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

Exploiting Negative Curvature in Deterministic and Stochastic Optimization

Frank E. Curtis[1] and Daniel P. Robinson[2]

[1]Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA
[2]Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD, USA

March 1, 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 directions, there has not yet been numerical evidence showing that such methods offer significant performance improvements. In this paper, we present new frameworks for combining descent and negative curvature directions: alternating two-step approaches and dynamic step approaches. A unique aspect of each of our frameworks is that fixed stepsizes can be used (rather than line searches or trust regions), which makes the methods viable for both deterministic and stochastic settings. For deterministic problems, we show that our dynamic framework yields significant gains in performance (in terms of lower objective function values in a fixed number of iterations) compared to a gradient descent method. We also show that while the gains offered in a stochastic setting might be more modest, they can be notable.

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, 12]. 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 stationary points, i.e., points at which the gradient of the objective is zero. This characterization is certainly accurate for line search methods, which seek to reduce the objective function by searching along descent directions. Very few researchers have designed line search (or other, such as trust region) algorithms that generate iterates that (provably) converge to second-order stationary points. 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 [17]; 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) for SG methods to first-order stationary points have been established under reasonable assumptions; e.g., see [2]. In fact, SG and its variants are the current state-of-the-art for training deep neural networks. As for methods that compute and follow negative curvature directions, it's no surprise that such methods have not been used or studied extensively since practical benefits have not even been shown in the deterministic case.

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 trust region methods, we focus on *fixed stepsize* methods that offer 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. Our focus on fixed stepsize methods is also important as it allows us to offer new strategies for stochastic optimization where, e.g., classical line search strategies are not viable. Overall, the theme of this paper can be summarized as the following:

> **In contrast to methods designed to avoid ill-effects associate 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 directions and negative curvature directions that allow us to guarantee second-order stationarity with a simple two-step iteration with fixed stepsizes; see §2.1. The analysis for this case reveals conditions (based on Lipschitz constants of the gradient and Hessian functions) that guide choices for stochastic optimization as well.
- Our second method for deterministic optimization uses the two-step iteration as motivation for the design of an adaptive choice for the direction and stepsize; see §2.2. This framework is shown to provide a significant gain in performance (measured by achieving a lower value of the objective function) when compared to only using a descent direction; see §2.4.
- For stochastic optimization, 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 refinement of that in [21], 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.

It should be obvious to the reader that computing directions of negative curvature carries an added cost that should ultimately be taken into consideration when comparing algorithm performance. As this is the first part in a longer investigation, we ignore such costs in this paper and merely demonstrate that there exists a *potential* to have practical algorithms that explore negative curvature directions. (We remark throughout the paper how such practical algorithms might be derived.) It is also worthwhile to mention that some researchers argue that negative curvature in, say, the training of deep networks should not be explored. They promote this view since some nonlinearity is introduced by the choice of activation function, and is not a true feature of the underlying problem of interest. We disagree with this view. After all, regardless of its origin, nonlinearity is a part of the optimization problem to be solved, and negative curvature directions can offer the possibility of climbing "over a hill" in ways that might not otherwise be explored.

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 [20] 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 was 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 solutions under extremely loose assumptions [7, 8]. For further instances of work employing negative curvature directions, see [1, 9, 10, 19]. Importantly, none of the papers above (or any others to the best of our knowledge) have established any gain in computational performance as a result of using negative curvature directions.

For stochastic optimization problems, there has been very 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, 13]. Instead of using negative curvature to their advantage, modern methods either ignore it (e.g., the SG method and its variants) or attempt to avoid its potential ill-effects (e.g., [4, 18]). Here, one potential ill-effect is that the reduction in the objective function over the sequence of iterates can stall in the neighborhood of a saddle point. This is potentially true even for a pure Newton iteration since saddle points are fixed points of such an iteration. The work [4] aims to avoid this potentially disappointing outcome by modifying the Newton system when computing a search direction, to promote decrease in the objective function. Although such an approach is reasonable, one might intuitively expect better performance if directions of negative curvature were used instead of simply avoided. In this critical respect, the methods that we propose are different from the prior algorithms designed for stochastic optimization.

