J. Indian Inst. Sci.
A Multidisciplinary Reviews Journal
ISSN: 0970-4140 Coden-JIISAD

REVIEW ARTICLE

Gradient Methods for Non-convex Optimization

Prateek Jain[*]

Abstract | Non-convex optimization forms bedrock of most modern machine learning (ML) techniques such as deep learning. While non-convex optimization problems have been studied for the past several decades, ML-based problems have significantly different characteristics and requirements due to large datasets and high-dimensional parameter spaces along with the statistical nature of the problem. Over the last few years, there has been a flurry of activity in non-convex optimization for such ML problems. This article surveys a few of the foundational approaches in this domain.

Keywords: *Non-convex optimization, Machine learning, First-order methods, SVRG*

1 Introduction

For most of the survey, we focus on the unconstrained optimization problem:

$$\min_{w\in\mathbb{R}^d} f(w), \tag{1}$$

where $f:\mathbb{R}^d\to\mathbb{R}$ is a scalar-valued function. The problem models several critical ML problems such as least squares regression, training SVMs, and training neural networks. The problem has been extensively studied in the literature, especially for convex functions[5] defined below.

Definition 1 (*Convex functions*) $f:\mathbb{R}^d\to\mathbb{R}$ is a convex function if for all $x,y\in$ Domain(f) and $1\geq\lambda\geq 0$, we have: $f(\lambda x+(1-\lambda)y)\leq\lambda f(x)+(1-\lambda)f(y)$. Equivalently,

$$f(x)+\langle\nabla f(x),y-x\rangle\leq f(y),\quad \text{or if,}\quad \nabla^2 f(x)\succeq 0,$$

where $\nabla f(x)$ is the gradient of f computed at x and $\nabla^2 f(x)$ is the Hessian of f at x.

For convex functions, it is well known that (1) can be solved optimally with techniques as simple as gradient descent (GD). That is, Gradient Descent (Algorithm 1) requires finite many iterations of GD to obtain an ε-approximate solution to (1), i.e.,

$$f(w_T)\leq\min_{w\in\mathbb{R}^d} f(w)+\varepsilon,$$

where w_T is the Tth iterate of the GD algorithm. Throughout the article, w_* refers to an optimal solution of (1).

Note that Algorithm 1 is defined only for differentiable functions f. While it can be extended to non-differentiable convex functions using sub-gradients, in this paper we will focus only on smooth differentiable functions.

Definition 2 (*Lipschitz continuous gradient functions*) $f:\mathbb{R}^d\to\mathbb{R}$ is L-smooth or L-Lipschitz Continuous Gradient if the following holds for all $x,y\in$ Domain(f): $\|\nabla_w f(x)-\nabla_w f(y)\|\leq L\|x-y\|$. Equivalently,

$$\|\nabla^2 f(x)\|_2\leq H,\quad \text{or,}$$
$$f(x)\leq f(y)+\langle\nabla f(x),y-x\rangle+\frac{L}{2}\|y-x\|^2.$$

Throughout the article, we use $\|x\|$ to denote L_2 norm of a vector. Similarly, $\|X\|$ denotes operator norm of X.

For smooth convex optimization, it is well known that Algorithm 1 converges to an ε-optimal solution in $T=O(L/\varepsilon^2)$ iterations, where L is the smoothness constant of f. Hence, the time complexity of the algorithm is $O((d+T_f)/\varepsilon^2)$, where T_f is the time complexity of computing gradient of f at any point w.

Problem (1) is significantly more complex for non-convex f, as the problem is in general NP

This article belongs to the Special issue—Recent Advances in Machine Learning.
[1] *Microsoft Research, Bengaluru, India.*
[*] *prajain@microsoft.com*

hard. For convex f, $\nabla_w f(w) = 0$ implies global optimality. However, for non-convex f, such critical points do not even imply local optimality. In fact, finding even local optima is NP hard in general. Hence, various methods in the literature try to either find the first-order or second-order stationary points of f. We first define the first-order stationary points (FOSP) of a function f below.

Definition 3 (*First-order stationary point (FOSP)*) w is a first-order stationary point (FOSP) of $f : \mathbb{R}^d \to \mathbb{R}$ iff the following holds: $\nabla_w f(w) = 0$. w is an ε-FOSP of f, iff: $\|\nabla_w f(w)\| \leq \varepsilon$. For "random" w, the equivalent condition is: $\mathbb{E}_w[\|\nabla_w f(w)\|] \leq \varepsilon$.

Note that FOSP can be a saddle point or even a local maxima (see Fig. 1). Hence, several works in the literature have studied methods for finding the second-order stationary point (SOSP) defined below.

Definition 4 (*Second-order stationary point (SOSP)*) w is a second-order stationary point (SOSP) of $f : \mathbb{R}^d \to \mathbb{R}$ iff the following holds: $\nabla_w f(w) = 0$, $\nabla_w^2 f(w) \succeq 0$. w is an (ε, ρ)-SOSP of f, iff:

$$\|\nabla_w f(w)\| \leq \varepsilon, \quad \nabla_w^2 f(w) \succeq -\sqrt{\rho\varepsilon} I.$$

That is, SOSP (see Fig. 1) avoids first-order saddle points. Now one can define higher order saddle points such as third-order stationary points, but in general computing fourth- or higher order stationary points is NP hard[4].

