\sum_{k=0}^{\infty}\|\hat{x}_k - x_k\|_2^3 = \beta^3\sum_{k=0}^{\infty}\|d_k\|_2^3 \leq \beta^3\theta^3\sum_{k=0}^{\infty}|(\lambda_k)_-|^3 < \infty,$$

from which it follows that

$$\lim_{k\to\infty} \|\hat{x}_k - x_k\|_2 = 0. \tag{8}$$

It follows from Lipschitz continuity of g, (7), and (8) that

$$\begin{aligned}
0 &\leq \limsup_{k\to\infty} \|g(x_k)\|_2 \\
&= \limsup_{k\to\infty} \|g(x_k) - g(\hat{x}_k) + g(\hat{x}_k)\|_2 \\
&\leq \limsup_{k\to\infty} \|g(x_k) - g(\hat{x}_k)\|_2 + \limsup_{k\to\infty} \|g(\hat{x}_k)\|_2 \\
&\leq L \limsup_{k\to\infty} \|x_k - \hat{x}_k\|_2 + \limsup_{k\to\infty} \|g(\hat{x}_k)\|_2 = 0,
\end{aligned}$$

which implies the first limit in (4). Finally, in order to derive a contradiction, suppose that $\liminf_{k\to\infty} \lambda_k < 0$, meaning that there exists some $\epsilon > 0$ and infinite index set $\mathcal{K} \subseteq \mathcal{D}$ such that $\lambda_k \leq -\epsilon$ for all $k \in \mathcal{K}$. This implies that

$$\sum_{k=0}^{\infty} |(\lambda_k)_-|^3 \geq \sum_{k\in\mathcal{K}} |(\lambda_k)_-|^3 \geq \sum_{k\in\mathcal{K}} \epsilon^3 = \infty,$$

contradicting (6a). This yields the second limit in (4). □

There are two potential weaknesses of this two-step approach. First, it simply alternates back-and-forth between descent and negative curvature directions, which might not always lead to the most productive step from each point. Second, even though our analysis holds for all stepsizes α and β in the intervals provided in Algorithm 1, the algorithm might suffer from poor performance if these values are chosen poorly. We next present a method that addresses these weaknesses.

2.2 Dynamic Method

Suppose that, in any iteration $k \in \mathbb{N}$ when $\lambda_k < 0$, one computes a nonzero direction of negative curvature satisfying (2a)–(2b) for some $\gamma \in (0, 1]$. Suppose also that, if $g(x_k) \neq 0$, then one computes a nonzero direction s_k satisfying the equivalent of the first condition in (3), namely, for some $\delta \in (0, 1]$,

$$-g(x_k)^T s_k \geq \delta \|s_k\|_2 \|g(x_k)\|_2. \tag{9}$$

Defining, for $(L_k, \sigma_k) \in (0, \infty) \times (0, \infty)$, the model reductions

$$\begin{aligned}
m_{s,k}(\alpha) &:= -\alpha g(x_k)^T s_k - \tfrac{1}{2} L_k \alpha^2 \|s_k\|_2^2 \\
\text{and} \quad m_{d,k}(\beta) &:= -\beta g(x_k)^T d_k - \tfrac{1}{2}\beta^2 d_k^T H(x_k) d_k - \tfrac{\sigma_k}{6}\beta^3 \|d_k\|_2^3,
\end{aligned}$$

ones finds that, if $L_k \geq L$ and $\sigma_k \geq \sigma$, then

$$f(x_k + \alpha s_k) \leq f(x_k) - m_{s,k}(\alpha) \tag{10a}$$

$$\text{and} \quad f(x_k + \beta d_k) \leq f(x_k) - m_{d,k}(\beta). \tag{10b}$$

These two inequalities suggest that, during iteration $k \in \mathbb{N}$, one could choose which of the two steps (s_k or d_k) to take based on which model reduction predicts the

larger decrease in the objective. One can verify that the reductions $m_{s,k}$ and $m_{d,k}$ are maximized (over the positive real numbers) by

$$\alpha_k := \frac{-g(x_k)^T s_k}{L_k \|s_k\|_2^2} \quad \text{and} \quad \beta_k := \frac{\left(-c_k + \sqrt{c_k^2 - 2\sigma_k \|d_k\|_2^3 g(x_k)^T d_k}\right)}{\sigma_k \|d_k\|_2^3}, \tag{11}$$

where $c_k := d_k^T H(x_k) d_k$ is a measure of $H(x_k)$-curvature for d_k.

Algorithm 2, stated below, follows this dynamic strategy of choosing between s_k and d_k for all $k \in \mathbb{N}$. It also involves dynamic updates for Lipschitz constant estimates, represented in iteration $k \in \mathbb{N}$ by L_k and σ_k. In this deterministic setting, a step is only taken if it yields an objective function decrease. Otherwise, a null step is effectively taken and a Lipschitz constant estimate is increased.

Algorithm 2 Dynamic Method

Require: $(\rho, L_1, \sigma_1) \in (1, \infty) \times (0, \infty) \times (0, \infty)$
for $k \in \mathbb{N}$ **do**
  if $\lambda_k \geq 0$ **then** set $d_k \leftarrow 0$ **else** set d_k satisfying (2a)–(2b)
  if $g(x_k) = 0$ **then** set $s_k \leftarrow 0$ **else** set s_k satisfying (9)
  if $d_k = s_k = 0$ **then return** x_k
  loop
    compute $\alpha_k > 0$ and $\beta_k > 0$ from (11)
    if $m_{s,k}(\alpha_k) \geq m_{d,k}(\beta_k)$ **then**
      if (10a) holds **then**
        set $x_{k+1} \leftarrow x_k + \alpha_k s_k$
        exit loop
      else
        set $L_k \leftarrow \rho L_k$
      end if
    else
      if (10b) holds **then**
        set $x_{k+1} \leftarrow x_k + \beta_k d_k$
        exit loop
      else
        set $\sigma_k \leftarrow \rho \sigma_k$
      end if
    end if
  end loop
  set $(L_{k+1}, \sigma_{k+1}) \in (0, L_k] \times (0, \sigma_k]$
end for

In the next two results, we establish that Algorithm 2 is well-defined and that it has convergence guarantees on par with Algorithm 1 (recall Theorem 1).

Lemma 1 *Algorithm 2 is well defined in the sense that it either terminates finitely or generates infinitely many iterates. In addition, at the end of each iteration $k \in \mathbb{N}$,*

$$L_k \leq L_{\max} := \max\{L_1, \rho L\} \quad \textit{and} \quad \sigma_k \leq \sigma_{\max} := \max\{\sigma_1, \rho\sigma\}. \tag{12}$$

Proof During iteration $k \in \mathbb{N}$, Algorithm 2 might finitely terminate. If it does not, then it enters the **loop**. If that loop were never to terminate, then the updates to L_k and/or σ_k would cause at least one of them to become arbitrarily large. Since (10a) holds whenever $L_k \geq L$, and (10b) holds whenever $\sigma_k \geq \sigma$, it follows that the loop must eventually terminate, thus proving the result. □

Theorem 2 *If Algorithm 2 terminates finitely in iteration $k \in \mathbb{N}$, then $g(x_k) = 0$ and $\lambda_k \geq 0$, i.e., x_k is second-order stationary. Otherwise, the computed iterates satisfy*

$$\lim_{k\to\infty} \|g(x_k)\|_2 = 0 \quad \textit{and} \quad \liminf_{k\to\infty} \lambda_k \geq 0. \tag{13}$$

Proof Algorithm 2 terminates finitely only if, for some $k \in \mathbb{N}$, $d_k = s_k = 0$. This can only occur if $\lambda_k \geq 0$ and $g(x_k) = 0$, which are the desired conclusions. Otherwise, Algorithm 2 does not terminate finitely and one can proceed as follows.