2 Deterministic Optimization

Consider the unconstrained optimization problem

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

where $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable and bounded below. 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. We close this section with a discussion of step computation techniques that satisfy the conditions required for the algorithms and the results of numerical experiments that demonstrate the gains achieved by the dynamic method on a test set of problems.

2.1 Two-Step Method

Our first method alternates between negative curvature and descent steps, and uses fixed stepsizes. (Either can be taken first; arbitrarily, we state our algorithm and analysis in a way that involves starting 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

$$d_k^T H(x_k) d_k \leq \gamma \lambda_k \|d_k\|_2^2 < 0, \tag{2a}$$
$$g(x_k)^T d_k \leq 0, \tag{2b}$$
$$\text{and } \|d_k\|_2 \leq \theta |\lambda_k|, \tag{2c}$$

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

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

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 §2.3.

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:     set $d_k$ satisfying (2) (or $d_k \leftarrow 0$ if $\lambda_k \geq 0$)
3:     set $\hat{s}_k$ satisfying (3) (or $\hat{s}_k \leftarrow 0$ if $g(\hat{x}_k) = 0$)
4:     if $d_k = \hat{s}_k = 0$, then return $x_k$
5:     set $x_{k+1} \leftarrow x_k + \alpha \hat{s}_k + \beta d_k$
6: end for
```

We now show that Algorithm 1 converges to second-order stationarity. Critical for this analysis are bounds on the constants (α, β), which are stated in the algorithm. 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 function $\mathcal{I}_{\mathcal{D}}(k)$, which takes the value 1 if $k \in \mathcal{D}$ and is 0 otherwise.

Theorem 1. *If Algorithm 1 terminates finitely in iteration $k \in \mathbb{N}$, then $g(x_k) = 0$ and $\lambda_k \geq 0$, so 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 (2), we have

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

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

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

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

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

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

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

This inequality and f being bounded below show that

$$\sum_{k=0,k\in\mathcal{D}}^{\infty}|\lambda_k|^3 \equiv \sum_{k=0}^{\infty}|(\lambda_k)_-|^3 < \infty \tag{5a}$$

$$\text{and} \quad \sum_{k=0}^{\infty}\|g(\hat{x}_k)\|_2^2 < \infty. \tag{5b}$$

The latter bound yields

$$\lim_{k\to\infty}\|g(\hat{x}_k)\|_2 = 0, \tag{6}$$

while the former bound yields

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

from which it follows that

$$\lim_{k\to\infty}\|\hat{x}_k - x_k\|_2 = 0. \tag{7}$$

It follows from Lipschitz continuity of g, (6), and (7) 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 an 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 (5a). 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 in practice if these values are chosen poorly. In the next section, we present a method that addresses these two weaknesses.

2.2 Dynamic Method

In this section, we present an alternative to our two-step method that overcomes the weaknesses described in the end of §2.1 by incorporating a dynamic choice for the search direction.

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

$$-g(x_k)^T s_k \geq \delta \|s_k\|_2 \|g(x_k)\|_2. \tag{3'}$$

Defining, for $(L_k, \sigma_k) \in (0, \infty)^2$, the model reductions

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

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

$$f(x_k + \alpha s_k) \leq f(x_k) - m_{s,k}(\alpha) \text{ and} \tag{8a}$$
$$f(x_k + \beta d_k) \leq f(x_k) - m_{d,k}(\beta). \tag{8b}$$

These two inequalities suggest that, during iteration k, 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

$$\alpha_k := \frac{-g(x_k)^T s_k}{L_k \|s_k\|_2^2} \text{ and } \beta_k := \frac{\left(-c_k + \sqrt{c_k^2 - 2\sigma_k \|d_k\|_2^3 g(x_k)^T d_k}\right)}{\sigma_k \|d_k\|_2^3}, \tag{9}$$

where $c_k := d_k^T H(x_k) d_k$ is the curvature along 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 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.

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 2. *Algorithm 2 is well defined, i.e., either it finitely terminates or generates infinitely many iterates.*

Proof. During iteration $k \in \mathbb{N}$, Algorithm 2 might finitely terminate. If it does not, then it enters the main 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 (8a) holds for sufficiently large L_k, and (8b) holds for sufficiently large σ_k, it follows that the loop must terminate eventually, thus proving the result. □

Theorem 3. *If Algorithm 2 terminates finitely in iteration $k \in \mathbb{N}$, then $g(x_k) = 0$ and $\lambda_k \geq 0$, so x_k is second-order stationary. Otherwise, the computed iterates satisfy*

$$\lim_{k\to\infty} \|g(x_k)\|_2 = 0 \quad \textit{and} \quad \liminf_{k\to\infty} \lambda_k \geq 0. \tag{10}$$

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 represent the desired conclusions for this case. 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 (9) and the condition on s_k in (3') ensure that

$$m_{s,k}(\alpha_k) = \frac{1}{2L_k}\left(\frac{g(x_k)^T s_k}{\|s_k\|_2}\right)^2 \geq \frac{\delta^2}{2L_k}\|g(x_k)\|_2^2. \tag{11}$$

Algorithm 2 Dynamic Method

