STOCHASTIC GRADIENT DESCENT: WHERE OPTIMIZATION MEETS MACHINE LEARNING

RACHEL WARD

ABSTRACT

Stochastic gradient descent (SGD) is the de facto optimization algorithm for training neural networks in modern machine learning, thanks to its unique scalability to problem sizes where the data points, the number of data points, and the number of free parameters to optimize are on the scale of billions. On the one hand, many of the mathematical foundations for stochastic gradient descent were developed decades before the advent of modern deep learning, from stochastic approximation to the randomized Kaczmarz algorithm for solving linear systems. On the other hand, the omnipresence of stochastic gradient descent in modern machine learning and the resulting importance of optimizing performance of SGD in practical settings have motivated new algorithmic designs and mathematical breakthroughs. In this note, we recall some history and state-of-the-art convergence theory for SGD which is most useful in modern applications where it is used. We discuss recent breakthroughs in adaptive gradient variants of stochastic gradient descent, which go a long way towards addressing one of the weakest points of SGD: its sensitivity and reliance on hyperparameters, most notably, the choice of step-sizes.

MATHEMATICS SUBJECT CLASSIFICATION 2020

Primary 74P99; Secondary 93E35, 46N30, 46N40

KEYWORDS

Adaptive gradient methods, machine learning, smoothness, stepsize, stochastic approximation

ICM INTERNATIONAL CONGRESS OF MATHEMATICIANS 2022 JULY 6—14

Proc. Int. Cong. Math. 2022, Vol. 7, pp. 5140–5153
DOI 10.4171/ICM2022/171

1. INTRODUCTION

In the past several decades, *randomized* algorithms have slowly gained popularity and established legitimacy as scalable extensions of classical deterministic algorithms to large scales. Perhaps the most widely used randomized algorithm today is *stochastic gradient descent*, which has established itself in the past decade as the de facto optimization method for training artificial neural networks.

A common optimization problem in large-scale machine learning involves a training set $\{(x_1, y_1), \dots, (x_n, y_n)\}$ (we suppose that the labels $y_j \in \mathbb{R}$ for simplicity), a parameterized family of prediction functions h, and a least squares minimization problem of the form

$$\min_{w \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \big(h(x_i; w) - y_i\big)^2. \tag{1.1}$$

In *linear* least squares regression, the prediction function is linear with respect to the weights w; for example, the prediction function is $h(x; w) = \sum_{j=1}^{p} w_j x_j^{p-1}$ in univariate polynomial regression. By contrast, in "neural network" regression, the prediction function h is a parameterized class of highly nonlinear functions inspired by models of how the human brain processes information. The neural network's compositional structure allows for the prediction function $h(x_i; w)$ to be computed at given values of x_i and w by recursively applying successive transformations to the input vector $x_i \in \mathbb{R}^{d_0}$ in layers. For example, a canonical fully-connected layer corresponds to the computation

$$x_i^{(j)} = \sigma(W_j x_i^{(j-1)} + b_j) \in \mathbb{R}^{d_j}, \tag{1.2}$$

where $x_i^{(0)} = x_i$, $W_j \in \mathbb{R}^{d_j \times d_{j-1}}$, the vector $b_j \in \mathbb{R}^{d_j}$ contains the jth layer parameters, and σ is a simple componentwise nonlinear activation function such as the ReLU function $\sigma(x) = \max\{0, x\}$; the total number of parameters $w \in \mathbb{R}^p$ in (1.1) is the sum of the parameters at each of L layers, $w = (W_1, b_1, W_2, b_2, \dots, W_L, b_L)$. "Neural network training" refers to solving the optimization problem (1.1), either exactly or approximately. A particular vector of parameters $w^* \in \mathbb{R}^p$ corresponding to an approximate solution of the optimization problem (1.1) is considered to be a "good" solution if the corresponding neural network function $h(\cdot; w^*)$ has good generalization properties, meaning that when applied to fresh data $\{(\tilde{x}_1, \tilde{y}_1), \dots, (\tilde{x}_m, \tilde{y}_m)\}$ from the same distribution from which the training data was drawn, the distortion $\frac{1}{m} \sum_{i=1}^{m} (h(\tilde{x}_i; w^*) - \tilde{y}_i)^2$ is small. Thus, optimization and generalization must both be taken into account when discussing the performance of a particular algorithm for neural network training. In this note, we will only discuss the optimization component of the stochastic gradient descent algorithm. The generalization of solutions $w^* \in \mathbb{R}^p$ found by SGD tends to be remarkably strong, but this is not as well understood mathematically and represents an important ongoing area of research.