Consider arbitrary $k \in \mathbb{N}$. If $s_k \neq 0$, then the definition of α_k in (11) and the step computation condition on s_k in (9) ensure that

$$m_{s,k}(\alpha_k) = \frac{1}{2L_k}\left(\frac{g(x_k)^T s_k}{\|s_k\|_2}\right)^2 \geq \frac{\delta^2}{2L_k}\|g(x_k)\|_2^2. \tag{14}$$

Similarly, if $d_k \neq 0$, then by the fact that β_k maximizes $m_{d,k}(\beta)$ over $\beta > 0$, (2a)–(2b), and defining $\hat{\beta}_k := -2d_k^T H(x_k)d_k/(\sigma_k\|d_k\|_2^3) > 0$, one finds

$$\begin{aligned} m_{d,k}(\beta_k) &\geq m_{d,k}(\hat{\beta}_k) \\ &\geq -\tfrac{1}{2}\hat{\beta}_k^2 d_k^T H(x_k)d_k - \tfrac{1}{6}\sigma_k\hat{\beta}_k^3\|d_k\|_2^3 \\ &= -\frac{2(d_k^T H(x_k)d_k)^3}{3\sigma_k^2\|d_k\|_2^6} \geq \frac{2\gamma^3}{3\sigma_k^2}|\lambda_k|^3. \end{aligned}$$

One can extend this inequality to all cases (i.e., not only when $d_k \neq 0$), via

$$m_{d,k}(\beta_k) \geq \frac{2\gamma^3}{3\sigma_k^2}|(\lambda_k)_-|^3 \quad \text{for all} \quad k \in \mathbb{N}. \tag{15}$$

Overall, it follows that, for all $k \in \mathbb{N}$,

$$f(x_k) - f(x_{k+1}) \geq \max\left\{\frac{\delta^2}{2L_k}\|g(x_k)\|_2^2, \frac{2\gamma^3}{3\sigma_k^2}|(\lambda_k)_-|^3\right\}. \tag{16}$$

Indeed, to show that (16) holds, let us consider two cases. First, suppose that the update $x_{k+1} \leftarrow x_k + \alpha_k s_k$ is completed, meaning that (10a) and $m_{s,k}(\alpha_k) \geq m_{d,k}(\beta_k)$ both hold. Combining these facts with (14) and (15) establishes (16) in this case. Second, suppose that $x_{k+1} \leftarrow x_k + \beta_k d_k$ is completed, meaning that (10b) and $m_{s,k}(\alpha_k) < m_{d,k}(\beta_k)$ both hold. Combining these facts with (14) and (15) establishes (16) in this case. Thus, (16) holds for $k \in \mathbb{N}$.

It now follows from (16), the bounds in (12), and a proof similar to that used in Theorem 1 (in particular, to establish (6a) and (6b)) that

$$\sum_{k=1}^{\infty}\|g(x_k)\|_2^2 < \infty \quad \text{and} \quad \sum_{k=1}^{\infty}|(\lambda_k)_-| < \infty.$$

One may now establish the desired results in (13) using the same arguments as used in the proof of Theorem 1. □

Let us add a few remarks about Algorithm 2.

– Our convergence theory allows $L_{k+1} \leftarrow L_k$ and $\sigma_{k+1} \leftarrow \sigma_k$ in Step 23 for all $k \in \mathbb{N}$, in which case the Lipschitz constant estimates are monotonically increasing. However, in Algorithm 2, we allow these estimates to decrease since this might yield better results in practice.
– If the Lipschitz constants L and σ for $g := \nabla f$ and $H := \nabla^2 f$, respectively, are known, then one could simply set $L_k = L$ and $\sigma_k = \sigma$ during each iteration; in this case, the loop would not actually be needed. Although this would simplify the presentation, it would generally result in more iterations being required to obtain (approximate) first- and/or second-order stationarity. However, if the cost of evaluating f is substantial, then static parameters might work well.
– Each time through the loop, condition (10a) or (10b) is tested, but not both, since this would require an extra evaluation of f. If the cost of evaluating the objective function is not a concern, then one could choose between the two steps based on *actual* objective function decrease rather than model decrease.

2.3 Complexity Analysis

We have the following complexity result for Algorithm 2. We claim that a similar result could be stated and proved for Algorithm 1 as well, where one employs the inequality (5) in place of (16) in the proof below. However, for brevity and since we believe it might often be the better method in practice, we focus on the following result for our dynamic method, Algorithm 2.

Theorem 3 *Consider any scalars $\epsilon_g \in (0, \bar{\epsilon}_g]$ and $\epsilon_H \in (0, \bar{\epsilon}_H]$ for some constants $(\bar{\epsilon}_g, \bar{\epsilon}_H) \in (0, \infty) \times (0, \infty)$. With respect to Algorithm 2, the cardinality of the index set*

$$\mathcal{G}(\epsilon_g) := \{k \in \mathbb{N} : \|g(x_k)\|_2 > \epsilon_g\}$$

is at most $\mathcal{O}(\epsilon_g^{-2})$. In addition, the cardinality of the index set

$$\mathcal{H}(\epsilon_H) := \{k \in \mathbb{N} : |(\lambda_k)_-| > \epsilon_H\}$$

is at most $\mathcal{O}(\epsilon_H^{-3})$. Hence, the number of iterations and derivative (i.e., gradient and Hessian) evaluations required until iteration $k \in \mathbb{N}$ is reached with

$$\|g(x_k)\|_2 \leq \epsilon_g \quad \text{and} \quad |(\lambda_k)_-| \leq \epsilon_H$$

is at most $\mathcal{O}(\max\{\epsilon_g^{-2}, \epsilon_H^{-3}\})$. Moreover, if Step 23 is modified such that

$$(L_{k+1}, \sigma_{k+1}) \in [L_{\min}, L_k] \times [\sigma_{\min}, \sigma_k], \tag{17}$$

*for some $(L_{\min}, \sigma_{\min}) \in (0, \infty) \times (0, \infty)$ with $L_1 \geq L_{\min}$ and $\sigma_1 \geq \sigma_{\min}$, then the number of iterations in the **loop** for all $k \in \mathbb{N}$ is uniformly bounded, meaning that the complexity bound above also holds for the number of function evaluations.*

Proof As in the proof of Theorem 2, one has that, for all $k \in \mathbb{N}$, the inequality (16) holds, which we restate here as

$$f(x_k) - f(x_{k+1}) \geq \max\left\{\frac{\delta^2}{2L_k}\|g(x_k)\|_2^2, \frac{2\gamma^3}{3\sigma_k^2}|(\lambda_k)_-|^3\right\}. \tag{18}$$

From this inequality and the bounds in (12), it follows that

$$k \in \mathcal{G}(\epsilon_g) \implies f(x_k) - f(x_{k+1}) \geq \frac{\delta^2}{2L_{\max}}\epsilon_g^2$$
$$\text{while } \; k \in \mathcal{H}(\epsilon_H) \implies f(x_k) - f(x_{k+1}) \geq \frac{2\gamma^3}{3\sigma_{\max}^2}\epsilon_H^3.$$