Note that while FOSP can be obtained using more standard techniques such as gradient descent or stochastic gradient descent, SOSP requires more carefully designed algorithms such as cubic regularization or noisy stochastic gradient descent and requires significantly more involved analysis in general.

In Sect. 2 we will discuss various methods for finding FOSP in different settings. Section 3 discusses a few techniques for finding SOSP. Finally, we conclude with Sect. 4.

2 Methods for Finding FOSP

The goal of FOSP of a function f is to find a point w s.t. $\nabla_w f(w) = 0$. While there exist several techniques to achieve this goal[5], most of the techniques use second- or higher order derivatives of f and hence are not suitable for large-scale ML problems. In contrast, first-order techniques based only on gradient of f perform well in practice. In the remaining section, we will focus on such first-order techniques for finding FOSP.

2.1 Gradient Descent for FOSP

Algorithm 1 Gradient Descent (GD)

Input: Oracle access to $f(w)$, $\nabla_w f(w)$. Initial point: w_0. Step-size: η.

1: **for** $t = 0, 1, \ldots, T$ **do**
2: $\quad w_{t+1} = w_t - \eta \nabla_w f(w)$
3: **end for**
4: **return** w_k s.t. $k = \arg\min_{k \in [T]} \|\nabla_w f(w_k)\|$

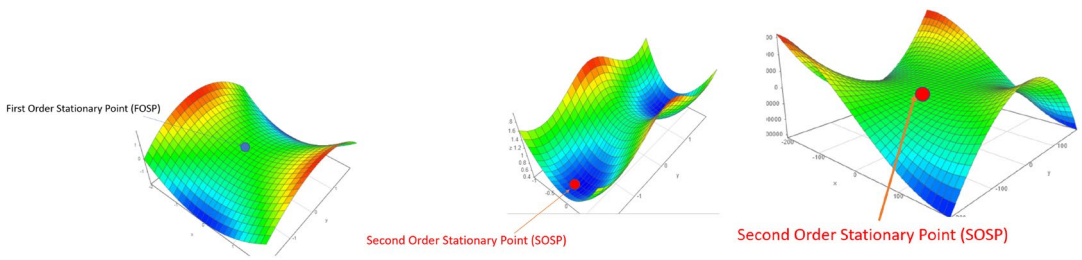

Figure 1: Various functions of two variables and their corresponding FOSP and SOSP.

In this section, we study gradient descent for finding FOSP. While standard analysis of GD exists for convex functions, it turns out that a similar analysis can be applied to arbitrary non-convex problems for finding FOSP.

Theorem 1 (GD finds FOSP) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be an L-smooth non-convex function. Then the output of Algorithm 1 with $\eta = \frac{1}{L}$ and $T = O(\frac{L(f(w_0)-f(w_*))}{\varepsilon^2})$ is an ε-FOSP (Definition 3) of Problem (1). w_0 is the initial point provided to Algorithm 1, and $f(w_*) = \min_w f(w)$ is the value at an optima.*

Proof

Using smoothness property (Definition 2), we have

$$\begin{aligned} f(w_{t+1}) &\leq f(w_t) + \langle \nabla_w f(w_t), w_{t+1} - w_t \rangle \\ &\quad + \frac{L}{2}\|w_{t+1} - w_t\|^2, \\ &\overset{\zeta_1}{=} f(w_t) - \eta\left(1 - \frac{L\cdot\eta}{2}\right)\|\nabla_w f(w_t)\|^2, \\ &\overset{\zeta_2}{\leq} f(w_t) - \frac{L}{2}\|\nabla_w f(w_t)\|^2, \end{aligned}$$

where ζ_1 follows from definition of w_{t+1} and ζ_2 follows from $\eta = \frac{1}{L}$. Theorem follows by adding the above equation for all T.

Note that for non-convex functions GD requires $O(1/\varepsilon^2)$ iterations while for convex functions, it requires $O(1/\varepsilon)$ iterations to achieve similar sub-optimality. Now, one can accelerate the convergence using momentum-based techniques[21, 23]; however, still the gap between the rates for non-convex functions and convex functions remains a lively area for research.

Algorithm 2 presents a pseudo-code of the acceleration-based algorithm for general non-convex functions, and the below Theorem bounds the number of iterations required to ε -approximately solve the FOSP problem.

Theorem 2 (Accelerated gradient descent for FOSP) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be an L-smooth non-convex function. Then there exists $\alpha_t, \beta_t, \beta_t$ s.t. Algorithm 2 with $T = O(\frac{\sqrt{L(f(w_0)-f(w_*))}}{\varepsilon})$ outputs an ε-FOSP (Definition 3) of Problem (1). w_0 is the initial point provided to Algorithm 2, and $f(w_*) = \min_w f(w)$ is the value at an optima.*

See[13] for a proof of the above theorem.

Algorithm 2 Accelerated Gradient Descent (AGD)

Input: Oracle access to $f(w)$, $\nabla_w f(w)$. Initial point: w_0. Maximum no. of iterations: T.