To motivate the stochastic gradient descent algorithm, let us first recall the basic gradient descent procedure for minimizing a differentiable function $F : \mathbb{R}^p \to \mathbb{R}$: starting from an initial point $w_0 \in \mathbb{R}^p$, iterate until convergence

$$w_{j+1} \leftarrow w_j - \eta_j \nabla F(w_j), \tag{1.3}$$

where $\eta_j > 0$ is the step-size prescribed for the jth step. Gradient descent with fixed step-size $\eta_j = \eta$ is guaranteed to converge to a minimizer of F under general conditions, such as if F is smooth (in the sense that F has Lipschitz gradient), convex, and has a finite lower bound. Gradient descent is a first-order iterative algorithm, where first-order means that it only requires computing gradients and not higher-order derivatives. A contributing reason to the feasibility of large-scale neural network training is that neural network optimization is particularly well suited for first-order optimization methods: for neural network prediction functions composed of layers as in (1.2), the gradient of the corresponding objective function $F(w) = \frac{1}{n}\sum_{i=1}^{n}(h(x_i; w) - y_i)^2$ with respect to the parameter vector w can be computed by the chain rule using algorithmic differentiation—a technique referred to as *back propagation* in the machine learning community. However, even a single gradient computation of the form $\nabla F(w) = \frac{1}{n}\sum_{i=1}^{n}\nabla(h(x_i; w) - y_i)^2$ becomes prohibitively expensive as the size of the training set n reaches multiple millions. More generally, when optimizing functions with "finite sum" form, $F(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$,[1] a single gradient evaluation $\nabla F(w)$ requires the computation of all n component function gradients $\nabla f_i(w)$. In such settings, it is natural to consider drawing a random subset of component functions and using the gradient of the random batch of components as a computationally efficient surrogate for the full gradient. This is the template for *stochastic gradient descent*, which is described in detail blow.

Algorithm 1 Stochastic Gradient Descent

```
// Return: ŵ, an intended approximation to
// w* ∈ arg min_{w∈ℝ^p} F(w), where F = (1/n) Σ_{i=1}^n f_i
procedure Stochastic Gradient Descent
    Initialize point w^(0). Prescribed step-size schedule {η_t}_{t=1}^∞
    for t := 1 to T − 1 do
        Draw an index i_t uniformly at random from {1, 2, ..., n}
        Iterate w^(t+1) ← w^(t) − η_t ∇f_{i_t}(w^(t)).
    end for
    return ŵ = w^(T)
end procedure
```

It is important that the index i_t is chosen uniformly at random, in which case the random vector $\nabla f_{i_t}(w^{(t)})$ is an unbiased estimate for the full gradient $\nabla F(w^{(t)})$, meaning that $\mathbb{E}_{i_t}\nabla f_{i_t}(w^{(t)}) = \nabla F(w^{(t)})$. In practice, one often implements *minibatch* stochastic gradient descent, which is a compromise between full gradient descent and stochastic gradient descent where a batch of component gradient directions are averaged at each step, to

1 We assume $n \in \mathbb{N}$ is a finite number for simplicity and because it is most relevant for applications, but all results can be extended in theory to continuous parameterizations $F(w) = \int_s f_s(w)d\mu(s)$.

reduce the variance of the stochastic gradient estimate. For a comprehensive overview of stochastic gradient descent methods in large-scale optimization, we refer the reader to the comprehensive article [4].

1.1. Stochastic gradient descent: Background

1.1.1. Stochastic approximation

The idea for stochastic gradient descent appeared almost 70 years ago in a paper by Robbins and Monro [25]. Suppose that $M : \mathbb{R} \to \mathbb{R}$ is a function with a unique root we wish to identify. We do not have access to exact exact evaluations $M(w)$, but rather we can access at a given point w a random variable $N(w)$ such that $\mathbb{E}[N(w)] = M(w)$. Within this framework of *stochastic approximation*, Robbins and Monro proposed the following root-finding algorithm: fix w_0 and a decreasing step-size schedule $\{a_n\}_{n=0}^{\infty}$, then iterate

$$w_{n+1} = w_n - a_n N(w_n). \tag{1.4}$$

Blum [3] subsequently extended the algorithm to the multivariate setting. One recognizes the SGD Algorithm 1 as a special case of the Robbins–Monro root-finding algorithm via the correspondence $M(w) = \nabla F(w)$ and $N(w^{(t)}) = \nabla f_{i_t}(w^{(t)})$. Under certain assumptions akin to strong convexity, smoothness, and bounded noise, Robbins, Monro, and Blum showed that algorithm (1.4) converges with probability 1, provided the step-size schedule $\{a_n\}_{n=0}^{\infty}$ is chosen to decrease at a rate such that

$$\sum_{n=0}^{\infty} a_n = \infty \quad \text{and} \quad \sum_{n=0}^{\infty} a_n^2 < \infty. \tag{1.5}$$

An important gap between the setting considered by Robbins and Monro and the application of stochastic gradient descent in large scale machine learning is in the model assumptions for the stochastic noise: Robbins–Monro convergence theory and the implied choice of step-sizes (1.5) assume that the stochastic noise on the observations is *uniformly bounded*, $\sup_\theta |N(\theta)| \leq N < \infty$. More realistic in the setting of large-scale machine learning is an *affine-variance* noise model, where the stochastic noise level is proportional to the size of the full gradient at any given point. Specifically, the affine-variance noise model is as follows: for parameters $\sigma_0, \sigma_1 > 0$,

$$\forall w \in \mathbb{R}^p : \mathbb{E}_i \left\| \nabla f_i(w) \right\|_2^2 \leq \sigma_0^2 + \sigma_1^2 \left\| \nabla F(w) \right\|^2. \tag{1.6}$$

We will come back to the discussion about the stochastic noise later on.