Since f is bounded below by $f_{\inf}$ and (18) ensures that $\{f_k\}$ is monotonically decreasing, the inequalities above imply that $\mathcal{G}(\epsilon_g)$ and $\mathcal{H}(\epsilon_H)$ are both finite. In addition, one has by summing the reductions over the index sets that

$$f_0 - f_{\inf} \geq \sum_{k \in \mathcal{G}(\epsilon_g)} f(x_k) - f(x_{k+1}) \geq |\mathcal{G}(\epsilon_g)| \frac{\delta^2}{2L_{\max}}\epsilon_g^2$$
$$\text{and} \quad f_0 - f_{\inf} \geq \sum_{k \in \mathcal{H}(\epsilon_H)} f(x_k) - f(x_{k+1}) \geq |\mathcal{H}(\epsilon_H)| \frac{2\gamma^3}{3\sigma_{\max}^2}\epsilon_H^3.$$

Rearranging, one obtains that

$$|\mathcal{G}(\epsilon_g)| \leq \left(\frac{2L_{\max}(f_0 - f_{\inf})}{\delta^2}\right)\epsilon_g^{-2} \quad \text{and} \quad |\mathcal{H}(\epsilon_H)| \leq \left(\frac{3\sigma_{\max}^2(f_0 - f_{\inf})}{2\gamma^3}\right)\epsilon_H^{-3},$$

as desired. Finally, the last conclusion follows by the fact that, with (17), the number of iterations within the **loop** for any $k \in \mathbb{N}$ is bounded above by

$$\left(1 + \left\lfloor \frac{1}{\log(\rho)} \log\left(\frac{L}{L_{\min}}\right) \right\rfloor\right) + \left(1 + \left\lfloor \frac{1}{\log(\rho)} \log\left(\frac{\sigma}{\sigma_{\min}}\right) \right\rfloor\right),$$

which is the maximum number of updates to L_k and/or σ_k to bring values within $[L_{\min}, L]$ and $[\sigma_{\min}, \sigma]$ up to satisfy $L_k \geq L$ and $\sigma_k \geq \sigma$. □

2.4 Step Computation Techniques

There is flexibility in the ways in which the steps d_k and s_k (or $\hat{s}_k$) are computed in order to satisfy the desired conditions (in (2) and (3)/(9)). For example, s_k might be the steepest descent direction $-g(x_k)$, for which (3) holds with $\delta = \zeta = \eta = 1$. Another option, with a symmetric positive definite $B_k^{-1} \in \mathbb{R}^{n \times n}$ that has $\kappa(B_k^{-1}) \leq \delta^{-1}$ with a spectrum falling in $[\zeta, \eta]$, is to compute s_k as the modified Newton direction $-B_k^{-1}g(x_k)$. A particularly attractive option for certain applications (when Hessian-vector products are readily computed) is to compute s_k via a Newton-CG routine [24] with safeguards that terminate the iteration before a CG iterate is computed that violates (3) for prescribed (δ, ζ, η).

There are multiple ways to compute the negative curvature direction. In theory, the most straightforward approach is to set $d_k = \pm v_k$, where v_k is a leftmost eigenvector of $H(x_k)$. With this choice, it follows that $d_k^T H(x_k) d_k = \lambda_k \|d_k\|^2$, meaning that (2a) is satisfied, and one can scale d_k to ensure that (2b) and (2c) hold. A second approach for large-scale settings (i.e., when n is large) is to compute d_k via matrix-free Lanczos iterations [16]. Such an approach can produce a direction d_k satisfying (2a), which can then be scaled to yield (2b) and (2c).

2.5 Numerical Results

In this section, we demonstrate that there can be practical benefits of following directions of negative curvature if one follows the dynamic approach of Algorithm 2. To do this, we implemented software in MATLAB that, for all $k \in \mathbb{N}$, has the option to compute s_k via several options (see §2.5.1 and §2.5.2) and $d_k = \pm v_k$ (recall §2.4). Our test problems include a subset of the CUTEst collection [12]. Specifically, we selected all of the unconstrained problems with number of variables $n \leq 500$ and second derivatives explicitly available. This left us with a test set of 97 problems.

Using this test set, we considered two variants of Algorithm 2: (*i*) a version in which the **if** condition in Step 7 is always presumed to test true so that the descent step s_k is chosen for all $k \in \mathbb{N}$ (which we refer to as Algorithm 2(s_k)), and (*ii*) a version that, as in our formal statement of the algorithm, chooses between descent and negative curvature steps by comparing model reduction values for each $k \in \mathbb{N}$ (which we refer to as Algorithm 2(s_k, d_k)). In our experiments, we used $L_1 \leftarrow 1$ and $\sigma_1 \leftarrow 1$, updating them and setting subsequent values in their respective sequences using the following strategy. For increasing one of these values in Step 12 or Step 19, we respectively set the quantities

$$\hat{L}_k \leftarrow L_k + \frac{2\big(f(x_k + \alpha_k s_k) - f(x_k) + m_{s,k}(\alpha_k)\big)}{\alpha_k^2 \|s_k\|^2} \quad \text{or}$$
$$\hat{\sigma}_k \leftarrow \sigma_k + \frac{6\big(f(x_k + \beta_k d_k) - f(x_k) + m_{d,k}(\beta_k)\big)}{\beta_k^3 \|d_k\|^3},$$

then, with $\rho \leftarrow 2$, use the update

$$\begin{aligned} L_k &\leftarrow \max\{\rho L_k, \min\{10^3 L_k, \hat{L}_k\}\} \quad \text{in Step 12 of Algorithm 2 or} \\ \sigma_k &\leftarrow \max\{\rho \sigma_k, \min\{10^3 \sigma_k, \hat{\sigma}_k\}\} \quad \text{in Step 19 of Algorithm 2.} \end{aligned} \tag{19}$$

The quantity $\hat{L}_k$ is defined so that (10a) holds at equality with $\alpha = \alpha_k$ and L_k replaced by $\hat{L}_k$ in the definition of $m_{s,k}(\alpha_k)$, i.e., $\hat{L}_k$ is the value that makes the model decrease agree with the exact function decrease at $x_k + \alpha_k s_k$. The meaning of $\hat{\sigma}_k$ is analogous. We remark that the updates in (19) ensure that $L_k \in [\rho L_k, 10^3 L_k]$ and $\sigma_k \in [\rho \sigma_k, 10^3 \sigma_k]$, which we claim maintains the convergence and complexity guarantees of Algorithm 2. Moreover, when Step 12 (respectively, Step 19) is reached, then it must be the case that $\hat{L}_k > L_k$ (respectively, $\hat{\sigma}_k > \sigma_k$). On the other hand, in Step 23 we use the following updates:

$$\begin{aligned} &L_{k+1} \leftarrow \max\{10^{-3}, 10^{-3} L_k, \hat{L}_k\} \quad \text{and} \quad \sigma_{k+1} \leftarrow \sigma_k \quad \text{when} \quad x_{k+1} \leftarrow x_k + \alpha_k s_k; \\ &\sigma_{k+1} \leftarrow \max\{10^{-3}, 10^{-3} \sigma_k, \hat{\sigma}_k\} \quad \text{and} \quad L_{k+1} \leftarrow L_k \quad \text{when} \quad x_{k+1} \leftarrow x_k + \beta_k d_k. \end{aligned}$$

Over the next two sections, we discuss numerical results when two different options for computing the descent direction s_k are used.

2.5.1 Choosing s_k as the direction of steepest descent

The tests in this section use the steepest descent direction, i.e., $s_k = -g(x_k)$ for all $k \in \mathbb{N}$. Although this is a simple choice, it gives a starting point for understanding the potential benefits of using directions of negative curvature.