1: $w_0^{ag} = w_0$
2: **for** $t = 0, 1, \ldots, T$ **do**
3: $\quad w_{t+1}^{md} = (1 - \alpha_t) w_t^{ag} + \alpha_t w_t$
4: $\quad w_{t+1} = w_t - \eta_t \nabla_w f(w_{t+1}^{md})$
5: $\quad w_{t+1}^{ag} = w_t^{ag} - \beta_t \nabla_w f(w_{t+1}^{md})$
6: **end for**
7: **return** w_k s.t. $k = \arg\min_{k\in[T]} \|\nabla_w f(w_k)\|$

2.2 Stochastic Gradient Methods for FOSP

Algorithm 3 Stochastic Gradient Descent (SGD)

Input: Oracle access to $f(w) = \frac{1}{n}\sum_i f_i(w)$, $\nabla_w f_i(w)$. Initial point: w_0. Step-size: η. Maximum no. of iterations: T.

1: **for** $t = 0, 1, \ldots, T$ **do**
2: $\quad i_t \sim Unif[1, n]$
3: $\quad w_{t+1} = w_t - \eta \nabla_w f_{i_t}(w_t)$
4: **end for**
5: **return** w_T

Typical ML-based optimization problems require optimizing a function over several data points. For example, deep learning requires minimizing a loss function which in itself is a sum of loss functions over individual data points. That is, the function f can be written as a finite sum $f(w) = \frac{1}{n}\sum_i f_i(w)$.

Thus, in general, computing gradient itself requires $O(nd)$, where n is the number of points and d is the number of parameters. Now, in general, n is very large. For example, ImageNet[9], a standard image classification benchmark has about 14M data points. So computing gradient itself is prohibitive.

However, one can exploit the fact that the function f is a finite sum function, to devise easily computable randomized gradients which in expectation match the actual gradient. That is, at tth iteration, we can sample a function f_{i_t} and just compute gradient of the sampled function. Note that $\mathbb{E}_{i_t\sim \text{Unif}[1,n]}[\nabla_w f_{i_t}(w)] = \nabla_w f(w)$ for any fixed w with respect to i_t.

Such proxy for gradient is utilized by the well-known stochastic gradient descent (SGD) algorithm (Algorithm 3). Interestingly, similar to GD, analysis for SGD also follows easily from the smoothness definition.

$$\begin{aligned}\mathbb{E}[f(w_{t+1})] &\le \mathbb{E}[f(w_t)] + \mathbb{E}\Big[\langle \nabla_w f(w_t), w_{t+1} - w_t\rangle \\ &\quad + \frac{L}{2}\|w_{t+1} - w_t\|^2\Big], \\ &\overset{\zeta_1}{=} \mathbb{E}[f(w_t)] - \eta\left(1 - \frac{\eta\cdot L}{2}\right)\|\nabla_w f(w_t)\|^2 \\ &\quad + \frac{L}{2}\eta^2\sigma^2,\end{aligned}$$

where $\sigma^2 = \max_t \mathbb{E}[\|\nabla_w f_{i_t}(w_t)\|^2] - \|\nabla_w f(w_t)\|^2$ and ζ_1 follows by definition of w_{t+1} and the fact that i_t is selected independent of w_t.

Adding the above terms and rearranging, we have

$$\begin{aligned}&\min_{k\in[T]} \mathbb{E}[\|\nabla_w f(w_k)\|^2] \\ &\quad\le \frac{2(f(w_0) - f(w_*))}{\eta} + L\sigma^2\eta, \\ &\quad\le \frac{\sigma\sqrt{2L\cdot(f(w_0) - f(w_*))}}{\sqrt{T}} \le \varepsilon^2,\end{aligned}$$

where the second inequality follows by setting $\eta = \frac{\sqrt{f(w_0)-f(w_*)}}{\sigma\cdot\sqrt{L}\cdot\sqrt{T}}$.

Note that while the number of iterations is larger for SGD (in terms of sub-optimality ε), the time complexity per step is $O(n)$ smaller[12]. Hence, the usage of such methods for deep learning where ε is not required to be extremely small.

Algorithm 4 Stochastic Variance Reduction Gradient (SVRG)

Input: Oracle access to $f(w) = \frac{1}{n}\sum_i f_i(w)$, $\nabla_w f_i(w)$. Initial point: w_0. Step-size: η. Maximum no. of iterations: T. Batch size: m

1: $w_m^0 = w_0$
2: **for** $\tau = 1, \ldots, T$ **do**
3: $\quad w_0^\tau = w_m^0$
4: $\quad$ **for** $t = 0, 1, \ldots, m$ **do**
5: $\quad\quad i_t \sim Unif[1, n]$
6: $\quad\quad w_{t+1}^\tau = w_t^\tau - \eta[\nabla_w f_{i_t}(w_t^\tau) - \nabla_w f_{i_t}(w_0^\tau) + \frac{1}{n}\sum_i \nabla_w f_i(w_0^\tau)]$
7: $\quad$ **end for**
8: **end for**
9: **return** w_m^k s.t. $k = \arg\min_{k\in[T]} \|\nabla_w f(w_m^k)\|$

Theorem 3 (SGD finds FOSP) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be an L-smooth non-convex function. Then $T = O(\frac{L\sigma^2(f(w_0)-f(w_*))}{\varepsilon^4})$th iterate of Algorithm 3 with $\eta = \frac{\sqrt{f(w_0)-f(w_*)}}{\sigma\cdot\sqrt{L}\cdot\sqrt{T}}$ satisfies:*

$$\min_{k\in[T]} \mathbb{E}[\|\nabla_w f(w_k)\|^2] \le \varepsilon^2.$$

w_0 is the initial point provided to Algorithm 3, and $f(w_*) = \min_w f(w)$ is the value at an optima.