```
Require: $\eta_c \in (0,1]$, $\eta_e \in (1,\infty)$, $(L_0,\sigma_0) \in (0,\infty)^2$
 1: for $k \in \mathbb{N}$ do
 2:   set $d_k$ satisfying (2a)–(2b) (or $d_k \leftarrow 0$ if $\lambda_k \geq 0$)
 3:   set $s_k$ satisfying (3') (or $s_k \leftarrow 0$ if $g(x_k) = 0$)
 4:   if $d_k = s_k = 0$, then return $x_k$
 5:   loop
 6:     compute $\alpha_k > 0$ and $\beta_k > 0$ from (9)
 7:     if $m_{s,k}(\alpha_k) \geq m_{d,k}(\beta_k)$ then
 8:       if (8a) holds then
 9:         set $x_{k+1} \leftarrow x_k + \alpha_k s_k$
10:         set $L_k \leftarrow \eta_c L_k$
11:         exit loop
12:       else
13:         set $L_k \leftarrow \eta_e L_k$
14:       end if
15:     else
16:       if (8b) holds then
17:         set $x_{k+1} \leftarrow x_k + \beta_k d_k$
18:         set $\sigma_k \leftarrow \eta_c \sigma_k$
19:         exit loop
20:       else
21:         set $\sigma_k \leftarrow \eta_e \sigma_k$
22:       end if
23:     end if
24:   end loop
25: end for
```

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

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

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

$$m_{d,k}(\beta_k) \geq \frac{2}{3}\frac{\gamma^3}{\sigma_k^2}|(\lambda_k)_-|^3 \quad \text{for all} \quad k \in \mathbb{N}. \tag{12}$$

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

$$f(x_k) - f(x_{k+1}) \geq \max\left\{\frac{\delta^2}{2L_k}\|g(x_k)\|_2^2, \frac{2}{3}\frac{\gamma^3}{\sigma_k^2}|(\lambda_k)_-|^3\right\}. \tag{13}$$

Indeed, to show (13) 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 (8a) and $m_{s,k}(\alpha_k) \geq m_{d,k}(\beta_k)$ both hold. Combining these facts with (11) and (12) establishes (13) in this case. Second, suppose that $x_{k+1} \leftarrow x_k + \beta_k d_k$ is completed, meaning that (8b) and $m_{s,k}(\alpha_k) < m_{d,k}(\beta_k)$ both hold. Combining these facts with (11) and (12) establishes (13) in this case. Thus, (13) holds for $k \in \mathbb{N}$.

It now follows from (13) and a proof similar to that used in Theorem 1 (to establish (5a) and (5b)) 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 go about establishing the desired results in (10) by using the same arguments as used early in the proof of Theorem 1. □

Let us add a few remarks about Algorithm 2.

- Our convergence theory allows $\eta_c = 1$, in which case the Lipschitz constant estimates are monotonically increasing. In Algorithm 2, we allow $\eta_c < 1$ since this often yields better results in practice.
- If the Lipschitz constants L and σ for ∇f and $\nabla^2 f$, 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 such static parameters might work well.
- Each time through the loop, condition (8a) or (8b) 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 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)/(3')). 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 $B_k \in \mathbb{R}^{n\times n}$ that has $\kappa(B_k) \leq \delta^{-1}$ with a spectrum falling in $[\zeta, \eta]$, is to compute s_k as the (quasi-) 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 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 choose the sign of d_k (i.e., ± 1) to ensure that (2b) holds. A second approach for large-scale settings (i.e., when n is large) is to compute d_k via matrix-free Lanczos iterations [15]. Such an approach can produce a direction d_k satisfying (2a), which can then be scaled by ± 1 to yield (2b).

2.4 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 wrote an implementation of the algorithm in Python 2.7.13 that, for all $k \in \mathbb{N}$, computes $s_k = -g(x_k)$ and $d_k = \pm v_k$ (recall §2.3). Our test problems include a subset of the CUTEst collection [11]. The particular subset of problems was obtained as follows. First, we selected all of the unconstrained problems with number of variables $n \leq 2500$. Second, we ran Algorithm 2 on this subset for 100 iterations and only kept those for which at least one direction of negative curvature was used. This process left us with the 26 test problems listed under the column "Problem" in Table 1.

Using this test set, we evaluated 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 for each $k \in \mathbb{N}$ (which we refer to as Algorithm 2(s_k, d_k)). For both algorithms, we terminate when the iteration limit of 100 is reached, a high-accuracy approximate first-order point x_k is reached satisfying $\|g(x_k)\|_2 \leq 10^{-10}$, or a stepsize below 10^{-20} is computed. Such tight tolerances are chosen to allow both variants to reach the iteration limit on nearly all problems. In fact, Algorithm $2(s_k)$ terminates early only on problem HIMMELBB (due to obtaining an approximate first-order stationary point), while Algorithm $2(s_k, d_k)$ terminates early only on problems DENSCHNE, HIMMELBB, and STRATEC (with approximate first-order points for the former two problems and a small stepsize for the latter). We report in Table 1 the final value of the objective ("f-final") as well as the total number of evaluations of the objective ("$\#f$") for each test problem and both variants.