We ran Algorithm 2(s_k) and Algorithm 2(s_k, d_k) on the previously described CUTEST problems with commonly used stopping conditions. Specifically, an algorithm terminates with an approximate second-order stationary solution if

$$\|g(x_k)\| \leq 10^{-5} \max\{1, \|g(x_0)\|\} \quad \text{and} \quad |(\lambda_k)_-| \leq 10^{-5} \max\{1, |(\lambda_0)_-|\}. \tag{20}$$

We also terminate an algorithm if an iteration limit of 10,000 is reached or if a trial step smaller than 10^{-16} is computed. The results can be found in Figure 1.

In Figure 1a, letting $f_{\text{final}}(s_k)$ and $f_{\text{final}}(s_k, d_k)$ be the final computed objective values for Algorithm 2(s_k) and Algorithm 2(s_k, d_k), respectively, we plot

$$\frac{f_{\text{final}}(s_k) - f_{\text{final}}(s_k, d_k)}{\max\{|f_{\text{final}}(s_k)|, |f_{\text{final}}(s_k, d_k)|, 1\}} \in [-1, 1] \tag{21}$$

for each problem that Algorithm 2(s_k, d_k) used at least one negative curvature direction and for which (21) is larger than 10^{-5} in absolute value. In this manner, an upward pointing bar implies that Algorithm 2(s_k, d_k) terminated with a lower value of the objective function, while its magnitude represents how much better was this value. We can observe from Figure 1a that among the 31 problems, Algorithm 2(s_k, d_k) terminated with a lower objective value 25 times.

We are also interested in the number of iterations and objective function evaluations needed to obtain these final objective function values. The relative performances of the algorithms with respect to these measures are shown in Figure 1b and Figure 1c. In particular, letting $\#its(s_k)$ be the total number of iterations required by Algorithm 2(s_k) and $\#its(s_k, d_k)$ be the total number of iterations required by Algorithm 2(s_k, d_k), we plot in Figure 1b the values

$$\frac{\#its(s_k) - \#its(s_k, d_k)}{\max\{\#its(s_k), \#its(s_k, d_k), 1\}} \in [-1, 1], \tag{22}$$

and, letting $\#fevals(s_k)$ be the total number of objective function evaluations for Algorithm 2(s_k) and $\#fevals(s_k, d_k)$ be the total number of objective function evaluations for Algorithm 2(s_k, d_k), we plot in Figure 1c the values

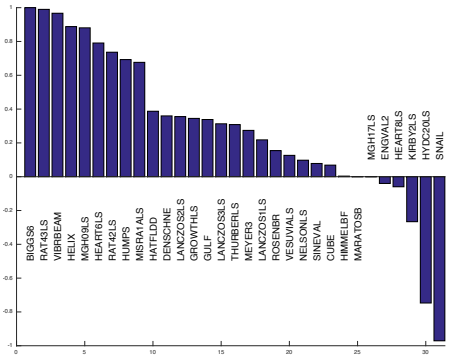
$$\frac{\#fevals(s_k) - \#fevals(s_k, d_k)}{\max\{\#fevals(s_k), \#fevals(s_k, d_k), 1\}} \in [-1, 1]. \tag{23}$$

These two figures show that Algorithm 2(s_k, d_k) tends to perform better than Algorithm 2(s_k) in terms of both the number of required iterations and function evaluations. Overall, we find these results interesting since Algorithm 2(s_k, d_k) tends to find lower values of the objective function (see Figure 1a) while typically requiring fewer iterations (see Figure 1b) and function evaluations (see Figure 1c).

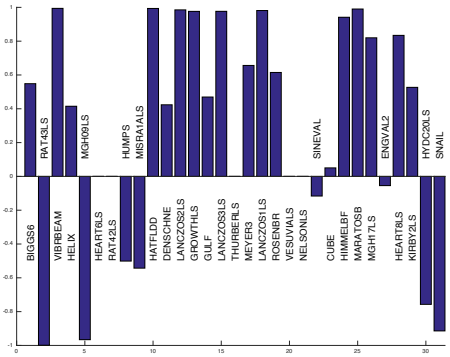
2.5.2 Choosing s_k using a modified-Newton strategy

In this subsection, we show the results of tests similar to those in §2.5.1 except that now we compute the descent direction s_k by a modified-Newton approach. Specifically, we compute s_k as the unique vector satisfying $B_k s_k = -g(x_k)$ with

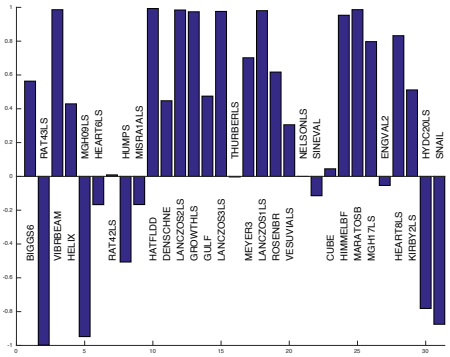
$$B_k = H(x_k) + \delta_k I, \tag{24}$$



(a) Plot associated with the quantity (21).



(b) Plot associated with the quantity (22).



(c) Plot associated with the quantity (23).

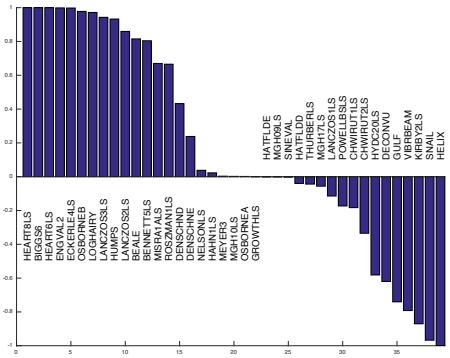
Fig. 1: Plots for the choice $s_k \equiv -g(x_k)$ for all $k \in \mathbb{N}$. Only problems for which at least one negative curvature direction is used and for which the value in (21) is larger than 10^{-5} in absolute value are presented. The problems are ordered based on the values in plot (a).

where I is the identify matrix and δ_k is the smallest nonnegative real number such that B_k is positive definite with a condition number less than or equal to 10^8.

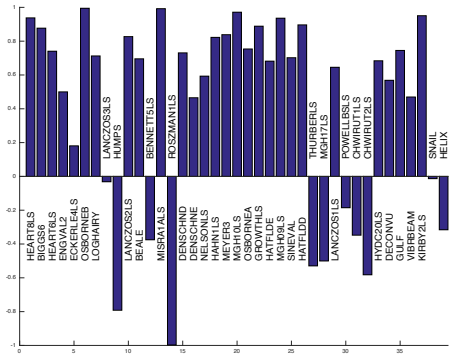
The results from solving our CUTEST test set with this choice of B_k using the stopping condition (20) are presented in Figure 2. We can observe that Algorithm 2(s_k, d_k) outperforms Algorithm 2(s_k) in that it often computes lower objective function values (see Figure 2a), while typically requiring fewer iterations (see Figure 2b) and objective function evaluations (see Figure 2c).

2.6 Behavior Near Strict Saddle Points

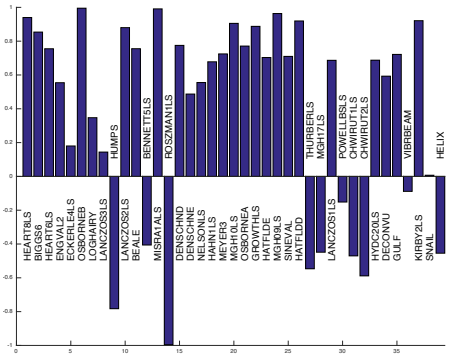
As previously mentioned (recall §1.2), much recent attention has been directed toward the behavior of nonconvex optimization algorithms in the neighborhood of



(a) Plot associated with the quantity (21).



