J. Indian Inst. Sci.
A Multidisciplinary Reviews Journal
ISSN: 0970-4140 Coden-JIISAD
© Indian Institute of Science 2019.

REVIEW ARTICLE

Stochastic Gradient Descent and Its Variants in Machine Learning

*Praneeth Netrapalli**

Abstract | Stochastic gradient descent (SGD) is a fundamental algorithm which has had a profound impact on machine learning. This article surveys some important results on SGD and its variants that arose in machine learning.

Keywords: *Stochastic optimization, Gradient descent, Large scale optimization*

1 Introduction

Optimization problems are ubiquitous and arise in almost all branches of Science and engineering. Given a function $f : \mathbb{R}^d \to \mathbb{R}$, how do we find

$$\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}). \qquad (1)$$

One of the most classical methods to solve this problem is gradient descent (GD), which was originally proposed by Cauchy[23]. In order to use this algorithm, one has to be able to compute the gradient of $f(\cdot)$, $\nabla f(\mathbf{w})$ at any given point $\mathbf{w}$. GD starts with an initial vector $\mathbf{w_0}$ and performs the following updates:

$$\mathbf{w_{k+1}} = \mathbf{w_k} - \alpha_k \nabla f(\mathbf{w_k}), \quad \text{for } k = 0, \ldots,$$

where α_k are step sizes. GD has played a fundamental role in the field of optimization.

In several settings, however, one might not have access to the *exact* gradients of $f(\cdot)$. Consider for example, the case where one does not have a closed form expression of $f(\cdot)$ but rather has to perform experiments/measurements to compute properties (such as gradient at a point) of $f(\cdot)$. These experiments and measurements have some amount of error and randomness in their outputs so that one might have to perform a large number of independent measurements and average them if one wishes to obtain exact gradients. Is it possible to avoid this and make do with only one measurement per point? Robbins and Monro wrote a seminal paper[59] motivated by this question. Robbins and Monro[59] showed that essentially the same update as that of GD, appropriately modified, works in this setting. More concretely, suppose for every k, we have access to a random vector $\zeta_\mathbf{k}$ which in expectation is $\nabla f(\mathbf{w_k})$, i.e., $\mathbb{E}[\zeta_\mathbf{k}] = \nabla f(\mathbf{w_k})$. Then Kiefer and Wolfowitz[41] and Robbins and Monro[59] show that the updates

$$\mathbf{w_{k+1}} = \mathbf{w_k} - \alpha_k \zeta_\mathbf{k}, \qquad (2)$$

where the sequence α_k chosen appropriately will asymptotically converge to the solution of $\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w})$ under some conditions on $f(\cdot)$. The setting is known as stochastic approximation and (2) is known as stochastic gradient descent (SGD).

When one talks about the performance of SGD, one wants to understand how the sequence $\mathbf{w_k}$ behaves. Qualitatively, one could ask for instance, does $\mathbf{w_k}$ converge to a global minimum of (1)? Does it converge to a local minimum? Does it converge at all? The answers to these questions depend on properties of the function $f(\cdot)$. One could ask quantitatively as to how fast does it converge to wherever it converges. Let us denote by $\mathrm{Err}(\mathbf{w_k})$ some notion of error in the iterate $\mathbf{w_k}$. For example, this could be function suboptimality $f(\mathbf{w_k}) - \min_\mathbf{w} f(\mathbf{w})$ or distance from a local minimum, etc. which usually goes to 0 as $k \to \infty$. The question of quantitative convergence results can be decomposed into two parts:

1. *Asymptotic* How does $\mathrm{Err}(\mathbf{w_k})$ behave as $k \to \infty$? Does it behave like $\mathcal{O}\left(\frac{1}{k}\right)$ or $\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$?
2. *Nonasymptotic* For any given k, can I obtain a bound on $\mathrm{Err}(\mathbf{w_k})$? This would include, for instance, obtaining the right problem-dependent constants in the asymptotic bounds above.

This article belongs to the Special issue—Recent Advances in Machine Learning.

Microsoft Research, #9, Vigyan 1st Floor, Lavelle Road, Bengaluru, Karnataka 560001, India.
**praneeth@microsoft.com*

Springer IISc Press

Since[41, 59], stochastic approximation has been a very active area of research and has made significant progress in answering both qualitative and quantitative questions above in a variety of settings. Works that build upon Kiefer and Wolfowitz[41] and Robbins and Monro[59] have significantly enlarged the classes of functions for which one could show asymptotic convergence of (2), stepsizes that obtain improved asymptotic convergence rates, and in some cases bounds on the nonasymptotic convergence rates. The literature is too vast to cite in this short survey. Interested readers may consult the monographs[15, 18, 42, 43] for a detailed overview.

Not long after the setting of stochastic approximation and SGD were introduced, they found applications in several fields including signal processing and machine learning[11, 65]. While SGD was used in some specific machine learning settings since the 80s e.g., LeCu[45, 46] and Yann[68], it was not until the work of Bottou and Bousquet[19] that its importance to machine learning as a whole was realized. The main point of Bottou and Bousquet[19] was that in several machine learning problems, where we might be able to compute exact gradients, and it is possible to implement GD, SGD might still offer significant computational benefits over GD. Contrast this with the motivation of Robbins and Monro[59] which was purely the inability to compute exact gradients in certain settings. It is a hallmark of profound ideas that they outgrow the confines of the initial settings in which they were thought of.

In order to understand the computational benefits offered by SGD over GD, and apply it to machine learning problems, one needs to thoroughly understand the convergence behavior of these algorithms. Moreover, as the computational budget is limited, understanding asymptotic convergence rates is not sufficient; understanding nonasymptotic rates is of paramount importance. By now, there is a vast amount of literature answering these questions in various settings. There are also improved algorithms which take advantage of the specifics of the machine learning setting.

2 Setting and Outline of the Paper

This survey is concerned with a (small) selection of such convergence results of SGD (2) and its variants, under some widely studied settings of the function $f(\cdot)$ and stochastic gradient $\zeta_{\mathbf{k}}$. These results form the basis of the application of SGD in various fields, including machine learning. Different sections of this paper deal with different settings.

Outline Some preliminaries are presented in Sect. 3. Section 4 provides a brief primer on how SGD is used in the machine learning context. Section 5 presents results for the case when $f(\cdot)$ is convex, Sect. 6 for the case when $f(\cdot)$ solves the principal component analysis (PCA) problem and Sect. 7 for the case when $f(\cdot)$ is a nonconvex function. Within each of these sections there are subsections presenting improved results for important special cases. Section 8 summarizes these results and presents several important open directions for future work. Finally Sect. 9 presents some resources for using these algorithms in practice.

Due to the vast literature on this topic, and the size of this survey, it is unavoidable that we will not talk about several important works. Interested readers may refer more comprehensive surveys such as[20].

Notation Normal font a, b, etc. is used to denote scalars. Bold font, small case letters $\mathbf{a}, \mathbf{b}$, etc. are used to denote vectors and bold font, upper case letters $\mathbf{A}, \mathbf{B}$ are used to denote matrices. $\|\mathbf{a}\|$ denotes the ℓ_2 norm of $\mathbf{a}$ and $\|\mathbf{A}\|$ denotes the operator norm of $\mathbf{A}$. $\mathcal{O}(\cdot)$ and $\Omega(\cdot)$ denote the standard big-Oh and Omega notation, respectively. $\widetilde{\mathcal{O}}(\cdot)$ is the same as $\mathcal{O}(\cdot)$ up to a multiplicative factor of polylog$(\cdot)$ in all the relevant parameters. $[n]$ denotes the set $\{1, \cdots, n\}$.

3 Preliminaries

In this section, we will give some preliminaries that will be helpful in following the rest of this article. An important notion that will arise repeatedly in the sections to follow is the *gradient* of a function. Given a function $f : \mathbb{R}^d \to \mathbb{R}$, and a point $\mathbf{w} \in \mathbb{R}^d$, the gradient of $f(\cdot)$ at $\mathbf{w}$ denoted by $\nabla f(\mathbf{w}) \in \mathbb{R}^d$ is the quantity that satisfies

$$\lim_{\epsilon \to 0} \frac{f(\mathbf{w} + \epsilon \mathbf{v}) - f(\mathbf{w}) - \epsilon \cdot \langle \nabla f(\mathbf{w}), \mathbf{v} \rangle}{\epsilon} = 0 \quad \forall \quad \mathbf{v} \in \mathbb{R}^d.$$

If $\nabla f(\mathbf{w})$ exists (satisfying the property above), then $f(\cdot)$ is said to be differentiable at $\mathbf{w}$. A function $f(\cdot)$ is said to be differentiable if it is differentiable everywhere in $\mathbb{R}^d$. While we assume the function to be differentiable throughout this article, results in Sects. 5.1 and 5.2 hold even without this property. We will also use the following useful facts in some of our proofs. The proofs of these statements are left as exercises.