Proof

Using smoothness property (Definition 2), we have

However, the time complexity is dependent on the variance quantity which can be large. Similar to convex optimization literature, there are several variance reduction techniques to further improve the time complexity. Below we present one such approach. Algorithm 4 presents a pseudo-code for the NC-SVRG algorithm and its iteration complexity is given by

Theorem 4 (SVRG finds FOSP) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be an L-smooth non-convex function. Then the output of Algorithm 4 with $T = O(\frac{L(f(w_0)-f(w_*))\cdot n^{-1/3}}{\varepsilon^2})$, $m = O(n)$, and $\eta = O(1/(Ln^{2/3}))$ satisfies*

$$\min_{k\in[T]} \mathbb{E}[\|\nabla_w f(w_k)\|^2] \le \varepsilon^2.$$

w_0 is the initial point provided to Algorithm 4, and $f(w_*) = \min_w f(w)$ is the value at an optima.

Note that the above theorem shows that the total number of gradient calls made by the algorithm scales as $O(L(f(w_0) - f(w_*)) \cdot (n + \frac{n^{2/3}}{\varepsilon^2}))$. See[3, 25] for a proof of the above theorem.

2.3 Summary

Finding FOSP is an important problem in the non-convex optimization literature and several algorithms have been proposed recently. In particular, a lot of progress has been made for the finite sum case. Table 1 summarizes the current state-of-the-art for finding FOSP of smooth but non-convex functions. Also see companion article[22] for a discussion on finding FOSP/SOSP for finite sum setting. Recently, several other problem settings have been considered. For example, if the function f is given $f(w) = \frac{1}{n}\sum_i f_i(w) + g(w)$, where g is a non-smooth but convex function[14, 24]. Despite the progress, several critical problems still persist in the area. For example, can we obtain tighter rates for convergence to FOSP for various methods discussed above. SVRG for convex functions has a $\sqrt{n}$ term while for non-convex functions it is $n^{2/3}$, so another interesting open question is can we bridge this gap in time complexity.

Table 1: FOSP: no. of first-order oracle calls (gradient computations) required by different methods for converging to ε-FOSP (Definition 3) of f when f is a non-convex and a convex function, respectively.

Algorithm	No. of gradient calls (non-convex)	No. of gradient calls (convex)
GD (Algorithm 1)	$O\left(\frac{1}{\varepsilon^2}\right)$	$O\left(\frac{1}{\varepsilon}\right)$
AGD (Algorithm 2)	$O\left(\frac{1}{\varepsilon}\right)$	$O\left(\frac{1}{\sqrt{\varepsilon}}\right)$

Assumption: all problem parameters such and L, $f(w_0) - f(w_*)$ are constant

3 Methods for Finding SOSP

As discussed in the previous section, standard gradient descent type techniques can find FOSP, which is not entirely surprising as the goal is to only guarantee the first-order stationary ($\nabla_w f(w) = 0$). However, SOSP requires a second-order condition $\nabla^2_w f(w) \succeq 0$ and hence the standard gradient descent techniques fail. For example, if the gradient descent method is initialized at a first-order saddle point, then it will not even move as the gradient is 0.

Hence, it was widely believed that we require a "second-order" method for finding SOSP, i.e., a method that has access to the Hessian of the function f along with the gradient. Nesterov and Polyak[20] introduced one such method that updates w in each iteration using

$$\begin{aligned} w_{t+1} = \arg\min_w & f(w_t) + \langle f(w_t), w - w_t\rangle \\ & + \frac{1}{2}(w - w_t)^{\mathrm{T}} \nabla^2_w f(w_t)(w - w_t) \\ & + \frac{1}{6}\|w - w_t\|^3. \end{aligned} \quad (2)$$

Nesterov and Polyak[20] showed that the above method converges to an SOSP in $O(\frac{1}{\varepsilon^{1.5}})$ iterations

Table 2: FOSP in finite-sum setting ($f(w) = \sum_{i=1}^n f_i(w)$): No. of first-order oracle calls (individual gradient computations $\nabla f_i(w)$) required by different methods for converging to ε-FOSP (Definition 3) of f when f is a non-convex and a convex function, respectively.