(b) Plot associated with the quantity (22).



(c) Plot associated with the quantity (23).

Fig. 2: Plots for when s_k is chosen to satisfy $B_k s_k = -g(x_k)$ with B_k defined via (24). Only problems for which at least one negative curvature direction is used and for which the value in (21) is larger than 10^{-5} in absolute value are presented. The problems are ordered based on the values in plot (a).

saddle points. In particular, in order to prove guarantees about avoiding saddle points, an assumption is often made about all saddle points being *strict* (or *ridable*) or even *non-degenerate*. Strict saddle points are those at which the Hessian has at least one negative eigenvalue; intuitively, these are saddle points from which an algorithm should usually be expected to avoid. Nondegenerate saddle points are ones at which the Hessian has no zero eigenvalues, a much stronger assumption.

One can show that in certain problems of interest, all saddle points are strict. This is interesting. However, much of the work that has discussed the behavior of nonconvex algorithms in the neighborhood of strict saddle points have focused on standard types of descent methods, about which one can only prove high probability results. By contrast, when one considers explicitly computing directions of negative curvature, this is all less of an issue. After all, notice that our convergence and complexity analyses for our framework did not require careful consideration of the nature of any potential saddle points.

In any case, for ease of comparison to other recent analyses, let us discuss the convergence/complexity properties for our method in the context of a strict saddle point assumption. Suppose that the set of maximizers and saddle points of f, say $\{\bar{x}_i\}_{i\in\mathcal{I}}$ for some index set $\mathcal{I}$, which must necessarily have $g(\bar{x}_i) = 0$ for all $i \in \mathcal{I}$, also has $\bar{\lambda}_i < 0$ for all $i \in \mathcal{I}$, where $\{\bar{\lambda}_i\}_{i\in\mathcal{I}}$ are the leftmost eigenvalues of $\{H(\bar{x}_i)\}_{i\in\mathcal{I}}$ and are uniformly bounded away from zero. In this setting, we may draw the following conclusions from Theorems 2 and 3.

- Any limit point of the sequence $\{x_k\}$ computed by Algorithm 2 is a minimizer of f. This follows since Theorem 2 ensures that for any limit point, say $\bar{x}$, the gradient must be zero and the leftmost eigenvalue must be nonnegative. It follows from these facts and the assumptions on the maximizers and saddle points $\{\bar{x}_i\}$ that $\bar{x}$ must be a minimizer, as claimed.
- From the discussion in the previous bullet and Theorem 3 we know, in fact, that the iterates of Algorithm 2 must eventually enter a region consisting only of minimizers (i.e., one that does not contain any maximizers or saddle points) in a number of iterations that is polynomial in $\epsilon \in (0, \infty)$, where the negative value $-\epsilon$ is greater than the largest of the leftmost eigenvalues of $\{H(\bar{x}_i)\}_{i\in\mathcal{I}}$. This complexity result holds without assuming that the minimizers, maximizers, and saddle points are all nondegenerate (i.e., the Hessian matrix at these points are nonsingular), which, e.g., should be contrasted with the analysis in [25, see Assumption 3]. The primary reason for our stronger convergence/complexity properties is that our method incorporates negative curvature directions when they exist, as opposed to only descent directions.

3 Stochastic Optimization

Let us now consider the problem to minimize a twice continuously differentiable and bounded below (by $f_{\inf} \in \mathbb{R}$) objective function $f : \mathbb{R}^n \to \mathbb{R}$ defined by the expectation, in terms of the distribution of a random variable ξ with domain Ξ, of a stochastic function $F : \mathbb{R}^n \times \Xi \to \mathbb{R}$, namely,

$$\min_{x\in\mathbb{R}^n} \; f(x), \quad \text{where} \quad f(x) := \mathbb{E}_\xi[F(x, \xi)]. \tag{25}$$

In this context, we expect that, at an iterate x_k, one can only compute stochastic gradient and Hessian estimates. We do not claim that we are able to prove convergence guarantees to second-order stationarity as in the deterministic case. That said, we are able to present a two-step method with convergence guarantees to first-order stationarity whose structure motivates a dynamic method that we show can offer beneficial practical performance by exploring negative curvature.

3.1 Two-Step Method: Stochastic Gradient/Newton with "Curvature Noise"

At an iterate x_k, let ξ_k be a random variable representing a seed for generating a vector $s_k \in \mathbb{R}^n$. For example, if f is the expected function value over inputs from a dataset, then ξ_k might represent sets of points randomly drawn from the dataset.

With $\mathbb{E}_{\xi_k}[\cdot]$ denoting expectation taken with respect to the distribution of ξ_k given the current iterate x_k, we require the vector s_k to satisfy

$$-\nabla f(x_k)^T \mathbb{E}_{\xi_k}[s_k] \geq \delta \|\nabla f(x_k)\|_2^2, \tag{26a}$$

$$\mathbb{E}_{\xi_k}[\|s_k\|_2] \leq \eta \|\nabla f(x_k)\|_2, \quad \text{and} \tag{26b}$$

$$\mathbb{E}_{\xi_k}[\|s_k\|_2^2] \leq M_{s1} + M_{s2} \|\nabla f(x_k)\|_2^2 \tag{26c}$$

for some $\delta \in (0, 1]$, $\eta \in [1, \infty)$, and $(M_{s1}, M_{s2}) \in (0, \infty) \times (1, \infty)$. For example, as in a stochastic gradient method, these conditions are satisfied if s_k is an unbiased estimate of $\nabla f(x_k)$ with second moment bounded as in (26c). They are also satisfied in the context of a stochastic Newton method wherein a stochastic gradient estimate is multiplied by a stochastic inverse Hessian estimate, assuming that the latter is conditionally uncorrelated with the former and has eigenvalues contained within an interval of the positive real line uniformly over all $k \in \mathbb{N}$.

Let us also define ξ_k^H, conditionally uncorrelated with ξ_k given x_k, as a random variable representing a seed for generating an unbiased Hessian estimate H_k such that $\mathbb{E}_{\xi_k^H}[H_k] = \nabla^2 f(x_k)$. We use H_k to compute a direction d_k. For the purpose of ideally following a direction of negative curvature (for the true Hessian), we ask that d_k satisfies a curvature condition similar to that used in the deterministic setting. Importantly, however, the second moment of d_k must be bounded similar to that of s_k above. Overall, with λ_k being the left-most eigenvalue of H_k, we set $d_k \leftarrow 0$ if $\lambda_k \geq 0$, and otherwise require the direction d_k to satisfy

$$d_k^T H_k d_k \leq \gamma \lambda_k \|d_k\|_2^2 < 0 \quad \text{given } H_k \tag{27a}$$

$$\text{and} \quad \mathbb{E}_{\xi_k^H}[\|d_k\|_2^2] \leq M_{d1} + M_{d2} \|\nabla f(x_k)\|_2^2 \tag{27b}$$

for some $\gamma \in (0, 1]$ and $(M_{d1}, M_{d2}) \in (0, \infty) \times (1, \infty)$. One manner in which these conditions can be satisfied is to compute d_k as an eigenvector corresponding to the left-most eigenvalue λ_k, scaled such that $\|d_k\|_2^2$ is bounded above in proportion to the squared norm $\|s_k\|_2^2$ where s_k satisfies (26).