Table 1: Results on a subset of the CUTEST test problems. For both variants of Algorithm 2, the final value of the objective function ("f-final") and the number of objective function evaluations ("$\#f$") are provided after running up to 100 iterations.

	Algorithm 2 (s_k)		Algorithm 2 (s_k, d_k)	
Problem	f-final	$\#f$	f-final	$\#f$
BEALE	4.3351e-04	204	3.6128e-04	205
BIGGS6	2.7346e-01	204	7.5567e-02	202
CHNROSNB	4.3505e+01	205	3.7360e+01	206
CHNRSNBM	3.9720e+01	205	3.3355e+01	208
CUBE	1.6139e-01	205	1.4405e-01	208
DENSCHNE	9.9929e-01	202	1.4929e-21	141
DJTL	-8.2292e+03	212	-8.0318e+03	216
ERRINROS	4.1066e+01	206	4.1082e+01	208
FLETCHCR	9.8597e+02	207	9.8282e+02	208
GENROSE	3.0485e+02	207	2.9297e+02	208
HAIRY	5.0381e+01	205	2.0166e+01	214
HEART6LS	3.5758e+01	207	1.7318e+00	211
HEART8LS	2.1118e+00	206	1.1145e+00	210
HIMMELBB	1.2906e-19	72	8.9750e-20	75
HUMPS	1.0296e+01	204	2.3886e+00	210
HYDC20LS	7.3906e+05	215	5.3975e+05	225
LOGHAIRY	6.1802e+00	200	5.9037e+00	199
MEYER3	6.9773e+06	220	2.8843e+07	229
OSBORNEB	3.3387e-01	205	1.5024e-01	201
PARKCH	2.1029e+03	211	1.9788e+03	215
PENALTY2	4.7116e+13	209	3.6793e+13	232
PENALTY3	1.7627e+06	210	4.0213e+04	210
SINEVAL	4.8349e+00	207	3.1400e+00	209
STRATEC	2.6898e+03	210	-7.7360e+08	159
VIBRBEAM	5.9990e+03	223	1.2379e+03	244
WATSON	8.7542e-02	206	8.3212e-02	207

We can see in Table 1 that Algorithm $2(s_k, d_k)$ achieves a lower objective function on 23 of the 26 test problems. Importantly, it can also be observed that these gains in performance are generally not accompanied by any significant increase in the total number of evaluations of f, and in some instances (e.g., DENSCHNE and STRATEC) we can even see a significant reduction. Overall, these results clearly show that a judicious use of directions of negative curvature can improve practical performance.

3 Stochastic Optimization

Let us now consider the problem to minimize a twice continuously differentiable and bounded below 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,

$$\min_{x \in \mathbb{R}^n} \; f(x), \quad \text{where} \;\; f(x) := \mathbb{E}_\xi[F(x, \xi)]. \tag{14}$$

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 stationary points 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 strong practical performance.

3.1 Two-Step Method (SG-type with "Smart" Noise)

At an iterate x_k, let ξ_k be a random variable representing a seed for generating a vector $s_k \in \mathbb{R}^n$ and a symmetric matrix $H_k \in \mathbb{R}^{n\times n}$. 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. The vector s_k is required to satisfy

$$\begin{aligned} -g(x_k)^T\mathbb{E}_{\xi_k}[s_k] &\geq \delta\|g(x_k)\|_2^2, && \text{(15a)}\\ \zeta\|g(x_k)\|_2 \leq \mathbb{E}_{\xi_k}[\|s_k\|_2] &\leq \eta\|g(x_k)\|_2, \quad \text{and} && \text{(15b)}\\ \mathbb{E}_{\xi_k}[\|s_k\|_2^2] &\leq M_{s1} + M_{s2}\|g(x_k)\|_2^2, && \text{(15c)}\end{aligned}$$

for some $(\delta,\zeta) \in (0,1]\times(0,1]$, $\eta \in [1,\infty)$, and $(M_{s1}, M_{s2}) \in (0,\infty)\times(1,\infty)$. For example, these conditions are satisfied if s_k is an unbiased estimate of $\nabla f(x_k)$ with second moment bounded as in (15c). On the other hand, for simplicity, let us assume that H_k is an unbiased Hessian estimate such that $\mathbb{E}_{\xi_k}[H_k] = \nabla^2 f(x_k)$. 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

$$\begin{aligned} d_k^T H_k d_k \leq \gamma\lambda_k\|d_k\|_2^2 < 0 \quad &\text{given } H_k && \text{(16a)}\\ \text{and} \quad \mathbb{E}_{\xi_k}[\|d_k\|_2^2] \leq M_{d1} &+ M_{d2}\|g(x_k)\|_2^2 && \text{(16b)}\end{aligned}$$

for some $\gamma \in (0,1]$ and 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 $\|s_k\|_2^2$ where s_k satisfies (15).

Note that our conditions in (16) 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 enforceable. 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, we have $\mathbb{E}[\omega_k d_k] = 0$ such that, ultimately, $\omega_k d_k$ is merely a "smart" choice for adding noise to s_k in a manner that ideally follows a negative curvature direction. This leads to Algorithm 3 below.

Algorithm 3 Two-Step Method