Algorithm	No. of gradient calls (non-convex)	No. of gradient calls (convex)
GD (Algorithm 1)	$O\left(\frac{n}{\varepsilon^2}\right)$	$O\left(\frac{n}{\varepsilon}\right)$
AGD (Algorithm 2)	$O\left(\frac{n}{\varepsilon}\right)$	$O\left(\frac{n}{\sqrt{\varepsilon}}\right)$
SGD (Algorithm 3)	$O\left(\frac{1}{\varepsilon^4}\right)$	$O\left(\frac{1}{\varepsilon^2}\right)$
SVRG (Algorithm 4)	$O\left(n + \frac{n^{2/3}}{\varepsilon^2}\right)$	$O\left(n + \frac{\sqrt{n}}{\varepsilon^2}\right)$
MSVRG[25]	$O\left(\min\left(\frac{1}{\varepsilon^4}, \frac{n^{2/3}}{\varepsilon^2}\right)\right)$	$O\left(n + \frac{\sqrt{n}}{\varepsilon^2}\right)$

Assumption: all problem parameters such as L, $f(w_0) - f(w_*)$ are constant

and each iteration requires computation of the Hessian, which can be significantly expensive. Several recent results can relax this Hessian computation requirement if they are provided access to the product of the Hessian with an arbitrary vector, albeit with worse convergence rates[1, 27]. Unfortunately, even computing Hessian-vector product is significantly expensive for standard ML problems such as deep learning training which leads to the following question:

can we find SOSP using first-order methods, i.e., by just using gradients of f?

Interestingly, the answer is *yes*, assuming an additional condition on f which is that the Hessian is Lipschitz continuous.

Definition 5 (*Lipschitz continuous Hessian*) $f : \mathbb{R}^d \to \mathbb{R}$ is M-Lipschitz continuous Hessian if the following holds for all $x, y \in \text{Domain}(f)$: $\|\nabla_w^2 f(x) - \nabla_w^2 f(y)\| \leq \rho \|x - y\|$.

Below, we describe one such technique for finding SOSP, and then extend it to the finite-sum setting, and summarize with some more advanced results.

3.1 Noisy Gradient Descent for SOSP

Algorithm 5 Noisy Gradient Descent (NGD)

Input: Oracle access to $f(w)$, $\nabla_w f(w)$. Initial point: w_0. Step-size: η. Maximum no. of iterations: T.

for $t = 1, 2, \ldots, T$ **do**
  if $\|\nabla_w f(w_t)\| \geq \varepsilon$ **then**
    $w_{t+1} = w_t - \eta \nabla_w f(w_t)$
  else
    $\tau = 0$
    $w_{t+1} = w_t + \eta \zeta_t$, $\zeta_t \sim \mathcal{N}(0, I)$
    while $\tau \leq r$ **do**
      $t = t + 1$, $\tau = \tau + 1$
      $w_{t+1} = w_t - \eta \nabla_w f(w_t)$
    end while
  end if
end for
return w_T

As the standard gradient descent can get stuck in a FOSP due to 0 gradient, a natural approach would be to perturb the solution so that it can escape the first-order saddle point. Noisy gradient descent techniques precisely exploit this intuition; see Algorithm 5 for a pseudo-code of the algorithm. Note that while the perturbation should be large enough to ensure escaping a saddle point, but it should not be large enough to escape local minima or global minima, else the method might not even converge. The following theorem shows that such a fine balance can be struck and the algorithm converges to a SOSP in $O(1/\varepsilon^2)$ iterations.

Theorem 5 (Convergence to SOSP) *Let $f : \mathbb{R}^d \to \mathbb{R}$ be an L-smooth, ρ-Hessian–Lipschitz Continuous function. Then the output of Algorithm 5 with $T = O(\text{poly}(d/\varepsilon))$, $R \leq 5 \log d/(\eta \cdot \sqrt{\rho\varepsilon})$, and $\eta \leq \text{poly}(d, 1/\varepsilon, \rho, L, f(w_0) - f(w_*))$ is an ε -SOSP, i.e.,*

$$\mathbb{E}[\|\nabla_w f(w_T)\|^2] \leq \varepsilon^2, \mathbb{E}[\nabla_w^2 f(w_T)] \succeq -\sqrt{\rho\varepsilon}.$$

1 *Proof*

The proof can be broken into three parts:

1. $\|\nabla_w f(w_t)\| > \varepsilon$: during this phase, Algorithm 5 performs standard Gradient Descent. Hence, the standard GD analysis would guarantee convergence to FOSP ($\|\nabla_w f(w_t)\| \leq \varepsilon$) in $O(\frac{1}{\varepsilon^2})$ iterations.
2. $\|\nabla_w f(w_t)\| \leq \varepsilon$ but w_t is *not* a SOSP: for such w_t, we need to show that after r iterations of gradient descent, $f(w_{t+r})$ is significantly smaller than $f(w_t)$ thus escaping the first-order saddle point.
3. $\|\nabla_w f(w_t)\| \leq \varepsilon$ but w_t is a SOSP: for such w_t, we need to show that after r iterations of gradient descent, $f(w_{t+r})$ is not much larger than $f(w_t)$, thus it does not escape the second-order saddle point.

Combining all the above three parts would give us the desired result. As mentioned above, the first part follows from Theorem 1 directly.

Escaping first-order stationary point For this second part, lets recall update for

$w_{t+r} = w_{t+r-1} - \eta \nabla_w f(w_{t+r-1})$. Now, as f is Hessian Lipschitz Continuous, we can essentially replace f by a quadratic function without significant error. That is, define

$$\hat{f}(w) = f(w_t) + \frac{1}{2}(w - w_t)^{\mathrm{T}} H (w - w_t),$$

where $H = \nabla_w^2 f(w_t)$. Also, define

$$\widehat{w}_{t+r+1} = \widehat{w}_{t+r} - \eta \nabla_w \hat{f}(\widehat{w}_{t+r}), \quad 1 \le r \le R,$$

where $\widehat{w}_{t+1} = w_{t+1} = w_t + \zeta_t$.