Fact 1 *For any natural numbers $n_1 < n_2$, we have*

1. $\log \frac{n_2+1}{n_1} \leq \sum_{i=n_1}^{n_2} \frac{1}{i} \leq 1 + \log \frac{n_2+1}{n_1}$
2. $\frac{1}{2}\left(\frac{1}{n_1} - \frac{1}{n_2+1}\right) \leq \sum_{i=n_1}^{n_2} \frac{1}{i^2} \leq \frac{1}{2}\left(\frac{1}{n_1-1} - \frac{1}{n_2}\right)$
3. $2\left(\sqrt{n_2+1} - \sqrt{n_1}\right) \leq \sum_{i=n_1}^{n_2} \frac{1}{\sqrt{i}} \leq 2\left(\sqrt{n_2} - \sqrt{n_1 - 1}\right)$

4 How is SGD Used in Machine Learning?

A vast majority of machine learning problems can be posed as minimizing a *loss* (think of it as cost) function which can be written as an expectation over some underlying distribution $\mathbf{x} \sim \mathcal{D}$

$$\min_{\mathbf{w}\in\mathbb{R}^d} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\phi(\mathbf{x},\mathbf{w})]. \tag{3}$$

For example, $\mathbf{x}$ may correspond to an image and $\mathcal{D}$ to the distribution of natural images. $\phi(\mathbf{x},\mathbf{w})$ might indicate how well $\mathbf{w}$ explains $\mathbf{x}$. The way we have access to the expectation is via some samples (also called *examples*) $\mathbf{x_1},\ldots,\mathbf{x_n}$. For example, each $\mathbf{x_i}$ may correspond to a particular image. Since the only access we have to the distribution $\mathcal{D}$ is via $\mathbf{x_1},\ldots,\mathbf{x_n}$, a natural way to solve (3) is by solving the following:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \phi(\mathbf{x_i},\mathbf{w}). \tag{4}$$

The above minimization problem is known as *empirical risk minimization (ERM)* or sample average approximation (SAA) since the minimization is over samples $\mathbf{x_1},\ldots,\mathbf{x_n}$. Defining the function $f(\mathbf{w}) \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n}\phi(\mathbf{x_i},\mathbf{w})$, one can obtain stochastic gradient for any step by choosing i uniformly at random from $1,\ldots,n$ and using $\zeta_\mathbf{k} \overset{\text{def}}{=} \nabla\phi(\mathbf{x_i},\mathbf{w})$. Note that it is indeed possible to compute the full gradient of the function in (4). However, one needs to go over all the examples, which requires $\Omega(n)$ time. The hope is that computing the stochastic gradient over one random example, which requires much less time, might still be enough to make progress on (4). To summarize, the appeal of SGD in machine learning applications is predominantly that of computational efficiency. In optimization language, problems of the form (4) are called finite sum problems.

Since most machine learning problems can be written in the form of (4), we are mostly interested in understanding convergence rates of finite sum problems. However, it turns out that there is lot of intuition to be gained from understanding the behavior of SGD on simpler problems. For example, one could consider the setting where we wish to $\min_\mathbf{w} f(\mathbf{w})$ and we have $\zeta_\mathbf{k} = \nabla f(\mathbf{w_k}) + \mathbf{g_k}$, where $\mathbf{g_k}$ is a standard multivariate normal random vector. Even these settings are interesting and results for such settings will also be presented in the sections to follow.

5 Convex Optimization

In this section, we will present results on the performance of stochastic gradient methods for convex optimization.

Assumption 1 A differentiable function $f(\cdot)$ is said to be convex if

$$f(\mathbf{v}) \geq f(\mathbf{w}) + \langle\nabla f(\mathbf{w}),\mathbf{v}-\mathbf{w}\rangle, \qquad \forall\, \mathbf{w},\mathbf{v}.$$

Convex optimization has played a significant role in the development of machine learning. Some well-known and representative applications of convex optimization are support vector machines (SVMs)[32], sparse linear regression using Lasso[64] and matrix completion using semidefinite relaxation[22]. Let us briefly describe linear regression as a representative example.

Linear regression We are given points $(\mathbf{x_1},y_1),\ldots,(\mathbf{x_n},y_n)$. We wish to find a vector $\mathbf{w}$ such that $\mathbf{w}^\top\mathbf{x_i} \approx y_i$ for every $i \in [n]$. A classical way to do this is linear regression:

$$\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{w}^\top\mathbf{x_i} - y_i\right)^2.$$

The function in the minimization above is convex. We will return to the convex finite sum problems in Sect. 5.3, but we will first present results in the simpler settings where the stochastic gradients have bounded variance.

5.1 Convex Lipschitz Functions

The simplest setting for analyzing (2) is when the function $f(\cdot)$ is Lipschitz (i.e., $\left\|\nabla f(\mathbf{w_k})\right\|$ is uniformly bounded) and the stochastic gradients have bounded variance (these two conditions are also equivalent to $\mathbb{E}\left[\|\zeta_\mathbf{k}\|^2\right]$ being bounded). The following theorem is a classical result[51] on the convergence of SGD in such settings:

Theorem 1 *Suppose $f(\cdot)$ satisfies Assumption 1. If we run the SGD algorithm (2), where the stochastic gradients $\zeta_\mathbf{k}$ are all independent and satisfy $\mathbb{E}[\zeta_\mathbf{k}] = \nabla f(\mathbf{w_k})$ and $\mathbb{E}\left[\|\zeta_\mathbf{k}\|^2\right] \leq \sigma^2$, and the stepsizes α_k are chosen to be $\frac{\|\mathbf{w_1}-\mathbf{w}^*\|\sigma}{\sqrt{k}}$. Then, we have*

$$\left(\frac{1}{\sum_{k=1}^{t}\alpha_k}\sum_{k=1}^{t}\alpha_k\mathbb{E}[f(\mathbf{w_k})]\right) - f(\mathbf{w}^*) \leq \frac{\|\mathbf{w_1}-\mathbf{w}^*\|\sigma\log t}{\sqrt{t}}.$$

Proof The main idea of the proof is to track the distance between $\mathbf{w_k}$ and $\mathbf{w}^*$:

$$\left\|\mathbf{w_{k+1}}-\mathbf{w}^*\right\|^2 = \left\|\mathbf{w_k}-\mathbf{w}^*\right\|^2 - 2\alpha_k\langle\zeta_\mathbf{k},\mathbf{w_k}-\mathbf{w}^*\rangle + \alpha_k^2\|\zeta_\mathbf{k}\|^2.$$

Taking expectations on both sides, we have

$$\mathbb{E}\left[\left\|\mathbf{w_{k+1}}-\mathbf{w}^*\right\|^2\right] \leq \mathbb{E}\left[\left\|\mathbf{w_k}-\mathbf{w}^*\right\|^2\right] - 2\alpha_k\mathbb{E}\left[\langle\nabla f(\mathbf{w_k}),\mathbf{w_k}-\mathbf{w}^*\rangle\right] + \alpha_k^2\sigma^2.$$

By Assumption 1, we know that $f(\mathbf{w}^*) \geq f(\mathbf{w_k}) + \langle\nabla f(\mathbf{w_k}),\mathbf{w}^*-\mathbf{w_k}\rangle$. Plugging this in the above inequality and reorganizing gives us

$$2\alpha_k\mathbb{E}[f(\mathbf{w_k})] - f(\mathbf{w}^*) \leq \left(\mathbb{E}\left[\left\|\mathbf{w_k}-\mathbf{w}^*\right\|^2\right] - \mathbb{E}\left[\left\|\mathbf{w_{k+1}}-\mathbf{w}^*\right\|^2\right]\right) + \alpha_k^2\sigma^2.$$

Adding the above expression over k, dividing by $2\sum_{k=1}^{t}\alpha_k$ and substituting the value of α_k gives us the result. □

The above theorem tells us that, for convex Lipschitz functions and stochastic gradients with bounded variance, SGD converges to the optimal solution at a rate of $\mathcal{O}\left(\frac{\log t}{\sqrt{t}}\right)$. This almost matches (up to log factors) the best rate possible for convex Lipschitz functions even with *exact* gradients[51]. This means that stochasticity in the gradients does not deteriorate the convergence rate for this setting.

5.2 Strongly Convex Lipschitz Functions

While $\mathcal{O}\left(\frac{1}{\sqrt{t}}\right)$ convergence rate is the best possible for convex Lipschitz functions, it turns out that SGD has much better convergence rate under the additional assumption of strong convexity.

Assumption 2 A differentiable function $f(\cdot)$ is μ-strongly convex if

$$f(\mathbf{v}) \geq f(\mathbf{w}) + \langle\nabla f(\mathbf{w}),\mathbf{v}-\mathbf{w}\rangle + \frac{\mu}{2}\|\mathbf{v}-\mathbf{w}\|^2, \qquad \forall\ \mathbf{w},\mathbf{v}.$$

Strong convexity is satisfied, for example, by *regularized* problems where $f(\mathbf{w}) = g(\mathbf{w}) + \frac{\mu}{2}\|\mathbf{w}\|^2$, with $g(\cdot)$ itself a convex function. The additional $\frac{\mu}{2}\|\mathbf{w}\|^2$ is included for stability reasons (in other words, to decrease overfitting to the training data). The convergence rate of Theorem 1 can be improved under Assumption 2[51].