```
Require: choose $\{\alpha_k\} \subset \mathbb{R}_{++}$ and $\{\beta_k\} \subset \mathbb{R}_{++}$
1: for $k \in \mathbb{N}$ do
2:     set $s_k$ satisfying (15)
3:     set $d_k$ satisfying (16)
4:     set $\omega_k$ uniformly in $[-1,1]$
5:     set $x_{k+1} \leftarrow x_k + \alpha_k s_k + \beta_k \omega_k d_k$
6: end for
```

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

Lemma 4. *There exist positive scalar constants* $(\rho_1,\rho_2,\rho_3,\rho_4)$ *such that, for all* $k \in \mathbb{N}$,

$$\mathbb{E}_{(\xi_k,\omega_k)}[f(x_{k+1})] - f(x_k) \leq -(\delta\alpha_k - \rho_1\alpha_k^2 - \rho_2\beta_k^2)\|g(x_k)\|_2^2 + \rho_3\alpha_k^2 + \rho_4\beta_k^2.$$

Proof. From Lipschitz continuity of g, it follows that

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

Taking expectations with respect to the distribution of (ξ_k, ω_k) and using (15)–(16), it follows that

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

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 to some finite threshold (if fixed stepsizes are used) or vanishes completely (if, say, $\alpha_k = \mathcal{O}(1/k)$ and $\beta_k = \mathcal{O}(1/k)$ with $\beta_k \leq \chi\alpha_k$ for some constant $\chi \in (0,\infty)$ for all $k \in \mathbb{N}$).

3.2 Dynamic Method

Borrowing ideas from the dynamic (deterministic) method in §2.2 and the two-step (stochastic) method in §3.1, we propose the following method. Note that, in the models $m_{s,k}$ and $m_{d,k}$ as well as in (9), the method uses g_k and H_k in place of $g(x_k)$ and $H(x_k)$, respectively.

Algorithm 4 Dynamic Method