Stochastic gradient descent was recognized as a powerful algorithm for training artificial neural networks in 1960, when it was used to train one of the earliest neural networks—the Adaline network (Adaline stands for "adaptive linear unit") [29]. The proposal of Adaline came shortly after Rosenblatt invented the perceptron, widely considered the first artificial neural network. Following Adaline, (stochastic) gradient descent persisted as the de facto algorithm for training artificial neural networks due to its simplicity and ability to extend (using back propagation) to multilayered neural network architectures. The full power of artificial neural networks was not realized until around 2010, when increased computing power

from GPUs and distributed computing allowed the use of larger networks, which became known as "deep learning," and neural networks began winning prizes in image recognition contests, approaching human level performance on various tasks. Just in time for the advent of modern deep learning, researchers began to formalize the nonasymptotic theory for stochastic gradient descent—convergence to global minimizers in the convex setting [2,23] and to stationary points the nonconvex setting [9].

1.1.2. The Kaczmarz method for solving linear systems

Independent of Robbins and Monro's work in stochastic approximation, Stefan Kaczmarz proposed an iterative method (now called the Kaczmarz method) for solving linear equations [15]. Consider an overdetermined system of consistent linear equations, $Aw = b$. Denote the ith row (out of a total of n rows) of A by a_i, and let $w^{(0)}$ be an arbitrary initial approximation to the solution of $Aw = b$. Kaczmarz observed that w^*, the unique solution to the overdetermined consistent system, corresponds to the unique point in the intersection of the hyperplanes $S_i = \{w : \langle a_i, w\rangle = b_i\}$. He proposed an iterative projection algorithm for finding the point w^* whereby one cycles through the hyperplanes in their natural ordering, and projects the current estimate for w^* onto the subsequent subspace, until convergence. That is, starting from an initial guess w^0 and for $t = 1, 2, \ldots$, iterate

$$w^{(t+1)} = w^{(t)} + \frac{b_i - \langle a_i, w^{(t)}\rangle}{\|a_i\|^2} a_i; \quad i = t \bmod n. \tag{1.7}$$

The Kaczmarz method can be viewed as an instance of what is now referred to as the method of successive projections onto convex sets (POCS). In 1933, John von Neumann proved convergence of POCS in the case of two ($n = 2$) hyperplanes [27]; Halperin later extended von Neumann's convergence result to arbitrarily many hyperplanes [11]. Aronszajn [1] later provided an explicit rate of convergence for the case of two hyperplanes—the convergence rate is linear and depends explicitly on the angle between the two hyperplanes. Kayalar and Weinert [17] proved that Aronszajn's rate of convergence is sharp. This sharp analysis has proved difficult to extend beyond the case of two hyperplanes, as it is related to the difficulty of analyzing the product of more than two orthogonal projection operators, see, for example, [5]. The Kaczmarz method was rediscovered in image reconstruction in 1970, where (along with additional positivity constraints) it is called the algebraic reconstruction technique (ART) [10]. ART is used extensively in computed tomography and, in fact, was used in the first medical scanner [13].

Later, in the 1990s, several works, including [8,12], observed that the Kaczmarz algorithm (1.7) tended to converge more quickly and consistently if the algorithm was changed so that the rows are selected in a *random*, rather than cyclic order. In a seminal paper, Strohmer and Vershynin proved in 2007 that if the rows are drawn from a particular weighted random distribution, the Kaczmarz algorithm converges in expectation with a sharp linear convergence rate [26] depending on a condition number of the matrix A. Precisely, the randomized Kaczmarz method proposed by Strohmer and Vershynin is as follows (Algorithm 2):

Algorithm 2 Randomized Kaczmarz method

1: // Return: $\hat{w}$, an intended solution to $Aw = b$
2: **procedure** Randomized Kaczmarz algorithm
3: Initialize point $w^{(0)}$. Denote rows of A by $\{a_i\}_{i=1}^n$.
4: **for** $t := 1$ **to** $T - 1$ **do**
5: Draw a row a_{i_t}, where i_t is chosen from the set $\{1, 2, \ldots, n\}$ at random according to a weighted probability distribution such that $\text{Prob}(i_t = j) \propto \|a_j\|_2^2$.
6: Iterate $w^{(t+1)} \leftarrow w^{(t)} + \frac{b_{i_t} - \langle a_{i_t}, w^{(t)} \rangle}{\|a_{i_t}\|^2} a_{i_t}$.
7: **end for**
8: **return** $\hat{w} = w^{(T)}$
9: **end procedure**