Theorem 2 *Suppose $f(\cdot)$ satisfies Assumption 2. If we run the SGD algorithm (2), where the stochastic gradients $\zeta_\mathbf{k}$ are all independent and satisfy $\mathbb{E}[\zeta_\mathbf{k}] = \nabla f(\mathbf{w_k})$ and $\mathbb{E}\left[\|\zeta_\mathbf{k}\|^2\right] \leq \sigma^2$, and the stepsizes $\alpha_k = \frac{1}{\mu k}$. Then,*

$$\left(\frac{1}{t}\sum_{k=1}^{t}\mathbb{E}[f(\mathbf{w_k})]\right) - f(\mathbf{w}^*) \leq \frac{\sigma^2\log t}{2\mu t}.$$

Proof The proof begins in the same way as in that of Theorem 1.

$$\mathbb{E}\left[\left\|\mathbf{w_{k+1}}-\mathbf{w}^*\right\|^2\right] \leq \mathbb{E}\left[\left\|\mathbf{w_k}-\mathbf{w}^*\right\|^2\right] - 2\alpha_k\mathbb{E}\left[\langle\nabla f(\mathbf{w_k}),\mathbf{w_k}-\mathbf{w}^*\rangle\right] + \alpha_k^2\sigma^2.$$

We now use the strong convex property to conclude that $\langle\nabla f(\mathbf{w_k}),\mathbf{w_k}-\mathbf{w}^*\rangle \geq f(\mathbf{w_k}) - f(\mathbf{w}^*) + \frac{\mu}{2}\|\mathbf{w_k}-\mathbf{w}^*\|^2$. Plugging this in the above inequality gives us

$$\mathbb{E}\left[\left\|\mathbf{w_{k+1}}-\mathbf{w}^*\right\|^2\right] \leq (1-\alpha_k\mu)\mathbb{E}\left[\left\|\mathbf{w_k}-\mathbf{w}^*\right\|^2\right] - 2\alpha_k\left(\mathbb{E}[f(\mathbf{w_k})] - f(\mathbf{w}^*)\right) + \alpha_k^2\sigma^2.$$

Reorganizing the above, we have

$$\mathbb{E}[f(\mathbf{w_k})] - f(\mathbf{w}^*) \leq \frac{\mu}{2}\left((k-1)\mathbb{E}\left[\left\|\mathbf{w_k}-\mathbf{w}^*\right\|^2\right] - k\mathbb{E}\left[\left\|\mathbf{w_{k+1}}-\mathbf{w}^*\right\|^2\right]\right) + \frac{\sigma^2}{2k\mu}.$$

Summing up over k and dividing by t proves the theorem. □

Note that Theorem 2 gives a convergence rate of $\mathcal{O}\left(\frac{\log t}{t}\right)$ as compared to $\mathcal{O}\left(\frac{\log t}{\sqrt{t}}\right)$ of Theorem 1. This matches the best known convergence rate for strongly convex Lipschitz functions even with an *exact* gradient oracle.

5.3 Finite Sums

Up till now, we have considered settings where we did not have the option of querying the exact gradient. We now consider a setting where we do have the option of computing the exact gradient but it is very expensive. The setting is that of finite sums already mentioned in Sect. 4. For notational simplicity, we rewrite it as follows:

$$\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n}\phi_i(\mathbf{w}), \tag{5}$$

where we used $\phi_i(\mathbf{w})$ instead of $\phi(\mathbf{x_i},\mathbf{w})$. Recall that in these problems, it is indeed possible to compute the exact gradient $\nabla f(\mathbf{w})$ but it requires going overall the n data points. On the other hand, it is much easier to obtain a stochastic gradient: sample a number i independently from 1 to n and return $\nabla\phi_i(\mathbf{w})$. The advantages and disadvantages of GD and SGD using exact and stochastic gradients, respectively, is summarized in Table 1.

The key question that arises after looking at Table 1 is whether it is possible to combine the fast iterations of SGD with the fast convergence rate of GD. It turns out this is possible under the additional assumption of smoothness on each component function $\phi_i(\cdot)$.

Assumption 3 A differentiable function $f(\cdot)$ is ℓ-smooth if

$$f(\mathbf{v}) \le f(\mathbf{w}) + \langle\nabla f(\mathbf{w}),\mathbf{v}-\mathbf{w}\rangle + \frac{\ell}{2}\|\mathbf{v}-\mathbf{w}\|^2, \quad \forall\ \mathbf{w},\mathbf{v}.$$

The main idea in obtaining improved rates for this setting is that one can reduce the variance in stochastic gradients by computing the *exact* gradient once in a while. The resulting algorithm is known as stochastic variance reduced gradient (SVRG) and was first proposed by Johnson and Zhang[38]. The SVRG algorithm is presented in Algorithm 1. The following theorem provides performance guarantees for SVRG:

Table 1: Comparison of GD and SGD.

Method	Per iteration	Convergence rate
GD	Slow	Fast
SGD	Fast	Slow

Theorem 3 *Consider the finite sum setting (5). Suppose that*

- *each $\phi_i(\cdot)$ is convex (Assumption 1) and ℓ-smooth (Assumption 3) and*
- *the total function $f(\cdot)$ is μ strongly convex (Assumption 2).*

Then, SVRG (Algorithm 1) with learning rate $\alpha = \frac{1}{8\ell}$ and condition number $\kappa = \frac{\ell}{\mu}$ satisfies $\mathbb{E}\left[f\left(\widetilde{\mathbf{w}}_{t+1}\right)\right] - f(\mathbf{w}^*) \le \left(\frac{3}{4}\right)^t\left(f(\mathbf{w}_{\text{init}}) - f(\mathbf{w}^*)\right)$.

In contrast to Theorems 1 and 2 which guarantee convergence rates of $\widetilde{\mathcal{O}}\left(\frac{1}{\sqrt{t}}\right)$ and $\widetilde{\mathcal{O}}\left(\frac{1}{t}\right)$, respectively, this theorem guarantees a much faster convergence rate of $2^{-\Omega(t)}$. We also note that SVRG was not the first algorithm to achieve these improved rates. Stochastic average gradient (SAG)[61] and stochastic dual coordinate ascent (SDCA)[63] also achieve similar rates in this setting. However, the concept of variance reduction was made explicit in SVRG and has since been extended to several other settings, one of which we will mention in Sect. 7.

Proof (Proof of Theorem 3) We will establish the inequality

Algorithm 1 Stochastic Variance Reduced Gradient (SVRG)

```
Input: functions $\phi_1(\cdot),\cdots,\phi_n(\cdot)$, $\mathbf{w}_{\text{init}}$, learning rate $\alpha$, condition number $\kappa$, time steps
  $t$.
  $\widetilde{\mathbf{w}}_\mathbf{1} \leftarrow \mathbf{w}_{\text{init}}$
  for $s \leftarrow 1,\cdots,t$ do
    $\mathbf{w}^\mathbf{s}_\mathbf{1} \leftarrow \widetilde{\mathbf{w}}_\mathbf{s}$
    $\mathbf{g} \leftarrow \nabla f(\widetilde{\mathbf{w}}_\mathbf{s})$
    for $r \leftarrow 1,\cdots,22\kappa$ do
      $i \leftarrow$ Uniformly at random from $1,\cdots,n$
      $\mathbf{w}^\mathbf{s}_{\mathbf{r+1}} \leftarrow \mathbf{w}^\mathbf{s}_\mathbf{r} - \alpha\left(\nabla\phi_i\left(\mathbf{w}^\mathbf{s}_\mathbf{r}\right) - \nabla\phi_i\left(\widetilde{\mathbf{w}}_\mathbf{s}\right) + \mathbf{g}\right)$
    $\widetilde{\mathbf{w}}_{\mathbf{s+1}} \leftarrow \frac{1}{22\kappa}\sum_{r=1}^{22\kappa}\mathbf{w}^\mathbf{s}_\mathbf{r}$
Output: $\widetilde{\mathbf{w}}_{\mathbf{t+1}}$
```

$$\mathbb{E}\left[f\left(\widetilde{\mathbf{w}}_{s+1}\right)\right]-f\left(\mathbf{w}^*\right) \leq \frac{3}{4}\left(\mathbb{E}\left[f\left(\widetilde{\mathbf{w}}_s\right)\right]-f\left(\mathbf{w}^*\right)\right),$$

from which the result follows. The key component of the proof lies in understanding the variance of stochastic gradient updates in

$$\mathbf{w}_{\mathbf{r}+1}^{\mathbf{s}} \leftarrow \mathbf{w}_{\mathbf{r}}^{\mathbf{s}}-\alpha\left(\nabla \phi_i\left(\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}\right)-\nabla \phi_i\left(\widetilde{\mathbf{w}}_s\right)+\mathbf{g}\right)$$

in Algorithm 1. We see that $\mathbb{E}\left[\nabla \phi_i\left(\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}\right)-\nabla \phi_i\left(\widetilde{\mathbf{w}}_s\right)+\mathbf{g}\right]=\nabla f\left(\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}\right)$ and

$$\begin{aligned}
&\mathbb{E}\left[\left\|\nabla \phi_i\left(\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}\right)-\nabla \phi_i\left(\widetilde{\mathbf{w}}_s\right)+\mathbf{g}\right\|^2\right] \\
&\quad\leq 2 \mathbb{E}\left[\left\|\nabla \phi_i\left(\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}\right)-\nabla \phi_i\left(\mathbf{w}^*\right)\right\|^2\right] \\
&\quad+2 \mathbb{E}\left[\left\|\nabla \phi_i\left(\widetilde{\mathbf{w}}_s\right)-\nabla \phi_i\left(\mathbf{w}^*\right)\right\|^2\right].
\end{aligned} \tag{6}$$