It is easy to observe that

$$\widehat{w}_{t+r+1} - w_t = (I - \eta H)^r \zeta_t.$$

That is, $\widehat{w}_{t+r+1}$ is the result of r power-method updates on $M = I - \eta H$. Note that as w_t is a saddle point, i.e., $\lambda_{\min}(H) \le -\sqrt{\rho\varepsilon}$ where $\lambda_{\min}(A)$ is the smallest (or most negative) eigenvalue of A. Hence, $\widehat{w}_{t+r+1}$ converges to the eigenvector of M corresponding to largest eigenvalue, i.e., it converges to the eigenvector corresponding to most negative eigenvalue of H. Hence, $\hat{f}(\widehat{w}_{t+R})$ should be significantly smaller than $f(w_t)$.

Formally, using $\mathbb{E}[\zeta_t] = 0$ and $\mathbb{E}[\zeta_t \zeta_t^{\mathrm{T}}] = \eta^2 I$:

$$\begin{aligned}\hat{f}(\widehat{w}_{t+R}) &= f(w_t) + \frac{\eta^2}{2} tr((I - \eta H)^{2R} H) \\ &\le f(w_t) - \eta/2.\end{aligned}$$

The inequality follows using $R = \frac{5\log(d)}{\eta\sqrt{\rho\varepsilon}}$ and $\eta \le 1/\|H\| \le 1/L$. Finally, using Hessian Lipschitz continuity and straightforward arguments, we get $f(w_{t+R}) \le \hat{f}(\widehat{w}_{t+R}) + \eta/4$. Hence, we get

$$f(w_{t+R}) \le f(w_t) - \eta/4. \tag{3}$$

Entrapment near SOSP For this third part, we again use $\hat{f}(w) = f(w_t) + \frac{1}{2}(w - w_t)^{\mathrm{T}} H (w - w_t)$. Using similar arguments, and the fact that $R = \log d/\eta\sqrt{\rho\varepsilon}$, we have: $f(w_{t+R}) \le f(w_t) - \eta^2/4$. Hence, once entrapped near SOSP, the algorithm does not leave the neighborhood.

The theorem follows by combining the above three observations.

Note that the above proof requires $O(\mathrm{poly}(d))$ iterations, while for convex functions, convergence is independent of d and only depends on L. Jin et al.[17] resolved this gap, by showing that a variant of the above algorithm converges in $O()$ iterations. The main weakness of the above analysis is that it only analyzes a quadratic approximation of the problem and then uses Hessian continuity arguments to show that the algorithm converges to SOSP. Jin et al.[17] directly analyzes how the iterates escape saddle point, leading to a tighter result albeit with more involved arguments.

Note that while cubic regularization converges in $O(1/\varepsilon^{1.5})$ iterations, noisy gradient descent requires $O(1/\varepsilon^2)$ iterations. A natural question is can be use a noisy version of the accelerated gradient descent[19] to similarly escape the saddle points but with lesser number of iterations. Jin et al.[18] show that it is possible but there is still a gap w.r.t. cubic regularization. That is, it shows that a noisy accelerated gradient descent method converges to ε-approximate SOSP in $O(1/\varepsilon^{1.75})$ iterations. Devising the first-order methods that can achieve same iteration complexity as the cubic regularization technique for finding SOSP or a proof of lower bound for first-order methods remains an important open question.

3.2 Noisy Stochastic Gradient Descent for SOSP

Algorithm 6 Noisy Stochastic Gradient Descent (NSGD)

Input: Oracle access to $f(w) = \frac{1}{n}\sum_i f_i(w)$, $\nabla_w f_i(w)$. Initial point: w_0. Step-size: η. Maximum no. of iterations: T.

1: $\tau \sim Unif[1,T]$
2: **for** $t = 1,2,\ldots,\tau$ **do**
3: $\quad i_t \sim Unif[1,n]$
4: $\quad w_{t+1} = w_t - \eta(\nabla_w f_{i_t}(w_t) + \eta\zeta_t),\ \zeta_t \sim \mathcal{N}(0,I)$
5: **end for**
6: **return** w_τ

Similar to FOSP, it is critical for several ML problems to design SOSP algorithms that can work with the finite sum setting, i.e., when $f(w) = \frac{1}{n}\sum_{i=1}^n f_i(w)$. For this setting, we can use gradient of a randomly sampled f_i as a proxy for the gradient of f, which leads to Algorithm 6. The convergence rate for the algorithm is similar to the one given in Theorem 5. Similar to FOSP, the convergence rate for Algorithm 6 also depends on the variance term $\max_w \mathbb{E}[\|\nabla_w f_i(w)\|^2] - \nabla_w f(w)^2$. Several recent results address this concern using variance reduction techniques [].

Table 3: SOSP: no. of first-order oracle calls (gradient computations) required by different methods for converging to ε-SOSP (Definition 3) of f when f is a non-convex and a convex function, respectively.