```
Require: $(L_0, \sigma_0) \in (0,\infty)^2$
 1: for $k \in \mathbb{N}$ do
 2:     generate a stochastic gradient $g_k$ and Hessian $H_k$
 3:     set $s_k$ satisfying (15)
 4:     set $d_k$ satisfying (16) and $g_k^T d_k \leq 0$
 5:     compute $\alpha_k > 0$ and $\beta_k > 0$ from (9)
 6:     if $m_{s,k}(\alpha_k) \geq m_{d,k}(\beta_k)$ then
 7:         set $x_{k+1} \leftarrow x_k + \alpha_k s_k$
 8:         choose $L_{k+1}$ and set $\sigma_{k+1} \leftarrow \sigma_k$
 9:     else
10:         set $x_{k+1} \leftarrow x_k + \beta_k d_k$
11:         choose $\sigma_{k+1}$ and set $L_{k+1} \leftarrow L_k$
12:     end if
13: end for
```

An aspect of the algorithm left unspecified is the manner in which the Lipschitz constant estimates are updated. One possibility, as used in our experiments for the next subsection, is to increase/decrease an estimate depending on increase/decrease observed in stochastic function estimates.

3.3 Numerical Experiments

In this section, we demonstrate practical benefits of following negative curvature directions in a stochastic setting. The improvements are somewhat more modest than seen in §2.4, but are still notable. For our experiments, we consider the training of a fully connected neural network with one hidden layer and squared ℓ_2-norm loss function. The network is defined by $x^{(j)} = s(W_j x^{(j-1)} + b_j)$ for $j \in \{1,2\}$ with componentwise sigmoid activation s. We trained the network over the first 10^3 elements in each of the well-known `mnist` and `cifar-10` datasets; see [16] and [14].

We experimented with our dynamic method, Algorithm 4, where stochastic gradient and Hessian estimates were computed using mini-batches of size 100. In each iteration, the stochastic gradient and Hessian were computed using the same mini-batch. In order to demonstrate the full potential benefits of exploring negative curvature directions, the eigenvector corresponding to the leftmost eigenvalue of each stochastic Hessian estimate was computed with high accuracy. It should be noted, however, that in a practical implementation one could employ an iterative method that only requires (stochastic) Hessian-vector products, which are known to be relatively inexpensive for problems of the type considered here; e.g., see [2].

We ran each algorithm instance for 2000 iterations (i.e., 20 epochs) and compare the results based on the training losses obtained. The Lipschitz constant estimates were initialized for both datasets at $L_0 = 50$ and $\sigma_0 = 5$. We chose to set $L_{k+1} \geq L_k$ and $\sigma_{k+1} \geq \sigma_k$ for all $k \in \mathbb{N}$, using an increase factor of $\eta_e = 2$ whenever the training loss evaluated over the mini-batch used in iteration k decreased.