Note that our conditions in (27) do *not* involve expected descent with respect to the true gradient at x_k. This can be viewed in contrast to (2), which involves (2b). The reason for this is that, in a practical setting, such a condition might not be verifiable without computing the exact gradient explicitly, which might be intractable or prohibitively expensive. Instead, without this restriction, a critical component of our algorithm is the generation of an independent random scalar ω_k uniformly distributed in $[-1, 1]$. With this choice, one finds $\mathbb{E}_{(\xi_k^H, \omega_k)}[\omega_k d_k] = 0$ such that, effectively, $\omega_k d_k$ merely adds noise to s_k in a manner that ideally follows a negative curvature direction. This leads to Algorithm 3 below.

Algorithm 3 maintains the convergence guarantees of a standard stochastic gradient (SG) method. To show this, let us first prove the following lemma.

Lemma 2 *For all $k \in \mathbb{N}$, it follows that*

$$\begin{aligned} &\mathbb{E}_{(\xi_k, \xi_k^H, \omega_k)}[f(x_{k+1})] - f(x_k) \\ &\leq -(\delta \alpha_k - \tfrac{1}{2} L M_{s2} \alpha_k^2 - \tfrac{1}{6} L M_{d2} \beta_k^2) \|\nabla f(x_k)\|_2^2 + \tfrac{1}{2} L M_{s1} \alpha_k^2 + \tfrac{1}{6} L M_{d1} \beta_k^2. \end{aligned} \tag{28}$$

Algorithm 3 Two-Step Method for Stochastic Optimization

Require: choose $\{\alpha_k\} \subset \mathbb{R}_{++}$ and $\{\beta_k\} \subset \mathbb{R}_{++}$
1: **for** $k \in \mathbb{N}$ **do**
2: generate uncorrelated random seeds ξ_k and ξ_k^H
3: set s_k satisfying (26)
4: set d_k satisfying (27)
5: set ω_k uniformly in $[-1, 1]$
6: set $x_{k+1} \leftarrow x_k + \alpha_k s_k + \beta_k \omega_k d_k$
7: **end for**

Proof From Lipschitz continuity of ∇f, it follows that

$$\begin{aligned} f(x_{k+1}) - f(x_k) &= f(x_k + \alpha_k s_k + \beta_k \omega_k d_k) - f(x_k) \\ &\leq \nabla f(x_k)^T (\alpha_k s_k + \beta_k \omega_k d_k) + \tfrac{1}{2} L \|\alpha_k s_k + \beta_k \omega_k d_k\|_2^2. \end{aligned}$$

Taking expectations with respect to the distribution of the random quantities $(\xi_k, \xi_k^H, \omega_k)$ given x_k and using (26)–(27), it follows that

$$\begin{aligned} & \mathbb{E}_{(\xi_k, \xi_k^H, \omega_k)}[f(x_{k+1})] - f(x_k) \\ \leq\ & \nabla f(x_k)^T (\alpha_k \mathbb{E}_{\xi_k}[s_k] + \beta_k \mathbb{E}_{(\xi_k^H, \omega_k)}[\omega_k d_k]) \\ & + \tfrac{1}{2} L (\alpha_k^2 \mathbb{E}_{\xi_k}[\|s_k\|_2^2] + \beta_k^2 \mathbb{E}_{(\xi_k^H, \omega_k)} \mathbb{E}[\|\omega_k d_k\|_2^2]) \\ & + L \alpha_k \beta_k \mathbb{E}_{(\xi_k, \xi_k^H, \omega_k)}[s_k^T (\omega_k d_k)] \\ \leq\ & -\alpha_k \delta \|\nabla f(x_k)\|_2^2 \\ & + \tfrac{1}{2} L \alpha_k^2 (M_{s1} + M_{s2} \|\nabla f(x_k)\|_2^2) + \tfrac{1}{6} L \beta_k^2 (M_{d1} + M_{d2} \|\nabla f(x_k)\|_2^2). \end{aligned}$$

Rearranging, we reach the desired conclusion. □

From this lemma, we obtain a critical bound similar to one that can be shown for a generic SG method; e.g., see Lemma 4.4 in [2]. Hence, following analyses such as that in [2], one can show that the *total* expectation for the gap between $f(x_k)$ and a lower bound for f decreases. For instance, we prove the following theorem using similar proof techniques as for Theorems 4.8 and 4.9 in [2].

Theorem 4 *Suppose that Algorithm 3 is run with $\alpha_k = \beta_k = \bar{\alpha}$ for all $k \in \mathbb{N}$ where*

$$0 < \bar{\alpha} \leq \frac{\delta}{L \max\{M_{s2}, M_{d2}\}}. \tag{29}$$

Then, for all $K \in \mathbb{N}$, one has that

$$\begin{aligned} \mathbb{E}\left[\frac{1}{K} \sum_{k=1}^{K} \|\nabla f(x_k)\|_2^2\right] &\leq \frac{\bar{\alpha} L \max\{M_{s1}, M_{d1}\}}{\delta} + \frac{2(f(x_1) - f_{\inf})}{K \delta \bar{\alpha}} \\ &\xrightarrow{K \to \infty} \frac{\bar{\alpha} L \max\{M_{s1}, M_{d1}\}}{\delta}. \end{aligned}$$

On the other hand, if Algorithm 3 is run with $\{\alpha_k\}$ satisfying

$$\sum_{k=1}^{\infty} \alpha_k = \infty \quad \text{and} \quad \sum_{k=1}^{\infty} \alpha_k^2 < \infty \tag{30}$$

and $\{\beta_k\} = \{\chi\alpha_k\}$ *for some* $\chi \in (0, \infty)$*, then it holds that*

$$\lim_{K\to\infty} \mathbb{E}\left[\sum_{k=1}^{K} \alpha_k \|\nabla f(x_k)\|_2^2\right] < \infty,$$

from which it follows, with $A_K := \sum_{k=1}^{K} \alpha_k$*, that*

$$\lim_{K\to\infty} \mathbb{E}\left[\frac{1}{A_K}\sum_{k=1}^{K} \alpha_k \|\nabla f(x_k)\|_2^2\right] = 0,$$

which implies that $\liminf_{k\to\infty} \mathbb{E}[\|\nabla f(x_k)\|_2^2] = 0$.

Proof First, suppose that Algorithm 3 is run with $\alpha_k = \beta_k = \bar{\alpha}$ for all $k \in \mathbb{N}$. Then, taking total expectation in (28), it follows with (29) and since $\frac{1}{2} > \frac{1}{6}$ that

$$\begin{aligned}
&\mathbb{E}[f(x_{k+1})] - f(x_k) \\
&\leq -(\delta - \tfrac{1}{2}L\max\{M_{s2}, M_{d2}\}\bar{\alpha})\bar{\alpha}\mathbb{E}[\|\nabla f(x_k)\|_2^2] + \tfrac{1}{2}L\max\{M_{s1}, M_{d1}\}\bar{\alpha}^2 \\
&\leq -\tfrac{1}{2}\delta\bar{\alpha}\mathbb{E}[\|\nabla f(x_k)\|_2^2] + \tfrac{1}{2}L\max\{M_{s1}, M_{d1}\}\bar{\alpha}^2.
\end{aligned}$$

Summing both sides of this inequality for $k \in \{1, \ldots, K\}$ yields

$$\begin{aligned}
f_{\inf} - f(x_1) &\leq \mathbb{E}[f(x_{K+1})] - f(x_1) \\
&\leq -\tfrac{1}{2}\delta\bar{\alpha}\sum_{k=1}^{K} \mathbb{E}[\|\nabla f(x_k)\|_2^2] + \tfrac{1}{2}KL\max\{M_{s1}, M_{d1}\}\bar{\alpha}^2.
\end{aligned}$$

Rearranging and dividing further by K yields the desired conclusion.