Algorithm	Iteration complexity (non-convex)	Oracle type
NGD (Algorithm 5)	$O\left(\frac{1}{\varepsilon^2}\right)$	Gradient
NAGD[18]	$O\left(\frac{1}{\varepsilon^{1.75}}\right)$	Gradient
Cubic regularization (2)	$O\left(\frac{1}{\varepsilon^{1.5}}\right)$	Hessian
Hessian-free cubic[1, 7]	$O\left(\frac{1}{\varepsilon^{1.75}}\right)$	Hessian-vector product

Assumption: all problem parameters such as L, $f(w_0) - f(w_*)$ are constant

Table 4: SOSP in finite-sum setting ($f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$: no. of first order oracle calls (individual gradient computations $\nabla f_i(w)$) required by different methods for converging to ε-SOSP (Definition 3) of f when f is a non-convex and a convex function, respectively.

Algorithm	No. of gradient calls (non-convex)	No. of gradient calls (convex)
Noisy GD (Algorithm 5)	$O\left(\frac{n}{\varepsilon^2}\right)$	$O\left(\frac{n}{\varepsilon}\right)$
Noisy AGD[18]	$O\left(\frac{n}{\varepsilon^{1.75}}\right)$	$O\left(\frac{n}{\sqrt{\varepsilon}}\right)$
Noisy SGD[10, 17]	$O\left(\frac{1}{\varepsilon^4}\right)$	$O\left(\frac{1}{\varepsilon^2}\right)$
Noisy SVRG[2]	$O\left(n + \frac{n^{3/4}}{\varepsilon^2}\right)$	$O\left(n + \frac{\sqrt{n}}{\varepsilon^2}\right)$

Assumption: all problem parameters such as L, $f(w_0) - f(w_*)$ are constant

3.3 Summary

Finding SOSP is critical in several domains. Even in deep learning, it is widely believed that second-order saddle points are reasonable solutions but first-order saddle points are not desirable. Several recent results have addressed the problem, we summarize a few results as follows: *Open problems* Note that all the above results require the Hessian to be Lipschitz continuous which is a stricter condition than the smoothness condition required by standard convex optimization problems. A natural open question is to design and analyze algorithms for convergence to SOSP that do not require such assumption on Hessian. Another important open question is to bridge the gap between iteration complexity of the cubic regularization and the first-order methods discussed above. Finally, further exploiting structure in f, like non-convex + smooth/non-convex + strongly-convex, to provide faster algorithms is another interesting research direction (Tables 2, 3, 4).

4 Discussion

Non-convex optimization is critical to several ML problems with applications in deep learning, recommendation systems (matrix completion), dimensionality reduction (PCA, sparse-PCA), robust learning, etc. Most of these problems are NP hard in general, and come up with certain structure that helps make them more tractable in practice.

In this survey, we focus on a canonical unconstrained non-convex function optimization formulation. However, several other formulations are equally critical in ML domain. For example, optimizing non-convex functions over convex sets. Another form is optimizing convex function over non-convex sets which represents several critical problems such as PCA, sparse regression, tensor decomposition. We refer readers to[15] for a survey of some of these techniques.

For unconstrained non-convex optimization, we discussed a few scalable techniques for FOSP as well as for SOSP. While SOSPs are desirable in practice, especially for deep learning techniques, most of the existing practical algorithms use variants of SGD (Algorithm 3) with momentum

term for acceleration. While theoretically we can only guarantee convergence to FOSP for such algorithms, in practice this simple algorithm converges to a SOSP[26]. One possible reason for the same might be the fact that SGD itself adds significant amount of noise thus avoiding saddle points but currently existing SGD analysis are unable to provide a general result without some assumptions on noise added by SGD in each iteration[8], and remains an active area of research. In addition, studying and possibly bridging the gap between time complexity for finding FOSP for convex functions and non-convex functions is an important problem.

Several recent results have shown that several non-convex optimizations do not admit spurious local minima or SOSP; hence, convergence to SOSP is enough to guarantee global optimality[6,11]. Past few years have seen a flurry of activity in the SOSP domain, with several new and scalable techniques. However, still a few fundamental questions remain open. For example, can we obtain solutions as efficient as cubic regularization (in terms of iteration complexity) using the first-order methods? Does randomly initialized gradient descent method already converge to SOSP without any additional noise? Similarly, does SGD's iterations add enough randomness to escape first-order saddle points efficiently?

Finally, statistics play a critical role in non-convex optimization problems in ML, which represent exciting new opportunities. For example, standard matrix completion problem is an NP-hard problem but by imposing statistical restrictions on the generated data, we can reduce the problem to an SOSP problem which can be solved in polynomial time (in $1/\varepsilon$)[11]. Similarly, sparse linear regression/compressed sensing problems admit optimal solution to L_0 constrained non-convex optimization problems due to statistical nature of the observations[16]. It is widely believed that techniques that are aware of both the statistical as well as optimization aspects of ML problems can produce strong results.