The training losses as a function of the iteration counter are shown in Figure 1 with the final losses (at iteration $k = 2000$) listed in Table 2. As can be seen in the plots, considering the exploration of negative curvature allows the algorithm to achieve lower training losses more quickly, and the gap between the losses continues to widen as the algorithm proceeds. At some point, the curves should be expected to reconnect as a minimizer is reached (assuming the algorithms reach comparable objective values), but such initial improvement has the potential to be beneficial.

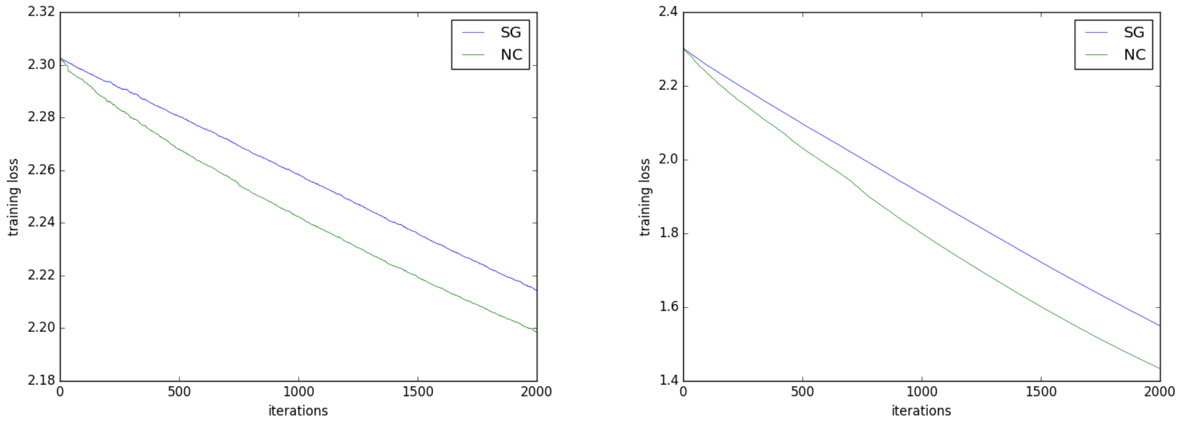

Figure 1: Training losses as a function of k (i.e., the iterations) when training a fully connected neural network over data in `mnist` (left) and `cifar-10` (right) using an SG-type method ("SG") and one that explores negative curvature ("NC").

Table 2: Results on training a fully connected neural network with one hidden layer. The final value of the training loss ("loss") is provided after running 2000 iterations.

Problem	Algorithm 4(s_k) f-final	Algorithm 4(s_k, d_k) f-final
`cifar-10`	2.2144e+00	2.1986e+00
`mnist`	1.5493e+00	1.4334e+00

4 Conclusion

We have shown that the computation and careful use of negative curvature directions allow optimization methods to achieve theoretical convergence guarantees (even to second-order stationarity) and, potentially, gains in practical performance in deterministic and stochastic settings.

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] Philip E Gill, Vyacheslav Kungurtsev, and Daniel P Robinson. A stabilized SQP method: global convergence. *IMA Journal on Numerical Analysis*, page drw004, 2016.

[8] Philip E. Gill, Vyacheslav Kungurtsev, and Daniel P. Robinson. A stabilized SQP method: superlinear convergence. *Mathematical Programming*, pages 1–42, 2016.

[9] Donald Goldfarb. Curvilinear path steplength algorithms for minimization which use directions of negative curvature. *Mathematical programming*, 18(1):31–40, 1980.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[15] 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.

[16] 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.

[17] J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht. Gradient descent only converges to minimizers. *Journal of Machine Learning Research*, 49, 2016.

[18] James Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 735–742, 2010.

[19] 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.

[20] 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.

[21] 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.