Subsequently, the paper [22] recognized the randomized Kaczmarz method as a special case of stochastic gradient descent Algorithm 1 applied in the setting of linear regression, where the objective function is $F(w) = \|Aw - b\|_2^2 = \frac{1}{n} \sum_{i=1}^n n(\langle a_i, w \rangle - b_i)^2$, and implemented with *importance sampling* so that $\text{Prob}(i_t = j) \propto \|a_j\|_2^2$ and $\eta_{i_t} = \frac{\gamma}{\|a_{i_t}\|_2^2}$ to maintain unbiasedness of the stochastic gradient estimator for the full gradient. Extending Strohmer and Vershynin's analysis beyond the linear regression setting improves on a previous linear convergence rate of Bach and Moulines [2] to show that stochastic gradient descent Algorithm 1 enjoys a linear convergence rate under a general set of conditions including convexity and smoothness. Moreover, the convergence rate can be improved when component functions are allowed to be drawn from an importance sampling weighted distribution, as extended to neural networks in [16, 20].

1.2. Stochastic gradient descent: convergence theory

In this section, we will lay out the convergence theory for stochastic gradient descent precisely. Enforce the following conditions on a loss function of the form $F(w) = \frac{1}{n} \sum_{i=1}^n f_i(w)$:

(a) each f_i is L-smooth: $\forall w, z, \|\nabla f_i(w) - \nabla f_i(z)\|_2 \leq L\|w - z\|_2$;

(b) each f_i is convex;

(c) F is μ-strongly convex.

Under these assumptions, the loss function F has a unique minimizer w^*, and SGD converges to this minimizer as follows [22].

Theorem 1. *Consider constant step-size $\eta \leq \frac{1}{\mu}$. Draw $w^{(0)}$ either as a random initial point or deterministically. Denote $\sigma^2 = \frac{1}{n} \sum_{i=1}^n \|\nabla f_i(w^*)\|_2^2$. Under the stated assumptions, the expected error of the SGD Algorithm 1 satisfies*

$$\mathbb{E}\left\|w^{(t)} - w^*\right\|_2^2 \leq \left[1 - 2\eta\mu(1 - \eta L)\right]^t \mathbb{E}\left\|w^{(0)} - w^*\right\|_2^2 + \frac{\eta\sigma^2}{\mu(1 - \eta L)}.$$

The error expression has two terms, highlighting that the algorithm enjoys a linear convergence rate up to a so-called "region of confusion" of radius $\frac{\eta\sigma^2}{\mu(1-\eta L)}$. Optimizing the step-size η to balance the two error terms results in the following sharp convergence rate.

Corollary 1.1. *Enforce the assumptions of Theorem 1. Fix the constant step-size* $\eta = \frac{\mu\epsilon}{2\epsilon\mu L + 2\sigma^2}$. *Denote by* $\epsilon_0 = \mathbb{E}_{w^{(0)}}\|w^{(0)} - w^*\|_2^2$. *After* $T = 2\log(\epsilon_0/\epsilon)(\frac{L}{\mu} + \frac{\sigma^2}{\mu^2\epsilon})$ *iterations, the expected error satisfies*

$$\mathbb{E}\left\|w^{(T)} - w^*\right\|_2^2 \leq \epsilon.$$

While this convergence rate cannot really be improved in the setting where the step-size is fixed, we can improve on this rate slightly by considering a carefully chosen piecewise constant decreasing step-size schedule and applying Corollary (1.1) recursively. We could not find the following result stated explicitly in the literature, so we provide the short proof.

Proposition 1.1. *Enforce the assumptions on smoothness and convexity from Theorem 1. For error function* $h(s) = \frac{\mu s}{2s\mu L + 2\sigma^2}$ *and times*

$$T_J = 2\left(\frac{L}{\mu} + \frac{2^J\sigma^2}{\mu^2\mathbb{E}\|w^{(0)} - w^*\|_2^2}\right), \quad J = 1, 2, \ldots$$

consider the SGD Algorithm 1 with piecewise constant decreasing step-size schedule

$$\eta_t = \eta_1 := h(\epsilon_0 \cdot 2^{-1}), \quad 1 \leq t \leq T_1,$$
$$\eta_t = \eta_J := h(\epsilon_0 \cdot 2^{-J}), \quad 1 + \sum_{j=1}^{J-1} T_j \leq t \leq \sum_{j=1}^{J} T_j,$$

After $T_{(K)} = 2K(\frac{L}{\mu} + \frac{2}{K} \cdot \frac{\sigma^2}{\mu^2\mathbb{E}\|w^{(0)} - w^*\|_2^2}2^K)$ *iterations,*

$$\mathbb{E}\left\|w^{(T_{(K)})} - w^*\right\|_2^2 \leq 2^{-K}\mathbb{E}\left\|w^{(0)} - w^*\right\|_2^2.$$

Comparing the error bounds in Proposition 1.1 and Corollary 1.1, we see that to achieve error $\mathbb{E}\|w^{(T_{(K)})} - w^*\|_2^2 \leq 2^{-K}\mathbb{E}\|w^{(0)} - w^*\|_2^2$, the piecewise constant decreasing step-size schedule requires a number of iterations $T = 2K(\frac{L}{\mu} + \frac{2}{K} \cdot \frac{\sigma^2}{\mu^2\epsilon_0}2^K)$ while the constant step-size schedule requires a larger number of iterations $T' = 2K(\frac{L}{\mu} + \frac{\sigma^2}{\mu^2\epsilon_0}2^K)$. This suggests that to get the best possible convergence rate of SGD when the region of confusion dominates the condition number, piecewise constant decreasing step-size schedules can outperform constant step-size schedules.