Now suppose that Algorithm 3 is run with $\{\alpha_k\}$ and $\{\beta_k\} = \{\chi\alpha_k\}$ such that the former satisfies (30). Since the second condition in (30) ensures that $\{\alpha_k\} \searrow 0$, we may assume without loss of generality that, for all $k \in \mathbb{N}$,

$$\alpha_k \leq \min\left\{\frac{\delta}{2LM_{s2}}, \frac{3\delta}{2LM_{d2}\chi^2}\right\}.$$

Thus, taking total expectation in (28) leads to

$$\begin{aligned}
&\mathbb{E}[f(x_{k+1})] - f(x_k) \\
&\leq -(\delta\alpha_k - \tfrac{1}{2}LM_{s2}\alpha_k^2 - \tfrac{1}{6}LM_{d2}\beta_k^2)\mathbb{E}[\|\nabla f(x_k)\|_2^2] + \tfrac{1}{2}LM_{s1}\alpha_k^2 + \tfrac{1}{6}LM_{d1}\beta_k^2 \\
&\leq -\tfrac{1}{2}\delta\alpha_k\mathbb{E}[\|\nabla f(x_k)\|_2^2] + (\tfrac{1}{2}LM_{s1} + \tfrac{1}{6}LM_{d1}\chi^2)\alpha_k^2.
\end{aligned}$$

Summing both sides for $k \in \{1, \ldots, K\}$ yields

$$\begin{aligned}
f_{\inf} - f(x_1) &\leq \mathbb{E}[f(x_{K+1})] - f(x_1) \\
&\leq -\tfrac{1}{2}\delta\sum_{k=1}^{K} \alpha_k\mathbb{E}[\|\nabla f(x_k)\|_2^2] + (\tfrac{1}{2}LM_{s1} + \tfrac{1}{6}LM_{d1}\chi^2)\sum_{k=1}^{K} \alpha_k^2,
\end{aligned}$$

from which it follows that

$$\sum_{k=1}^{K} \alpha_k\mathbb{E}[\|\nabla f(x_k)\|_2^2] \leq \frac{2(f(x_1) - f_{\inf})}{\delta} + \frac{LM_{s1} + \frac{1}{3}LM_{d1}\chi^2}{\delta}\sum_{k=1}^{K} \alpha_k^2.$$

The second of the conditions in (30) implies that the right-hand side here converges to a finite limit when $K \to \infty$. Then, the rest of the desired conclusion follows since the first of the conditions in (30) ensures that $A_K \to \infty$ as $K \to \infty$. □

3.2 Dynamic Method

Borrowing ideas from the two-step deterministic method in §2.1, the dynamic deterministic method in §2.2, and the two-step stochastic method in §3.1, we propose the dynamic method for stochastic optimization presented as Algorithm 4 below. After computing a stochastic Hessian estimate $H_k \in \mathbb{R}^{n\times n}$ and an independent stochastic gradient estimate $g_k \in \mathbb{R}^n$, the algorithm employs the conjugate gradient (CG) method [24] for solving the linear system $H_k s = -g_k$ with the starting point $s \leftarrow 0$. If $H_k \succ 0$, then it is well known that this method solves this system in at most n iterations (in exact arithmetic). However, if $H_k \not\succ 0$, then the method might encounter a direction of nonpositive curvature, say $p \in \mathbb{R}^n$, such that $p^T H_k p \leq 0$. If this occurs, then we terminate CG immediately and set $d_k \leftarrow p$. This choice is made rather than spend any extra computational effort attempting to approximate an eigenvector corresponding to the left-most eigenvalue of H_k. Otherwise, if no such direction of nonpositive curvature is encountered, then the algorithm chooses $d_k \leftarrow 0$. In either case, the algorithm sets s_k as the final CG iterate computed prior to termination. (The only special case is when nonpositive curvature is encountered in the first iteration; then, $s_k \leftarrow -g_k$ and $d_k \leftarrow 0$.)

For setting values for the dynamic parameters $\{L_k\}$ and $\{\sigma_k\}$, which in turn determine the stepsize sequences $\{\alpha_k\}$ and $\{\beta_k\}$, the algorithm employs stochastic function value estimates. In this manner, the algorithm can avoid computing the exact objective value at any point (which is often not tractable). In the statement of the algorithm, we use $f_k : \mathbb{R}^n \to \mathbb{R}$ to indicate a function that yields a stochastic function estimate during iteration $k \in \mathbb{N}$.

Algorithm 4 Dynamic Method for Stochastic Optimization

Require: $(L_1, \sigma_1) \in (0,\infty) \times (0,\infty)$
for $k \in \mathbb{N}$ **do**
  generate a stochastic gradient g_k and stochastic Hessian H_k
  run CG on $H_k s = -g_k$ to compute s_k and d_k (as described in the text above)
  set $\alpha_k \leftarrow 1/L_k$ and $\beta_k \leftarrow 1/\sigma_k$
  set $\hat{x}_k \leftarrow x_k + \alpha_k s_k$
  if $f_k(\hat{x}_k) > f_k(x_k)$ **then**
    set $L_{k+1} \in [L_k, \infty)$
    (optional) reset $\hat{x}_k \leftarrow x_k$
  else
    set $L_{k+1} \in (0, L_k]$
  end if
  set $x_{k+1} \leftarrow \hat{x}_k + \beta_k d_k$
  if $f_k(x_{k+1}) > f_k(\hat{x}_k)$ **then**
    set $\sigma_{k+1} \in [\sigma_k, \infty)$
    (optional) reset $x_{k+1} \leftarrow \hat{x}_k$
  else
    set $\sigma_{k+1} \in (0, \sigma_k]$
  end if
end for

As previously mentioned, we do not claim convergence guarantees for this method. However, we believe that it is well motivated by our previous algorithms. One should also note that the per-iteration cost between this method and an inexact Newton-CG method is negligible since any computed $d_k \neq 0$ comes essentially

for free from the CG routine. The only significant extra cost might come from the stochastic function estimates, though these can be made cheaper than any stochastic gradient estimate and might even be ignored completely if one is able to tune fixed values for L and σ that work well for a particular application.

3.3 Numerical Experiments

We implemented Algorithm 4 in Python 2.7.13. As a test problem, we trained a convolutional neural network to classify handwritten digits in the well-known `mnist` dataset; see [17]. Our neural network, implemented using `tensorflow`,[1] is composed of two convolutional layers followed by a fully connected layer. In each iteration, we computed stochastic gradient and Hessian estimates using independently drawn mini-batches of size 500. By contrast, the entire training dataset involves 60,000 feature/label pairs. The testing set involves 10,000 feature/label pairs.

In each iteration, our implementation runs the CG method for at most 10 iterations. If a direction of nonpositive curvature is found within this number of iterations, then it is employed as d_k; otherwise, $d_k \leftarrow 0$. In preliminary training runs, we occasionally witnessed instances in which a particularly large stochastic gradient led to a large step, which in turn spoiled all previous progress made by the algorithm. Hence, to control the iterate displacements, we scaled s_k and/or d_k down, when necessary, to ensure that $\|s_k\| \leq 10$ and $\|\beta_k d_k\| \leq 0.2\|\alpha_k s_k\|$. We witnessed similar poor behavior when the Lipschitz constant estimates were initialized to be too small; hence, for our experiments, we chose $L_1 \leftarrow 80$ and $\sigma_1 \leftarrow 100$ as the smallest values that yielded stable algorithm behavior for both algorithms. If stochastic function evaluations suggested an objective increase, then an estimate was increased by a factor of 1.2; see Steps 7 and 14 in Algorithm 4. The implementation never decreases these estimates. In addition, while the implementation always takes the step $\alpha_k s_k$ (i.e., it does not follow the optional Step 8), it does reset $x_{k+1} \leftarrow \hat{x}_k$ (recall Step 15) if/when the stochastic function estimates predict an increase in f due to the step $\beta_k d_k$.