We will now bound the last expression above using smoothness and convexity of each individual function $\phi_i(\cdot)$. Given any point $\mathbf{w}$, consider the point $\overline{\mathbf{w}} \stackrel{\text{def}}{=} \mathbf{w}-\frac{1}{\ell}\left(\nabla \phi_i(\mathbf{w})-\nabla \phi_i\left(\mathbf{w}^*\right)\right)$. By smoothness of $\phi_i(\cdot)$, we have

$$\begin{aligned}
\phi_i(\overline{\mathbf{w}}) &\leq \phi_i(\mathbf{w})-\frac{1}{\ell}\left\langle\nabla \phi_i(\mathbf{w}), \nabla \phi_i(\mathbf{w})-\nabla \phi_i\left(\mathbf{w}^*\right)\right\rangle \\
&+\frac{\ell}{2 \ell^2}\left\|\nabla \phi_i(\mathbf{w})-\nabla \phi_i\left(\mathbf{w}^*\right)\right\|^2.
\end{aligned}$$

On the other hand, from convexity, we also have

$$\phi_i(\overline{\mathbf{w}}) \geq \phi_i\left(\mathbf{w}^*\right)+\left\langle\nabla \phi_i\left(\mathbf{w}^*\right), \overline{\mathbf{w}}-\mathbf{w}^*\right\rangle.$$

Combining the above two inequalities and rearranging gives us

$$\begin{aligned}
\left\|\nabla \phi_i(\mathbf{w})-\nabla \phi_i\left(\mathbf{w}^*\right)\right\|^2 &\leq 2 \ell\left(\phi_i(\mathbf{w})-\phi_i\left(\mathbf{w}^*\right)\right. \\
&\left.-\left\langle\nabla \phi_i\left(\mathbf{w}^*\right), \mathbf{w}-\mathbf{w}^*\right\rangle\right).
\end{aligned}$$

Plugging this back in (6), we obtain

$$\begin{aligned}
&\mathbb{E}\left[\left\|\nabla \phi_i\left(\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}\right)-\nabla \phi_i\left(\widetilde{\mathbf{w}}_s\right)+\mathbf{g}\right\|^2\right] \\
&\quad\leq 4 \ell\left(\mathbb{E}\left[\phi_i\left(\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}\right)\right]-\mathbb{E}\left[\phi_i\left(\mathbf{w}^*\right)\right]\right. \\
&\quad\left.-\left\langle\mathbb{E}\left[\nabla \phi_i\left(\mathbf{w}^*\right)\right], \mathbf{w}_{\mathbf{r}}^{\mathbf{s}}-\mathbf{w}^*\right\rangle\right) \\
&\quad+4 \ell\left(\mathbb{E}\left[\phi_i\left(\widetilde{\mathbf{w}}_s\right)\right]-\mathbb{E}\left[\phi_i\left(\mathbf{w}^*\right)\right]\right. \\
&\quad\left.-\left\langle\mathbb{E}\left[\nabla \phi_i\left(\mathbf{w}^*\right)\right], \widetilde{\mathbf{w}}_s-\mathbf{w}^*\right\rangle\right) \\
&\quad=4 \ell\left(\mathbb{E}\left[f\left(\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}\right)\right]-f\left(\mathbf{w}^*\right)+\mathbb{E}\left[f\left(\widetilde{\mathbf{w}}_s\right)\right]-f\left(\mathbf{w}^*\right)\right),
\end{aligned} \tag{7}$$

where we used the fact that $\mathbb{E}[\nabla \phi_i(\mathbf{w}^*)]=\nabla f(\mathbf{w}^*)=0$. The key property of the variance bound in (7) is that as $\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}$ and $\widetilde{\mathbf{w}}_s \rightarrow \mathbf{w}^*$, the variance bound goes to 0. We are now in a place to analyze the SVRG algorithm. We have

$$\begin{aligned}
&\mathbb{E}\left[\left\|\mathbf{w}_{\mathbf{r}+1}^{\mathbf{s}}-\mathbf{w}^*\right\|^2\right]=\mathbb{E}\left[\left\|\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}-\mathbf{w}^*\right\|^2\right] \\
&\quad-2 \alpha \mathbb{E}\left[\left\langle\nabla \phi_i\left(\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}\right)-\nabla \phi_i\left(\widetilde{\mathbf{w}}_s\right)+\mathbf{g}, \mathbf{w}_{\mathbf{r}}^{\mathbf{s}}-\mathbf{w}^*\right\rangle\right] \\
&\quad+\alpha^2 \mathbb{E}\left[\left\|\nabla \phi_i\left(\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}\right)-\nabla \phi_i\left(\widetilde{\mathbf{w}}_s\right)+\mathbf{g}\right\|^2\right] \\
&\quad\leq \mathbb{E}\left[\left\|\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}-\mathbf{w}^*\right\|^2\right]-2 \alpha \mathbb{E}\left[\phi_i\left(\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}\right)-\phi_i\left(\mathbf{w}^*\right)\right] \\
&\quad+4 \alpha^2 \ell\left(\mathbb{E}\left[f\left(\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}\right)\right]-f\left(\mathbf{w}^*\right)+\mathbb{E}\left[f\left(\widetilde{\mathbf{w}}_s\right)\right]-f\left(\mathbf{w}^*\right)\right) \\
&\quad\leq \mathbb{E}\left[\left\|\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}-\mathbf{w}^*\right\|^2\right]-2 \alpha(1-2 \alpha \ell)\left(\mathbb{E}\left[f\left(\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}\right)\right]\right. \\
&\quad\left.-f\left(\mathbf{w}^*\right)\right)+4 \alpha^2 \ell\left(\mathbb{E}\left[f\left(\widetilde{\mathbf{w}}_s\right)\right]-f\left(\mathbf{w}^*\right)\right).
\end{aligned}$$

Summing the above inequality over r and rearranging gives us

$$\begin{aligned}
&\frac{1}{22 \kappa} \sum_{r=1}^{22 \kappa} \mathbb{E}\left[f\left(\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}\right)\right]-f\left(\mathbf{w}^*\right) \\
&\quad\leq \frac{8 \ell}{33 \kappa} \mathbb{E}\left[\left\|\mathbf{w}_{\mathbf{1}}^{\mathbf{s}}-\mathbf{w}^*\right\|\right]^2 \\
&\quad+\frac{1}{2}\left(\mathbb{E}\left[f\left(\widetilde{\mathbf{w}}_s\right)\right]-f\left(\mathbf{w}^*\right)\right) \\
&\quad\leq \frac{8}{33}\left(\mathbb{E}\left[f\left(\widetilde{\mathbf{w}}_s\right)\right]-f\left(\mathbf{w}^*\right)\right)+\frac{1}{2}\left(\mathbb{E}\left[f\left(\widetilde{\mathbf{w}}_s\right)\right]\right. \\
&\quad\left.-f\left(\mathbf{w}^*\right)\right) \leq \frac{3}{4}\left(\mathbb{E}\left[f\left(\widetilde{\mathbf{w}}_s\right)\right]-f\left(\mathbf{w}^*\right)\right).
\end{aligned}$$

Using convexity, we can now easily see that

$$\begin{aligned}
\mathbb{E}\left[f\left(\widetilde{\mathbf{w}}_{s+1}\right)\right]-f\left(\mathbf{w}^*\right) &\leq \frac{1}{22 \kappa} \sum_{r=1}^{22 \kappa} \mathbb{E}\left[f\left(\mathbf{w}_{\mathbf{r}}^{\mathbf{s}}\right)\right] \\
&-f\left(\mathbf{w}^*\right) \leq \frac{3}{4}\left(\mathbb{E}\left[f\left(\widetilde{\mathbf{w}}_s\right)\right]-f\left(\mathbf{w}^*\right)\right).
\end{aligned}$$

This proves the result. □

6 Principal Component Analysis (PCA)

We now move from the convex optimization setting to a new one: that of principal component analysis (PCA). PCA, originally introduced in Hotelling[33], is a way of finding important directions in the data and do dimensionality reduction. It has been widely used in several applications[39]. PCA is equivalent to the following: suppose each data point $\mathbf{x_i} \in \mathbb{R}^d$, $i=1,\ldots,n$ is drawn independently from some underlying probability distribution; then given an integer $k \geq 1$, find the top-k eigenvectors of $\mathbf{\Sigma} \stackrel{\text{def}}{=} \mathbb{E}\left[\mathbf{x_i}\mathbf{x_i}^\top\right]$. Here again, one solves this problem using samples by constructing the empirical covariance matrix $\widehat{\mathbf{\Sigma}} \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^n \mathbf{x_i}\mathbf{x_i}^\top$ and finding the top-q eigenvectors of $\widehat{\mathbf{\Sigma}}$.