Publisher's Note

Received: 7 January 2019 Accepted: 26 April 2019
Published online: 6 June 2019

References

1. Agarwal N, Allen-Zhu Z, Bullins B, Hazan E, Ma T (2017) Finding approximate local minima faster than gradient descent. In: Proceedings of the 49th annual ACM SIGACT symposium on theory of computing (STOC)
2. Allen-Zhu Z (2018) Natasha 2: faster non-convex optimization than SGD. In: Advances in neural information processing systems 31: annual conference on neural information processing systems 2018. NeurIPS 2018, 3–8 December 2018, Montréal, Canada, pp 2680–2691. http://papers.nips.cc/paper/7533-natasha-2-faster-non-convex-optimization-than-sgd
3. Allen-Zhu Z, Hazan E (2016) Variance reduction for faster non-convex optimization. In: International conference on machine learning, pp 699–707
4. Anandkumar A, Ge R (2016) Efficient approaches for escaping higher order saddle points in non-convex optimization. In: Proceedings of the 29th conference on learning theory (COLT), pp 81–102
5. Bertsekas DP (2016) Nonlinear programming, 3rd edn. Athena Scientific, Belmont
6. Bhojanapalli S, Boumal N, Jain P, Netrapalli P (2018) Smoothed analysis for low-rank solutions to semidefinite programs in quadratic penalty form. In: Conference on learning theory, COLT 2018, Stockholm, Sweden, 6–9 July 2018, pp 3243–3270. http://proceedings.mlr.press/v75/bhojanapalli18a.html
7. Carmon Y, Duchi JC (2016) Gradient descent efficiently finds the cubic-regularized non-convex Newton step. CoRR. arXiv:1612.00547
8. Daneshmand H, Kohler J, Lucchi A, Hofmann T (2018) Escaping saddles with stochastic gradients. arXiv preprint. arXiv:1803.05999
9. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) ImageNet: a large-scale hierarchical image database. In: CVPR09
10. Ge R, Huang F, Jin C, Yuan Y (2015) Escaping from saddle points—online stochastic gradient for tensor decomposition. In: Proceedings of the 28th conference on learning theory (COLT), pp 797–842
11. Ge R, Jin C, Zheng Y (2017) No spurious local minima in nonconvex low rank problems: a unified geometric analysis. In: Proceedings of the 34th international conference on machine learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017, pp 1233–1242. http://proceedings.mlr.press/v70/ge17a.html
12. Ghadimi S, Lan G (2013) Stochastic first- and zeroth-order methods for nonconvex stochastic programming. SIAM J Optim 23(4):2341–2368
13. Ghadimi S, Lan G (2016) Accelerated gradient methods for nonconvex nonlinear and stochastic programming. Math Program 156(1–2):59–99
14. Ghadimi S, Lan G, Zhang H (2016) Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. Math Program 155(1–2):267–305

15. Jain P, Kar P et al (2017) Non-convex optimization for machine learning. Found Trends® Mach Learn 10(3–4):142–336
16. Jain P, Tewari A, Kar P (2014) On iterative hard thresholding methods for high-dimensional M-estimation. In: Proceedings of the 28th annual conference on neural information processing systems (NIPS)
17. Jin C, Ge R, Netrapalli P, Kakade SM, Jordan MI (2017) How to escape saddle points efficiently. In: Proceedings of the 34th international conference on machine learning (ICML), pp 1724–1732
18. Jin C, Netrapalli P, Jordan MI (2017) Accelerated gradient descent escapes saddle points faster than gradient descent. CoRR. arXiv:1711.10456
19. Nesterov Y (2003) Introductory lectures on convex optimization: a basic course. Kluwer-Academic, Dordrecht
20. Nesterov Y, Polyak B (2006) Cubic regularization of Newton method and its global performance. Math Program 108(1):177–205
21. Nesterov YE (1983) A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. Doklady AN SSSR 269
22. Netrapalli P (2019) Stochastic gradient descent and it's variants in machine learning
23. Polyak BT (1964) Some methods of speeding up the convergence of iteration methods. USSR Comput Math Math Phys 4
24. Reddi SJ, Sra S, Poczos B, Smola AJ (2016) Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. In: Lee DD, Sugiyama M, Luxburg UV, Guyon I, Garnett R (eds) Advances in neural information processing systems, vol 29. Curran Associates, Inc., New York, pp 1145–1153. http://papers.nips.cc/paper/6116-proximal-stochastic-methods-for-nonsmooth-nonconvex-finite-sum-optimization.pdf
25. Reddi SJ, Hefny A, Sra S, Poczos B, Smola A (2016) Stochastic variance reduction for nonconvex optimization. In: International conference on machine learning, pp 314–323
26. Sagun L, Evci U, Guney VU, Dauphin Y, Bottou L (2017) Empirical analysis of the Hessian of over-parametrized neural networks. arXiv preprint. arXiv:1706.04454
27. Tripuraneni N, Stern M, Jin C, Regier J, Jordan MI (2018) Stochastic cubic regularization for fast nonconvex optimization. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R (eds) Advances in neural information processing systems, vol 31. Curran Associates, Inc., New York, pp 2904–2913. http://papers.nips.cc/paper/7554-stochastic-cubic-regularization-for-fast-nonconvex-optimization.pdf

Prateek Jain is a member of the Machine Learning and Optimization and the Algorithms and Data Sciences Group at Microsoft Research, Bangalore, India. He is also an Adjunct Faculty in the Department of CSE, IIT Kanpur. His research interests are in machine learning, non-convex optimization, high-dimensional statistics, and optimization algorithms in general and their applications to privacy, computer vision, text mining and natural language processing. He holds a PhD at the University of Texas at Austin and was advised by Prof. Inderjit S. Dhillon.