Proof of Proposition 1.1. Theorem 1.1 is proved by induction on the bound in Corollary 1.1 with the number of levels K. For the base case $K = 1$, we get the result by applying Corollary 1.1 with $\epsilon_1 = \epsilon_0/2$ and fixed step-size $\eta_1 = h(\epsilon_0/2)$. For the induction, suppose the result holds at $K - 1$, that is, suppose that $\mathbb{E}\|w^{(T_{(K-1)})} - w^*\|_2^2 \leq \epsilon_{K-1}$, where $\epsilon_K := 2^{-K}\mathbb{E}\|w^{(0)} - w^*\|_2^2$. Apply Corollary 1.1 with $\epsilon_0 = \epsilon_{K-1}$ and $\epsilon = \epsilon_0/2$ and with $\eta_K = h(\epsilon_0 \cdot 2^{-K})$ to arrive at the stated bound at K. ∎

We draw the reader's attention to the fact that we have focused on stochastic gradient descent convergence theory under assumptions such as smoothness and convexity which

are not satisfied in the setting of training neural networks. However, increasingly, neural networks are implemented to be highly *overparameterized*, or configured so that the number of parameters p (length of the parameter vector w) is set to be larger than the size of the training data. In this regime, recent works have shown that in a certain "neural tangent kernel regime," the loss function associated to training overparameterized neural networks is locally strongly convex around a random initialization $w^{(0)}$ [6,14]. While it is an active area of research to try and understand the extent to which overparameterized neural networks remain similar to linear systems in regimes where neural networks are most powerful in practice, there is evidence that points to a strong connection. One important piece of evidence is the fact that SGD is typically trained using piecewise constant decreasing step-sizes to optimize convergence speed, just as suggested by Proposition 1.1. Thus, the convergence theory for SGD in the strongly convex setting (and the corresponding step-size schedule which results for optimizing convergence) is surprisingly relevant in the application of training large-scale neural networks.

1.3. Adaptive step-size rules in stochastic gradient descent

Proposition 1.1 suggests that in training neural networks using stochastic gradient descent, piecewise constant decreasing step-sizes should be effective. In practice, neural networks are indeed trained using piecewise constant decreasing step-size schedules; however, the particular choice of step-size schedule in Proposition 1.1 is not so useful in practice as it is a function of several parameters of the optimization problem: the strong convexity parameter $\mu > 0$, the Lipschitz smoothness constant L associated to the loss function, the stochastic noise level $\sigma^2 > 0$, and the error at initialization $\|w^{(0)} - w^*\|_2^2$. In practice, none of these quantities is known to the user in advance. Indeed, this represents a serious disconnect between the theory for SGD and the practical implementation, as the convergence behavior of the basic SGD Algorithm 1 is quite sensitive to the choice of step-size schedule. Fortunately, simple modifications to the basic SGD algorithm have been developed, such as Adagrad [7,21], RMSprop, and Adam [18], which are significantly more robust to the step-size schedule. A convergence theory for these algorithms as adaptive step-size learners in the setting of stochastic gradient descent was initiated independently in [19,28]. We will focus on the results from [28], which focuses on guarantees for the AdaGrad adaptive gradient algorithm.

As a precursor to discussing adaptive gradient methods in the context of stochastic gradient descent, let us first understand their behavior in the setting of batch (full) gradient descent (where where the gradients $\nabla F(w)$ are measured exactly).[2] In the batch setting, the AdaGrad algorithm is as follows.

2 We note that in the batch setting, line search methods are efficient black-box plugins for adaptively updating the step-size. However, such methods lose effectiveness in the presence of stochastic noise, and have a tendency to overfit the noisy gradient directions.

Algorithm 3 Gradient Descent with AdaGrad

// Return: $\hat{w}$, an intended approximation to
// $w^* \in \arg\min_{w \in \mathbb{R}^p} F(w)$
procedure Gradient Descent with AdaGrad
    Initialize point $w^{(0)}$, initial step-size parameters $b_0, \eta > 0$. Tolerance $\epsilon > 0$.
    repeat
        $t + 1 \leftarrow t$.
        Update step-size $b_t^2 = b_{t-1}^2 + \|\nabla F(w^{(t-1)})\|^2$

        Iterate $w^{(t)} \leftarrow w^{(t-1)} - \frac{\eta}{b_t} \nabla F(w^{(t-1)})$.
    until $\|\nabla F(w^{(t)})\|^2 \leq \epsilon$
    return $\hat{w} = w^{(T)}$
end procedure

To put our main result in context, let us first review the following classical result (see, for example, [24, (1.2.13)]) on the convergence rate for gradient descent with fixed step-size.