In several settings either due to the sequential nature of data generation or due to the large ambient dimension d, storing all the samples $\mathbf{x_1},\ldots,\mathbf{x_n}$ or the full empirical covariance matrix $\widehat{\mathbf{\Sigma}}$ might not be feasible[31, 60, 67]. Instead we would like to estimate the top-k eigenvectors of $\boldsymbol{\Sigma}$ using only $\mathcal{O}(dk)$ space, by sequentially going over the samples $\mathbf{x_1}, \ldots, \mathbf{x_n}$ (and not storing them). Let us first consider the simple case of $q = 1$, i.e., we are interested in estimating the top eigenvector of $\boldsymbol{\Sigma}$. The optimization problem corresponding to this is given by

$$\max_{\mathbf{w} \in \mathbb{R}^d : \|\mathbf{w}\|=1} \mathbf{w}^\top \boldsymbol{\Sigma} \mathbf{w} = \mathbb{E}\left[\langle \mathbf{x}, \mathbf{w} \rangle^2\right].$$

A natural approach would be to perform stochastic gradient ascent (SGA) (*ascent* as opposed to *descent* since we are dealing with a *maximization* rather than a *minimization* problem). Note also that we have the constraint $\|\mathbf{w}\| = 1$, so the SGA step will be followed by a projection step onto the unit ball. So the SGA with projection algorithm is

$$\tilde{\mathbf{w}} = \mathbf{w_k} + \alpha_k \langle \mathbf{x_k}, \mathbf{w_k} \rangle \mathbf{x_k} \quad \text{and} \quad \mathbf{w_{k+1}} = \frac{\tilde{\mathbf{w}}}{\|\tilde{\mathbf{w}}\|}. \tag{8}$$

The above algorithm was first considered by Oja[53] and since, has been known as Oja's algorithm. Oja[53] in fact showed that for a suitable choice of stepsize sequence α_k, the iterates $\mathbf{w_k}$ asymptotically converge to the top eigenvector of $\boldsymbol{\Sigma}$. Obtaining nonasymptotic convergence rates has been a much more recent endeavor[8, 13, 34, 48]. As an example of the kind of results one has in this situation, we state here (a simplified version of) the main result of Jain et al.[34].

Theorem 4 Suppose data points $\mathbf{x_i}$ *come from a distribution* $\mathcal{D}$ *such that*

- $\|\mathbf{x_i}\| \leq \mathcal{M}$ *with probability* 1,
- $\lambda_1 > \lambda_2$ *denote the largest and second largest eigenvalues respectively of the covariance matrix* $\boldsymbol{\Sigma} \overset{\text{def}}{=} \mathbb{E}\left[\mathbf{x_i}\mathbf{x_i}^\top\right]$ *and*
- $\mathbf{w}^*$ *denotes the largest eigenvector of* $\boldsymbol{\Sigma}$ (*i.e., eigenvector corresponding to eigenvalue* λ_1).

If $\mathbf{w_1}$ *is chosen from standard normal distribution, then for an appropriate stepsize sequence* α_k, *for any fixed* $t \geq \frac{40\mathcal{M}\lambda_1 \log^2 d}{(\lambda_1-\lambda_2)^2}$, *with probability at least* 3 / 4, *we have*

$$\sin^2(\mathbf{w_t}, \mathbf{w}^*) \leq \frac{\mathcal{M}\lambda_1 \log \mathrm{d}}{(\lambda_1 - \lambda_2)^2} \cdot \frac{1}{\mathrm{t}},$$

where $\sin(\mathbf{w_t}, \mathbf{w}^*)$ *denotes the* sin *of the angle between* $\mathbf{w_t}$ *and* $\mathbf{w}^*$.

We will only give a proof outline of the above theorem. Interested readers may refer Jain et al.[34] for the full proof.

Proof (Proof outline) The main idea behind the proof of Theorem 4 is that one can write the final iterate $\mathbf{w_t}$ as a scaled version of a linear function of the initial vector $\mathbf{w_1}$:

$$\mathbf{w_t} = \frac{1}{Z} \prod_{k=1}^{t-1} \left(\mathbf{I} + \alpha_k \mathbf{x_k}\mathbf{x_k}^\top\right) \cdot \mathbf{w_1}.$$

Here $\mathbf{I}$ denotes the identity matrix and Z is a normalization constant (that depends on $\mathbf{x_k}$ and $\mathbf{w_1}$). Since $\sin(\cdot, \cdot)$ does not depend on the magnitude of the vectors, we can ignore Z and try to bound $\sin\left(\prod_{k=1}^{t-1}\left(\mathbf{I} + \alpha_k \mathbf{x_k}\mathbf{x_k}^\top\right) \cdot \mathbf{w_1}, \mathbf{w}^*\right)$. Let us denote $\mathbf{S_t} \overset{\text{def}}{=} \prod_{k=1}^{t-1}\left(\mathbf{I} + \alpha_k \mathbf{x_k}\mathbf{x_k}^\top\right)$ for notational simplicity. Using the definition of sin and the fact that $\|\mathbf{w}^*\| = 1$, we have

$$\begin{aligned}\sin^2\left(\mathbf{S_t}\mathbf{w_1}, \mathbf{w}^*\right) &= 1 - \frac{\langle \mathbf{S_t}\mathbf{w_1}, \mathbf{w}^* \rangle^2}{\|\mathbf{S_t}\mathbf{w_1}\|^2} \\ &= \frac{\mathbf{w_1}^\top \mathbf{S_t}^\top \left(\mathbf{I} - \mathbf{w}^*\mathbf{w}^{*\top}\right) \mathbf{S_t}\mathbf{w_1}}{\mathbf{w_1}^\top \mathbf{S_t}^\top \mathbf{S_t}\mathbf{w_1}}.\end{aligned} \tag{9}$$

Since $\mathbf{w_1}$ is chosen uniformly from the unit sphere, it turns out that the expression above is approximately equal to (up to constant factors) $\frac{\mathrm{Tr}\left(\mathbf{S_t}^\top\left(\mathbf{I}-\mathbf{w}^*\mathbf{w}^{*\top}\right)\mathbf{S_t}\right)}{\mathrm{Tr}\left(\mathbf{S_t}^\top \mathbf{S_t}\right)}$. The reason for this is that for any PSD matrix $\mathbf{A}$, if $\mathbf{w_1}$ is a standard normal vector, then we have $\mathbf{w_1}^\top \mathbf{A} \mathbf{w_1} \approx \mathrm{Tr}(\mathbf{A})$ (up to constant factors) with high probability. So we have concluded that

$$\sin^2\left(\mathbf{S_t}\mathbf{w_1}, \mathbf{w}^*\right) = \mathcal{O}\left(\frac{\mathrm{Tr}\left(\mathbf{S_t}^\top\left(\mathbf{I} - \mathbf{w}^*\mathbf{w}^{*\top}\right)\mathbf{S_t}\right)}{\mathrm{Tr}\left(\mathbf{S_t}^\top \mathbf{S_t}\right)}\right).$$

The nice thing about the above expression is that we have managed to aggregate the behavior of the algorithm into the trace of certain matrices. These matrices are relatively simple and comprise of product of matrices of the form $\mathbf{I} + \alpha_k \mathbf{x_k}\mathbf{x_k}^\top$. Since $\mathbf{x_k}$ themselves are random, one can bound the expected values of these quantities and use Markov/Chebyshev inequalities to obtain the desired bounds with constant probability. □

A particularly nice aspect of the above result is that it matches the best known results we know of even if one were allowed to compute the empirical covariance matrix. This result has since been extended to the case where there is no gap, i.e., $\lambda_1 = \lambda_2$ (in this case, one shows convergence of Rayleigh quotient)[9] and to the case $q \geq 2$[8]. The best known results for $q \geq 2$, however, are significantly suboptimal compared to the case when we are allowed to compute the empirical covariance matrix. Bridging this gap is an important open problem.