For comparison purposes, we ran the algorithm twice using the same starting point and initial seeds for the random number generators: once with β_k being reset to zero for all $k \in \mathbb{N}$ (so that no step along d_k is ever taken) and once with it set as in Algorithm 4. We refer to the former algorithm as `SG` since it is a stochastic-gradient-type method that does not explore negative curvature. We refer to the latter as `NC` since it attempts to exploit negative curvature.

The training losses as a function of the iteration counter are shown in Figure 3. As can be seen in the plot, the performance of the two algorithms is initially very similar. However, after some initial iterations, following the negative curvature steps consistently offers additional progress, allowing `NC` to reduce the loss and increase both the training and testing accuracy more rapidly than `SG`. Eventually, the plots in each of the figures near each other (after around two epochs, when the algorithms were terminated). This should be expected as both algorithms eventually near stationary points. However, prior to this point, `NC` has successfully avoided the early stagnation experienced by `SG`.

[1] https://www.tensorflow.org/

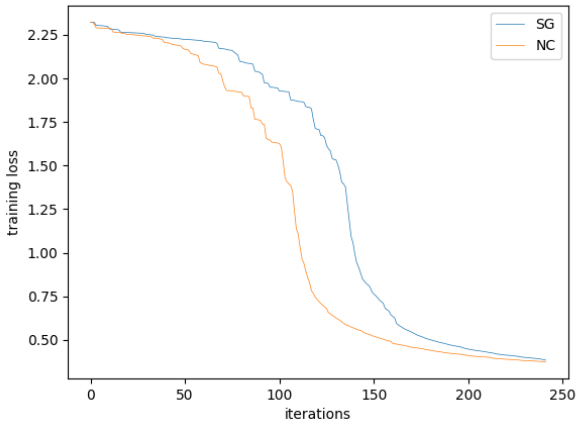


Fig. 3: Training loss as a function of the iteration counter when training a convolutional neural network over `mnist` using an SG-type method (`SG`) and one that explores negative curvature (`NC`).

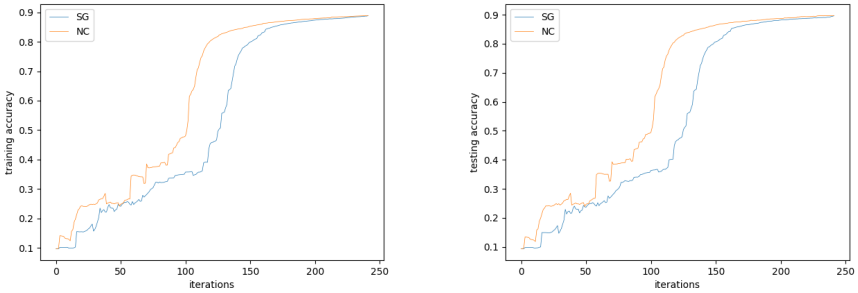


Fig. 4: Training (left) and testing (right) accuracy as a function of the iteration counter when training a convolutional neural network over `mnist` using an SG-type method (`SG`) and one that explores negative curvature (`NC`).

4 Conclusion

We have confronted the question of whether it can be beneficial for nonconvex optimization algorithms to compute and explore directions of negative curvature. We have proposed new algorithmic frameworks based on the idea that an algorithm might alternate or choose between descent and negative curvature steps based on properties of upper-bounding models of the objective function. In the case of deterministic optimization, we have shown that our frameworks possess convergence and competitive complexity guarantees in the pursuit of first- and second-order stationary points, and have demonstrated that instances of our framework outperform descent-step-only methods in terms of finding points with lower objective

values typically within fewer iterations and function evaluations. In the case of stochastic optimization, we have shown that an algorithm that employs "curvature noise" can outperform a stochastic-gradient-based approach.

References

1. E. G. Birgin and J. M. Martínez. *A Box-Constrained Optimization Algorithm with Negative Curvature Directions and Spectral Projected Gradients*, pages 49–60. Springer Vienna, Vienna, 2001.
2. L. Bottou, F. E. Curtis, and J. Nocedal. Optimization Methods for Large-Scale Machine Learning. arXiv 1606.04838, 2016.
3. Yann Dauphin, Harm de Vries, and Yoshua Bengio. Equilibrated adaptive learning rates for non-convex optimization. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1504–1512. Curran Associates, Inc., 2015.
4. Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014.
5. Ehsan Elhamifar and René Vidal. Sparse subspace clustering. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2790–2797. IEEE, 2009.
6. Anders Forsgren, Philip E Gill, and Walter Murray. Computing modified newton directions using a partial cholesky factorization. *SIAM Journal on Scientific Computing*, 16(1):139–150, 1995.
7. R. Ge, F. Huang, C. Jin, and Y. Yuan. Escaping From Saddle Points — Online Stochastic Gradient for Tensor Decomposition. In *JMLR: Workshop and Conference Proceedings*, New York, NY, USA, 2015. JMLR.
8. Philip E Gill, Vyacheslav Kungurtsev, and Daniel P Robinson. A stabilized SQP method: global convergence. *IMA Journal on Numerical Analysis*, page drw004, 2016.
9. Philip E. Gill, Vyacheslav Kungurtsev, and Daniel P. Robinson. A stabilized SQP method: superlinear convergence. *Mathematical Programming*, pages 1–42, 2016.
10. Donald Goldfarb. Curvilinear path steplength algorithms for minimization which use directions of negative curvature. *Mathematical programming*, 18(1):31–40, 1980.
11. N. I. M. Gould, S. Lucidi, M. Roma, and PH. L. Toint. Exploiting negative curvature directions in linesearch methods for unconstrained optimization. *Optimization Methods and Software*, 14(1-2):75–98, 2000.
12. Nicholas IM Gould, Dominique Orban, and Philippe L Toint. Cutest: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, 60(3):545–557, 2015.
13. Hao Jiang, Daniel P. Robinson, and René Vidal. A nonconvex formulation for low rank subspace clustering: Algorithms and convergence analysis. In *submitted to CVPR*, 2017.
14. Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.
15. Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.
16. Cornelius Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA, 1950.
17. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE, 86(11)*, pages 2278–2324, 1009.
18. J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht. Gradient descent only converges to minimizers. *Journal of Machine Learning Research*, 49, 2016.
19. Jason D. Lee, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 1246–1257, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.

20. James Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 735–742, 2010.
21. Hassan Mohy-ud-Din and Daniel P. Robinson. A solver for nonconvex bound-constrained quadratic optimization. *SIAM Journal on Optimization*, 25(4):2385–2407, 2015.
22. Jorge J Moré and Danny C Sorensen. On the use of directions of negative curvature in a modified newton method. *Mathematical Programming*, 16(1):1–20, 1979.
23. A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens. Adding Gradient Noise Improves Learning for Very Deep Networks. arXiv 1511.06807, 2015.
24. J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer New York, Second edition, 2006.
25. S. Paternain, A. Mokhtari, and A. Ribeiro. A Second Order Method for Nonconvex Optimization. arXiv 1707.08028, 2017.