Lemma 1.1. *Suppose that F is L-smooth, and suppose that $F^* = \inf_x F(x) > -\infty$. Fix η and b, consider gradient descent, $w^{(t+1)} = w^{(t)} - \frac{\eta}{b} \nabla F(w^{(t)})$. If $b \geq \eta L$, then*

$$\min_{t=0:T-1} \left\| \nabla F(w^{(t)}) \right\|^2 \leq \varepsilon$$

after at most a number of steps

$$T = \frac{2b(F(w^{(0)}) - F^*)}{\eta \varepsilon}.$$

Alternatively, if $b \leq \frac{\eta L}{2}$, then convergence is not guaranteed at all—gradient descent can oscillate or diverge.

The following result on the convergence of AdaGrad Algorithm 3 from [28] shows that in contrast to fixed step-size gradient descent, AdaGrad always converges, and its convergence rate as a function of the parameters $b_0, \eta > 0$ can be understood in a sharp sense. It suggests that in practice, one should simply initialize AdaGrad with a large step-size $1/b_0$, and the algorithm will adapt on its own by decreasing the step-size to an appropriate limiting value.

Theorem 2 (AdaGrad—convergence). *Consider the AdaGrad Algorithm 3. Suppose that F is L-smooth and suppose that $F^* = \inf_w F(w) > -\infty$. Then*

$$\min_{t=0:T-1} \left\| \nabla F(w^{(t)}) \right\|^2 \leq \varepsilon$$

after ***Case 1:*** $T = 1 + \lceil \frac{2(F(w^{(0)})-F^*)(b_0+2(F(w^{(0)})-F^*)/\eta)}{\eta\varepsilon} \rceil$ *steps if* $\frac{b_0}{\eta} \geq L$, *and*

Case 2: $T = 1 + \lceil \frac{(\eta L)^2-b_0^2}{\varepsilon} + \frac{4((F(w^{(0)})-F^*)/\eta+(\frac{3}{4}+\log\frac{\eta L}{b_0})\eta L)^2}{\varepsilon} \rceil$ *steps if* $\frac{b_0}{\eta} < L$.

In either case, $\max_t \frac{b_t}{\eta} \leq \frac{b_{\max}}{\eta}$ *where* $b_{\max}/\eta = 2L(1+\log(\eta L/b_0)) + \frac{2}{\eta^2}(F(w^{(0)}) - F^*)$.

Comparing the convergence rate of AdaGrad with the convergence rate of gradient descent with fixed step-size, we see that in case $b = b_0 \geq \eta L$, the rates are essentially the same. But in case $b = b_0 < \eta L$, gradient descent can fail to converge as soon as $b \leq \eta L/2$, while AdaGrad converges for any $b_0 > 0$, and is extremely robust to the choice of $b_0 < \eta L$ in the sense that the resulting convergence rate remains close to the optimal rate of gradient descent with fixed step-size $\eta/b = 1/L$, paying only a factor of $\log(\frac{\eta L}{b_0})$ in the constant.

The convergence rate in Theorem 2 represents a worst-case analysis of AdaGrad over the class of L-smooth functions. In practice, the limiting step-size will obtain very quickly, and at a value much *larger* than $1/L$. This is not surprising since the smoothness parameter L represents only the globally worst-case bound on the magnitude of the ratio $\frac{\|\nabla F(w)-\nabla F(z)\|}{\|w-z\|}$ over all $w, z \in \mathbb{R}^p$. In other words, even if one has a priori bound on L, AdaGrad can converge significantly faster than gradient descent with fixed step-size $1/L$, and is thus advantageous to use even with such knowledge.

Now let us turn to the convergence analysis of AdaGrad in the stochastic setting, also from [28]. Recall that in the stochastic setting, instead of observing a full gradient at each iteration, we observe a stochastic gradient $g_t \in \mathbb{R}^p$ which is an unbiased estimator for the true gradient $\nabla F(w^{(t)})$.

We now state Adagrad in the stochastic setting (Algorithm 4):

Algorithm 4 Stochastic Gradient Descent with AdaGrad

1: `//Return` w^*, `an approximation to a stationary point of a smooth function` $F(\cdot)$ `over` $\mathbb{R}^p$.
2: **procedure** AdaGrad in Stochastic Setting
3: Initial point $w^{(0)} \in \mathbb{R}^p$. step-size parameters η, b_0.
4: **for** $t := 1$ **to** $T - 1$ **do**
5: step-size update:

$$\eta_t = \frac{\eta}{\sqrt{b_{t-1}^2 + \|g_t\|^2}} \quad \text{where } b_t^2 = b_{t-1}^2 + \|g_t\|^2$$

6: Iterate $w^{(t+1)} \leftarrow w^{(t)} - \eta_t g_t$.
7: **end for**
8: **return** $\hat{w} = w^{(T)}$
9: **end procedure**