7 Nonconvex Optimization

Nonconvex optimization has always been an important part of machine learning as most machine learning problems, in their original form, turn out to be nonconvex. Examples include k-means clustering, matrix factorization with sparsity/rank/nonnegativity constraints, density estimation, training neural networks and so on. Since nonconvex optimization is NP-hard in the worst case, one cannot hope to obtain polynomial convergence rates to global minima or even local minima. The goal is more modest—convergence to *stationary points*. Later on in this section, we will give examples where finding such stationary points guarantees a good solution in terms of function value. We first introduce the notion of first-order stationary point.

Definition 1 A point $\mathbf{w}$ is said to be an ϵ-first order stationary point (ϵ-FOSP) of $f(\cdot)$ if $\left\|\nabla f(\mathbf{w})\right\| \leq \epsilon$.

The following classical (and folklore) result shows the convergence of SGD to first-order stationary points:

Theorem 5 *Suppose the function $f(\cdot)$ is ℓ-smooth (Assumption 3) and lower bounded i.e., $\min_{\mathbf{w}} f(\mathbf{w}) > -\infty$. Suppose further that at each step, we perform SGD with stochastic gradient $\zeta_{\mathbf{k}}$ satisfying $\mathbb{E}[\zeta_{\mathbf{k}}] = \nabla f(\mathbf{w_k})$ and $\mathbb{E}\left[\|\zeta_{\mathbf{k}}\|^2\right] \leq \sigma^2$. Fix t and choose stepsizes $\alpha[k] = \alpha \overset{\mathrm{def}}{=} \sqrt{\frac{2(f(\mathbf{w_0}) - \min_{\mathbf{w}} f(\mathbf{w}))}{\ell\sigma^2}}$. Then,*

$$\min_{k\in[t]} \mathbb{E}\left[\left\|\nabla f(\mathbf{w_k})\right\|^2\right] \leq \sqrt{\frac{2\sigma^2\ell\left(f(\mathbf{w_0}) - \min_{\mathbf{w}} f(\mathbf{w})\right)}{t}}.$$

Proof Using smoothness and properties of stochastic gradients,

Reorganizing and adding, we obtain

$$\frac{1}{t}\sum_{k=1}^{t} \mathbb{E}\left[\left\|\nabla f(\mathbf{w_k})\right\|^2\right] \leq \frac{f(\mathbf{w_1}) - \min_{\mathbf{w}} f(\mathbf{w})}{\alpha t} + \frac{\alpha\ell\sigma^2}{2}.$$

Plugging in the choice of α proves the theorem. □

While GD converges to FOSPs, it turns out that FOSPs are not always good solutions to the minimization problem $\min_{\mathbf{w}} f(\mathbf{w})$ that we are interested in. We would like to do better than first-order stationarity. This brings us to

Definition 2 A point $\mathbf{w}$ is said to be an (ϵ_g, ϵ_h)-second order stationary point ((ϵ_g, ϵ_h)-SOSP) of $f(\cdot)$ if $\left\|\nabla f(\mathbf{w})\right\| \leq \epsilon_g$ and $\lambda_{\min}\left(\nabla^2 f(\mathbf{w})\right) \geq -\epsilon_h$.

In the context of several machine learning problems, it turns out that while these problems have several highly suboptimal FOSPs[25], all SOSPs are close to optimal[14, 16, 24, 30, 40, 49]. This motivates the quest for finding SOSPs and not just FOSPs. Let us call FOSPs which are not SOSPs as first-order saddle points. It turns out that FOSPs, which are not SOSPs are actually unstable fixed points of gradient flow (i.e., GD with step size $\to 0$), and the set of all points from where gradient flow converges to first-order saddle points is a measure zero set (in the ambient Lebesgue measure)[47, 54]. However, GD with random initialization might still take exponential time to escape these first order saddle points and converge to SOSPs[28]. In a remarkable result, Ge et al.[29] showed that a little amount of noise in the updates of GD can help in escaping first order saddles and converge to SOSPs in polynomial time.This means that not doing exact gradient descent is not a disadvantage but rather a potent weapon! Jin et al.[37] further improved upon the results of Ge et al.[29] by decreasing the dependence on the dimension from polynomial to polylog. The perturbed gradient descent (PGD) algorithm of Jin et al.[37] is given in Algorithm 2. Note that the algorithm essentially performs GD but perturbs the iterate once in a while by adding appropriate noise.

$$\begin{aligned}
\mathbb{E}[f(\mathbf{w_{k+1}})] &\leq \mathbb{E}[f(\mathbf{w_k})] + \mathbb{E}[\langle\nabla f(\mathbf{w_k}), \mathbf{w_{k+1}} - \mathbf{w_k}\rangle] + \frac{\ell}{2}\|\mathbf{w_{k+1}} - \mathbf{w_k}\|^2 \\
&= \mathbb{E}[f(\mathbf{w_k})] - \alpha\mathbb{E}[\langle\nabla f(\mathbf{w_k}), \zeta_{\mathbf{k}}\rangle] + \frac{\alpha^2\ell}{2}\|\zeta_{\mathbf{k}}\|^2 \\
&= \mathbb{E}[f(\mathbf{w_k})] - \alpha\mathbb{E}\left[\left\|\nabla f(\mathbf{w_k})\right\|^2\right] + \frac{\alpha^2\sigma^2\ell}{2}.
\end{aligned}$$

Algorithm 2 Perturbed Gradient Descent

```
Input: $\mathbf{w_1}$, learning rate $\alpha$, perturbation radius $r$, time interval $\mathcal{T}$, tolerance $\epsilon$.
  $k_{\text{perturb}} = 1$
  for $k = 1, \ldots, t$ do
    if $\|\nabla f(\mathbf{w_k})\| \le \epsilon$ and $k - k_{\text{perturb}} > \mathcal{T}$ then
      $\mathbf{w_k} \leftarrow \mathbf{w_k} - \eta\zeta_{\mathbf{k}}$, ($\zeta_{\mathbf{k}} \sim \text{Uniform}(B_0(r))$);   $k_{\text{perturb}} \leftarrow k$
    $\mathbf{w_{k+1}} \leftarrow \mathbf{w_k} - \alpha\nabla f(\mathbf{w_k})$
```

Before we state the result, we need to introduce the following assumption:

Assumption 4 A twice differentiable function $f(\cdot)$ is said to be ρ-Hessian Lipschitz if $\left\|\nabla^2 f(\mathbf{w}) - \nabla^2 f(\mathbf{z})\right\| \le \rho\|\mathbf{w} - \mathbf{z}\|$, for every $\mathbf{w}$ and $\mathbf{z}$.

The following theorem gives the convergence rate of PGD for smooth and Hessian Lipschitz functions:

Theorem 6 *Suppose $f(\cdot) : \mathbb{R}^d \to \mathbb{R}$ is ℓ-smooth (Assumption 3) and ρ-Hessian Lipschitz (Assumption 4). Then, for any given ϵ, for appropriate choice of the step size α and $t \ge \widetilde{\mathcal{O}}\left(\frac{\ell(f(\mathbf{w_0}) - \min_{\mathbf{w}} f(\mathbf{w}))}{\epsilon^2}\right)$, we have that at least half of the iterates $\mathbf{w_k}$, $k = 1, \ldots, t$ are $\left(\epsilon, \sqrt{\rho\epsilon}\right)$-SOSPs.*

We will only present a high level outline of the proof of the above theorem. Interested readers may refer Jin et al.[37] for the full proof.

Proof (Proof outline) First, the algorithm is a descent algorithm, i.e., the function value (almost) always decreases. The proof follows by showing that *if an iterate is not an SOSP then the function value decreases significantly in the next few steps*. Since the function is lower bounded, it can decrease significantly only a limited number of times. So at least half of the iterates are SOSPs. The main ideas in the proof of the above result are as follows:

1. From the proof of Theorem 5, the function value decreases significantly if an iterate is not an FOSP.
2. If the iterates $\mathbf{w_k}$ move significantly, then the function value again decreases significantly. This is called improve-or-localize property.
3. If the current iterate is an FOSP, but not an SOSP, and if the iterates $\mathbf{w_k}$ do not move much, then one can approximate the function with a quadratic (using the Hessian) by Hessian Lipschitz property.
4. If $f(\cdot)$ were quadratic with a significant negative eigenvalue, one can analyze the exact behavior of PGD as matrix power update and show that PGD decreases function value significantly.
5. By controlling the approximation from quadratic and combining with the decrease from item 3 above, one can show that the function value decreases significantly even when the iterates do not move much.
6. As mentioned above, since the function is lower bounded, the function value cannot decrease arbitrarily. So after enough number of iterations, it has to be the case that several of the Hessians do not have a large negative eigenvalue and hence the corresponding points have to be SOSPs.