More formally, denote

$$\mathcal{F}_t = \sigma\left\{w^1, g^1, \ldots, w^{(t)}, g^{(t)}, w^{(t+1)}\right\} \tag{1.8}$$

to be the sigma algebra generated by the observations of the algorithm after observing the first t stochastic gradients. We will assume that the stochastic gradients satisfy the following:

Assumptions 2.1 (Unbiased gradients). *For each time t, the stochastic gradient, g_t, is an unbiased estimate of $\nabla F(w^{(t)})$, i.e.,*

$$\mathbb{E}[g_t|\mathcal{F}_{t-1}] = \nabla F(w^{(t)}). \tag{1.9}$$

For the theory in this section, we will assume that the stochastic noise is uniformly bounded, as in the setting of Robbins and Monro.[3]

Assumptions 2.2 (Uniformly bounded gradient and uniformly bounded variance). *We assume $\sup_{w\in\mathbb{R}^p} \|\nabla F(w)\| \leq \gamma$. Moreover, for each time t, the variance satisfies*

$$\mathbb{E}\left[\left\|g_t - \nabla F(w^{(t)})\right\|^2 |\mathcal{F}_{t-1}\right] \leq \sigma^2. \tag{1.11}$$

The AdaGrad step-sizes $\frac{\eta}{b_t}$ in the stochastic setting exhibit quite different behavior than in the deterministic setting. Rather than converging to a fixed value proportional to the Lipschitz smoothness constant as in the batch setting, the step-size decreases to zero in the stochastic setting, roughly at the rate of $\frac{1}{b_t} \approx \frac{1}{\sigma\sqrt{t}}$. This rate is optimal in t in terms of the resulting convergence theorems in the setting of smooth but not necessarily convex F, or convex but not necessarily strongly convex or smooth F. Still, one must be careful with convergence theorems for AdaGrad because the step-size is a random variable and dependent on all previous points visited along the way.

Theorem 3. *Suppose F is L-smooth and $F^* = \inf_w F(w) > -\infty$. Suppose that the random variables $g_t, t \geq 0$, satisfy the above assumptions. Then with probability $1-\delta$,*

$$\min_{t\in[T-1]} \left\|\nabla F(w^{(t)})\right\|^2 \leq \left(\frac{2b_0}{T} + \frac{2\sqrt{2}(\gamma+\sigma)}{\sqrt{T}}\right)\frac{Q}{\delta^{3/2}}$$

where $Q = \frac{F(w^{(0)})-F^*}{\eta} + \frac{4\sigma+\eta L}{2}\log(\frac{20T(\gamma^2+\sigma^2)}{b_0^2} + 10)$.

This result implies that AdaGrad converges starting from any value of b_0 for a given η. To put this result in context, we can compare to Corollary 2.2 of [9], which implies that under similar assumptions, if the Lipschitz constant L and the variance σ are known a priori, and the step-size is

$$\eta_t = \eta = \min\left\{\frac{1}{L}, \frac{1}{\sigma\sqrt{T}}\right\}, \quad t = 0, 1, \ldots, T-1,$$

---

3 These assumptions can be weakened to a single affine-variance assumption: For each time t, the variance only needs to satisfy

$$\mathbb{E}\left[\left\|g_t - \nabla F(w^{(t)})\right\|^2 |\mathcal{F}_{t-1}\right] \leq \sigma_0^2 + \sigma_1^2\|\nabla F(w^{(t)}\|^2. \tag{1.10}$$

then with probability $1-\delta$,

$$\min_{\ell\in[T-1]} \left\|\nabla F(w^{(t)})\right\|^2 \leq \frac{2L(F(w^{(0)})-F^*)}{T\delta} + \frac{(L+2(F(w^{(0)})-F^*))\sigma}{\delta\sqrt{T}}.$$

Thus, essentially, AdaGrad convergence achieves the rate of [9], but without requiring a priori knowledge of L and σ to set the step-sizes. The constant in the $O(1/\sqrt{T})$ rate of AdaGrad scales according to σ^2 (up to a logarithmic factors in σ) while the results with well-tuned step-size scales linearly with σ.

The main technical difficulty in the proof of Theorem 3 is in dealing with the AdaGrad step-sizes which are random variables which depend on the current and all previous stochastic gradients. See [28] for details of the proof.

1.4. Outlook

Stochastic gradient descent is the de facto algorithm used for minimizing functions which arise in deep learning and neural network training. While there are many mysteries surrounding the behavior of stochastic gradient descent in applications, there are also several regimes in which we have a rich and sharp mathematical understanding. Remarkably, the strong linear convergence guarantees for stochastic gradient descent which are guaranteed in the setting of strongly convex finite sums (and the piecewise constant decreasing step-size rules they imply) are empirically verified in practice in training overparameterized neural networks. That is, in many practical settings, seemingly highly nonconvex and highly nonlinear neural network-based regression functions of interest are in reality perturbations of linear regression problems. In this sense, practice caught up with theory. A theoretical understanding of practical methods for making stochastic gradient descent more robust to hyperparameter specifications such as the step-size schedule has begun to emerge in recent years. In this sense, stochastic gradient enhancements developed in practice to meet the needs of large-scale machine learning inspired new theoretical directions in the study of stochastic gradient descent.

FUNDING

This work was partially supported by AFOSR MURI FA9550-19-1-0005, NSF DMS 1952735, NSF HDR-1934932, and NSF 2019844.

REFERENCES

[1] L. Aronszajn, Theory of reproducing kernels. *Trans. Amer. Math. Soc.* **68** (1950), 337–404.

[2] F. Bach and E. Moulines, Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in neural information processing systems. Vol. 24*, 2011.

[3] J. R. Blum, Approximation methods which converge with probability one. *Ann. Math. Stat.* (1954), 382–386.

[4] L. Bottou, F. E. Curtis, and J. Nocedal, Optimization methods for large-scale machine learning. *SIAM Rev.* **60** (2018), no. 2, 223–311.

[5] F. Deutsch, The rate of convergence for the method of alternating projections, II. *J. Math. Anal. Appl.* **205** (1997), 381–405.

[6] S. S. Du, X. Zhai, B. Poczos, and A. Singh, Gradient descent provably optimizes over-parameterized neural networks. In *International conference on learning representations*, 2018.

[7] J. Duchi, E. Hazan, and Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **12** (2011), 2121–2159.

[8] H. G. Feichtinger, C. Cenker, M. Mayer, H. Steier, and T. Strohmer, New variants of the POCS method using affine subspaces of finite codimension with applications to irregular sampling. In *Visual Communications and Image Processing'92*, pp. 299–310, Proc. SPIE 1818, International Society for Optics and Photonics, 1992.

[9] S. Ghadimi and G. Lan, Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM J. Optim.* **23** (2013), no. 4, 2341–2368.

[10] R. Gordon, R. Bender, and G. T. Herman, Algebraic reconstruction techniques (art) for three-dimensional electron microscopy and X-ray photography. *J. Theoret. Biol.* **29** (1970), no. 3, 471–481.

[11] I. Halperin, The product of projection operators. *Acta Sci. Math. (Szeged)* **23** (1962), 96–99.

[12] G. T. Herman and L. B. Meyer, Algebraic reconstruction techniques can be made computationally efficient (positron emission tomography application). *IEEE Trans. Med. Imag.* **12** (1993), no. 3, 600–609.

[13] G. N. Hounsfield, Computerized transverse axial scanning (tomography): Part 1. description of system. *Br. J. Radiol.* **46** (1973), no. 552, 1016–1022.

[14] A. Jacot, F. Gabriel, and C. Hongler, Neural tangent kernel: convergence and generalization in neural networks. In *Proceedings of the 53rd annual ACM SIGACT symposium on theory of computing*, p. 6, ACM, 2021.

[15] S. Karczmarz, Angenaherte Auflosung von Systemen linearer Gleichungen. *Bull. Int. Acad. Pol. Sic. Let., Cl. Sci. Math. Nat.* (1937), 355–357.

[16] A. Katharopoulos and F. Fleuret, Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pp. 2525–2534, PMLR, 2018.

[17] S. Kayalar and H. Weinert, Error bounds for the method of alternating projections. *Math. Control Signals Systems* **1** (1988), 43–59.

[18] D. Kingma and J. Ba. Adam, A method for stochastic optimization. 2014, arXiv:1412.6980.

[19] X. Li and F. Orabona, On the convergence of stochastic gradient descent with adaptive stepsizes. 2018, arXiv:1805.08114.

[20] I. Loshchilov and F. Hutter, Online batch selection for faster training of neural networks. 2015, arXiv:1511.06343.

[21] H. B. McMahan and M. Streeter, Adaptive bound optimization for online convex optimization. In *COLT 2010*, p. 244, 2010.

[22] D. Needell, R. Ward, and N. Srebro, Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm. In *Advances in neural information processing systems*, 2014.

[23] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, Robust stochastic approximation approach to stochastic programming. *SIAM J. Optim.* **19** (2009), no. 4, 1574–1609.

[24] Y. Nesterov, *Introductory lectures on convex programming volume I: Basic course*. 1998.

[25] H. Robbins and S. Monro, A stochastic approximation method. *Ann. Math. Stat.* **22** (1951), 400–407.

[26] T. Strohmer and R. Vershynin, A randomized Kaczmarz algorithm with exponential convergence. *J. Fourier Anal. Appl.* **15** (2009), no. 2, 262–278.

[27] J. von Neumann, Functional operators. vol. II. The geometry of orthogonal spaces. *Ann. of Math. Stud.* **22** (1950). This is a reprint of mimeographed lecture notes first distributed in 1933.

[28] R. Ward, X. Wu, and L. Bottou, AdaGrad stepsizes: sharp convergence over nonconvex landscapes. In *International conference on machine learning*, pp. 6677–6686, PMLR, 2019.

[29] B. Widrow, An adaptive "Adaline" neuron using chemical "memistors". Technical report No. 1553-2, 1960.

RACHEL WARD

2515 Speedway, Austin, TX 78712, USA, rward@math.utexas.edu