□

7.1 Finite Sum Setting

In this section, we consider the finite sum setting (5) where neither the individual $\phi_i(\cdot)$ nor the sum function $f(\cdot)$ is convex. In this setting one again wonders whether it is possible to combine GD and SGD in some way to obtain the best of both worlds. Allen-Zhu and Hazan[7] and Reddi et al.[58] independently showed that the SVRG algorithm can indeed be extended to the nonconvex setting and obtains better rates than GD and SGD. The nonconvex SVRG algorithm is presented in Algorithm 3. We now state the following theorem (with out proof) which gives convergence rates of nonconvex SVRG to FOSPs:

Algorithm 3 Nonconvex SVRG

```
Input: $\mathbf{w_1}$, learning rate $\alpha$, functions $\phi_1, \cdots, \phi_n$.
  $\widetilde{\mathbf{w}} \leftarrow \mathbf{w_1}$
  $\widetilde{\mathbf{g}} \leftarrow \frac{1}{n}\sum_{i=1}^{n} \nabla\phi_i(\widetilde{\mathbf{w}})$
  for $k = 1, \ldots, t$ do
    Pick $i$ uniformly at random from $1, \cdots, n$
    $\mathbf{w_{k+1}} \leftarrow \mathbf{w_k} - \alpha\left(\nabla\phi_i(\mathbf{w_k}) - \nabla\phi_i(\widetilde{\mathbf{w}}) + \widetilde{\mathbf{g}}\right)$
```

Theorem 7 Suppose $f(\cdot)$ is a finite sum function (5) where each component $\phi_i(\cdot)$ is ℓ-smooth (Assumption 3). For any $\epsilon > 0$, if the stepsize is chosen $\alpha = \frac{1}{\ell n^{2/3}}$ and *$t > \mathcal{O}(n)$, then we have*

$$\min_k \mathbb{E}\left[\left\|\nabla f(\mathbf{w_k})\right\|^2\right] \leq \frac{\ell n^{2/3}\left(f(\mathbf{w_0}) - \min_{\mathbf{w}} f(\mathbf{w})\right)}{t}.$$

Note that the above result obtains a convergence rate of $\mathcal{O}\left(\frac{1}{t}\right)$ as compared to the $\mathcal{O}\left(\frac{1}{\sqrt{t}}\right)$ of Theorem 5. There have also been recent works on designing SVRG style algorithms for finding SOSPs[5, 6, 10]. These papers combine ideas from the proof of Theorem 7 as well as the PCA results mentioned in Sect. 6 to efficiently escape saddle points.

8 Summary and Future Directions

In this work, we have surveyed some important results on the performance of SGD in various settings. The several applications of SGD in machine learning have led to new results on the performance of SGD in various contexts as well as new algorithms with better performance. Still, there are a number of important questions about SGD that we do not know the answer to. We mention (in our view) the most important ones below:

- *Step size schedules* Given a fixed time horizon t, what is the best step size schedule α_k for $k = 1, \ldots, t$? Even in the setting of convex optimization, while the $\frac{1}{\sqrt{k}}$ and $\frac{1}{k}$ step sizes of Theorems 1 and 2 are optimal in the worst case, they are suboptimal on a problem to problem basis. Polyak and Juditsky[56] indeed shows that using a constant learning rate with averaging at the end achieves asymptotically optimal rates for *quadratic problems*. However, for nonquadratic problems, decaying step sizes seems necessary. While there are several papers on this topic[12, 27, 50], there is still not a clear answer to the right decay schemes for general (even convex) functions.
- *Minibatching* In finite sum settings, in practice, one uses SGD with minibatching, i.e., instead of picking an i at random and using $\nabla\phi_i(\mathbf{w})$, one chooses $i_1, \ldots, i_b$ at random and uses the average $\frac{1}{b}\sum_{j=1}^{b} \nabla_{i_j}(\mathbf{w})$. Here b is known as the minibatch size. The main reason for doing this in practice is parallelization since the b gradient computations can be done simultaneously on different machines. At the same time, minibatching also leads to smaller variance in the stochastic gradient and one would expect this to give faster convergence rate. While $b = 1$ corresponds to full SGD, $b = n$ corresponds to full GD. We would like to know what the optimal value of b is for which we get the best convergence rates. Most existing results that try to do this comparison, e.g., Dekel et al.[26] use (loose) upper bounds on the performance of minibatch SGD which does not give conclusive answers. To the best of our knowledge, the only work that studies the true effect of minibatch SGD is for the linear least squares setting[36]. Extending such results beyond linear regression is an important open problem.
- *Acceleration* While any improvement in the convergence rate of an algorithm is termed acceleration, we are referring here in particular to the technique of momentum or Nesterov acceleration. Can Nesterov's acceleration technique be used to improve the convergence rate of SGD? In a remarkable result, Lan[44] showed for the first time that momentum techniques can indeed improve stochastic gradient methods if the stochastic gradient has bounded variance. Most machine learning settings, however, do not fall in this category. For the problem of strongly convex linear least squares regression (which is not covered by Lan[44]), Jain et al.[35] showed that it is indeed possible to again use these techniques to improve the convergence rate of SGD. Extending it to non-strongly convex linear least squares regression is an open problem. Much less is known for problems beyond linear least squares regression.
- *Parallelization and distributed algorithms* One serious issue with SGD on large-scale problems is its sequential nature—the iterations have to be performed one after another. How do we parallelize this computation. So far, there have been three approaches (1) minibatching, (2) model averaging and iii) asynchronous SGD. We have already mentioned questions related to minibatching above. Model averaging refers to running SGD independently on different machines and averaging the resultant vectors $\mathbf{w}$ from those different machines. First this makes sense only in the convex setting. Even in the convex (or locally convex even if the function is globally nonconvex) setting, this is known to be optimal only for linear least squares regression. While we do obtain some bounds in the general convex setting, it is unknown whether this is the optimal way to make use of different

machines. Finally, asynchronous SGD refers to the setting where different machines maintain their own local copies of the iterates $\mathbf{w_k}$ and perform SGD on them while synchronizing them only once in a while. There have been some interesting results on this approach, e.g., Recht et al.[57] but they cover only some special cases, and their relevance to real-world machine learning problems is unclear.

- *Stochastic quasi Newton methods* In the context of deterministic gradient-based methods, ones that construct approximations of the Hessian (known as quasi-Newton methods) such as BFGS, l-BFGS, see chapters 8 and 9 in Nocedal and Wright[52], are superior to gradient descent. While there has been some work on designing such methods with stochastic gradients[17, 21, 62, 66], they have so far not been successfully applied to large-scale machine learning problems.

We believe that making progress on answering the above questions is necessary to advance state of the art in optimization in machine learning.

9 Resources

The simplicity of SGD (2) and its variants such as SVRG means that in settings where stochastic gradients are computable, implementing it is straightforward. In cases where writing code for computing stochastic gradients might be involved, one might try to use autodifferentiation packages such as Autodiff [1], Autograd[2] and Casadi[3], etc. In the context of training neural networks, there are frameworks that are quite easy to use such as PyTorch[55], TensorFlow[4], etc. The painstaking part of using these in practice is, however, the choice of step sizes α_k for which there is, as yet, no fully automatic procedure that works well, and one has to choose them manually and pick what works best.

Received: 12 November 2018 Accepted: 16 January 2019
Published online: 12 February 2019

References

1. Autodiff. https://pypi.org/project/autodiff/. Accessed 30 Nov 2018
2. Autograd. https://github.com/HIPS/autograd. Accessed 30 Nov 2018
3. Casadi. https://web.casadi.org/. Aaccessed 30 Nov 2018
4. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M et al (2016) Tensorflow: a system for large-scale machine learning. OSDI 16:265–283
5. 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, ACM, pp 1195–1199
6. Allen-Zhu Z (2017) Natasha 2: faster non-convex optimization than sgd. arXiv preprint arXiv:1708.08694
7. Allen-Zhu Z, Hazan E (2016) Variance reduction for faster non-convex optimization. In: International conference on machine learning, pp 699–707
8. Allen-Zhu Z, Li Y (2017) First efficient convergence for streaming k-PCA: a global, gap-free, and near-optimal rate. In: Foundations of computer science (FOCS), 2017 IEEE 58th annual symposium on, IEEE, pp 487–492
9. Allen-Zhu Z, Li Y (2017) Follow the compressed leader: faster online learning of eigenvectors and faster MMWU. In: Precup D, Teh YW (eds) Proceedings of the 34th international conference on machine learning, pp 116–125
10. Allen-Zhu Z, Li Y (2017) Neon2: finding local minima via first-order oracles. arXiv preprint arXiv:1711.06673
11. Amari S (1967) A theory of adaptive pattern classifiers. IEEE Trans Electron Comput 3:299–307
12. Bach F, Moulines E (2013) Non-strongly-convex smooth stochastic approximation with convergence rate $O(1/n)$. In: Advances in neural information processing systems, pp 773–781
13. Balsubramani A, Dasgupta S, Freund Y (2013) The fast convergence of incremental PCA. In: Advances in neural information processing systems, pp 3174–3182
14. Bandeira AS, Boumal N, Voroninski V (2016) On the low-rank approach for semidefinite programs arising in synchronization and community detection. In: Conference on learning theory, pp 361–382
15. Benaïm M (1999) Dynamics of stochastic approximation algorithms. In: Seminaire de probabilites XXXIII, Springer, pp 1–68
16. Bhojanapalli S, Boumal N, Jain P, Netrapalli P (2018) Smoothed analysis for low-rank solutions to semidefinite programs in quadratic penalty form. In: Proceedings of the 31st conference on learning theory, pp 3243–3270
17. Bordes A, Bottou L, Gallinari P (2009) SGD-QN: careful quasi-Newton stochastic gradient descent. J Mach Learn Res 10(Jul):1737–1754
18. Borkar VS (2009) Stochastic approximation: a dynamical systems viewpoint, vol 48. Springer, Berlin
19. Bottou L, Bousquet O (2008) The tradeoffs of large scale learning. In: Advances in neural information processing systems, pp 161–168
20. Bottou L, Curtis FE, Nocedal J (2018) Optimization methods for large-scale machine learning. SIAM Rev 60(2):223–311

21. Byrd RH, Hansen SL, Nocedal J, Singer Y (2016) A stochastic quasi-Newton method for large-scale optimization. SIAM J Optim 26(2):1008–1031
22. Candès EJ, Recht B (2009) Exact matrix completion via convex optimization. Found Comput Math 9(6):717
23. Cauchy A (1847) Compte rendu des s eances de lacademie des sciences. Comptes Rendus Hebd Seances Acad Sci 21(25):536–538
24. Choromanska A, Henaff M, Mathieu M, Arous GB, LeCun Y (2015) The loss surfaces of multilayer networks. In: Artificial intelligence and statistics, pp 192–204
25. Dauphin YN, Pascanu R, Gulcehre C, Cho K, Ganguli S, Bengio Y (2014) Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In: Advances in neural information processing systems, pp 2933–2941
26. Dekel O, Gilad-Bachrach R, Shamir O, Xiao L (2012) Optimal distributed online prediction using mini-batches. J Mach Learn Res 13(Jan):165–202
27. Dieuleveut A, Durmus A, Bach F (2017) Bridging the gap between constant step size stochastic gradient descent and markov chains. arXiv preprint arXiv:1707.06386
28. Du SS, Jin C, Lee JD, Jordan MI, Singh A, Poczos B (2017) Gradient descent can take exponential time to escape saddle points. In: Advances in neural information processing systems, pp 1067–1077
29. Ge R, Huang F, Jin C, Yuan Y (2015) Escaping from saddle pointsonline stochastic gradient for tensor decomposition. In: Conference on learning theory, pp 797–842
30. Ge R, Jin C, Zheng Y (2017) No spurious local minima in nonconvex low rank problems: a unified geometric analysis. arXiv preprint arXiv:1704.00708
31. Hall PM, Marshall AD, Martin RR (1998) Incremental eigenanalysis for classification. In: BMVC, vol 98, Citeseer, pp 286–295
32. Hearst MA, Dumais ST, Osuna E, Platt J, Scholkopf B (1998) Support vector machines. IEEE Intell Syst Appl 13(4):18–28
33. Hotelling H (1933) Analysis of a complex of statistical variables into principal components. J Educ Psychol 24(6):417
34. Jain P, Jin C, Kakade SM, Netrapalli P, Sidford A (2016) Streaming PCA: matching matrix Bernstein and near-optimal finite sample guarantees for Oja's algorithm. In: Conference on learning theory, pp 1147–1164
35. Jain P, Kakade SM, Kidambi R, Netrapalli P, Sidford A (2017) Accelerating stochastic gradient descent. arXiv preprint arXiv:1704.08227
36. Jain P, Kakade SM, Kidambi R, Netrapalli P, Sidford A (2018) Parallelizing stochastic gradient descent for least squares regression: mini-batching, averaging, and model misspecification. J Mach Learn Res 18(223):1–42. http://jmlr.org/papers/v18/16-595.html
37. 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, pp 1724–1732
38. Johnson R, Zhang T (2013) Accelerating stochastic gradient descent using predictive variance reduction. In: Advances in neural information processing systems, pp 315–323
39. Jolliffe I (2011) Principal component analysis. In: International encyclopedia of statistical science, Springer, pp 1094–1096
40. Kawaguchi K (2016) Deep learning without poor local minima. In: Advances in neural information processing systems, pp 586–594
41. Kiefer J, Wolfowitz J (1952) Stochastic estimation of the maximum of a regression function. Ann Math Stat 23(3):462–466
42. Kushner H, Yin GG (2003) Stochastic approximation and recursive algorithms and applications, vol 35. Springer Science & Business Media, Berlin
43. Kushner HJ, Clark DS (2012) Stochastic approximation methods for constrained and unconstrained systems, vol 26. Springer Science & Business Media, Berlin
44. Lan G (2012) An optimal method for stochastic composite optimization. Math Program 133(1–2):365–397
45. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
46. LeCun YA, Bottou L, Orr GB, Müller KR (2012) Efficient backprop. In: Neural networks: tricks of the trade, Springer, pp 9–48
47. Lee JD, Simchowitz M, Jordan MI, Recht B (2016) Gradient descent only converges to minimizers. In: Conference on learning theory, pp 1246–1257
48. Li CJ, Wang M, Liu H, Zhang T (2018) Near-optimal stochastic approximation for online principal component estimation. Math Program 167(1):75–97
49. Mei S, Misiakiewicz T, Montanari A, Oliveira RI (2017) Solving SDPS for synchronization and MaxCut problems via the Grothendieck inequality. arXiv preprint arXiv:1703.08729
50. Moulines E, Bach FR (2011) Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In: Advances in neural information processing systems, pp 451–459
51. Nemirovsky AS, Yudin DB (1983) Problem complexity and method efficiency in optimization
52. Nocedal J, Wright SJ (2006) Numerical optimization 2nd
53. Oja E (1982) Simplified neuron model as a principal component analyzer. J Math Biol 15(3):267–273
54. Panageas I, Piliouras G (2016) Gradient descent only converges to minimizers: non-isolated critical points and invariant regions. arXiv preprint arXiv:1605.00405
55. Paszke A, Gross S, Chintala S, Chanan G (2017) Pytorch. https://pytorch.org/. Accessed 1 Nov 2018
56. Polyak BT, Juditsky AB (1992) Acceleration of stochastic approximation by averaging. SIAM J Control Optim 30(4):838–855

57. Recht B, Re C, Wright S, Niu F (2011) Hogwild: a lock-free approach to parallelizing stochastic gradient descent. In: Advances in neural information processing systems, pp 693–701
58. 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
59. Robbins H, Monro S (1951) A stochastic approximation method. Ann Math Stat 22(3):400–407
60. Ross DA, Lim J, Lin RS, Yang MH (2008) Incremental learning for robust visual tracking. Int J Comput Vis 77(1–3):125–141
61. Schmidt M, Le Roux N, Bach F (2017) Minimizing finite sums with the stochastic average gradient. Math Program 162(1–2):83–112
62. Schraudolph NN, Yu J, Günter S (2007) A stochastic quasi-Newton method for online convex optimization. In: Artificial intelligence and statistics, pp 436–443
63. Shalev-Shwartz S, Zhang T (2013) Stochastic dual coordinate ascent methods for regularized loss minimization. J Mach Learn Res 14(Feb):567–599
64. Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Stat Soc Ser B (Methodol) 58(1):267–288
65. Tsypkin YZ, Nikolic ZJ (1971) Adaptation and learning in automatic systems, vol 73. Academic Press, New York
66. Wang X, Ma S, Goldfarb D, Liu W (2017) Stochastic quasi-Newton methods for nonconvex stochastic optimization. SIAM J Optim 27(2):927–956
67. Weng J, Zhang Y, Hwang WS (2003) Candid covariance-free incremental principal component analysis. IEEE Trans Pattern Anal Mach Intell 25(8):1034–1040
68. Yann L (1987) Modeles connexionnistes de lapprentissage. Ph.D. thesis, These de Doctorat, Universite Paris 6

Praneeth Netrapalli is currently a researcher at Microsoft Research India, Bangalore. His research focuses on designing efficient and provable algorithms for machine learning. His work was one of the first to shed light on the performance of alternating minimization (a meta-heuristic which is widely used across several fields) on some well-studied machine-learning problems, and subsequently led to the design of other faster algorithms. His recent work is focused on designing simple and effective algorithms for general non-convex optimization as well as stochastic optimization problems, which are ubiquitous in machine learning. Praneeth obtained B-tech from IIT Bombay in 2007 and MS and PhD from The University of Texas at Austin in 2011 and 2014, respectively, all in Electrical Engineering. From 2014 to 2016, he was a postdoctoral researcher at Microsoft Research New England, Cambridge MA. From 2007 to 2009, he worked as a quantitative analyst at Goldman Sachs, Bangalore, where his work focused on evaluating prices and risk of financial derivatives. More information about his research is available on his home page http://praneethnetrapalli.org/.