Adaptive Gradient Descent without Descent

Yura Malitsky [1] **Konstantin Mishchenko** [2]

Abstract

We present a strikingly simple proof that two rules are sufficient to automate gradient descent: 1) don't increase the stepsize too fast and 2) don't overstep the local curvature. No need for functional values, no line search, no information about the function except for the gradients. By following these rules, you get a method adaptive to the local geometry, with convergence guarantees depending only on the smoothness in a neighborhood of a solution. Given that the problem is convex, our method converges even if the global smoothness constant is infinity. As an illustration, it can minimize arbitrary continuously twice-differentiable convex function. We examine its performance on a range of convex and nonconvex problems, including logistic regression and matrix factorization.

1. Introduction

Since the early days of optimization it was evident that there is a need for algorithms that are as independent from the user as possible. First-order methods have proven to be versatile and efficient in a wide range of applications, but one drawback has been present all that time: the stepsize. Despite certain success stories, line search procedures and adaptive online methods have not removed the need to manually tune the optimization parameters. Even in smooth convex optimization, which is often believed to be much simpler than the nonconvex counterpart, robust rules for stepsize selection have been elusive. The purpose of this work is to remedy this deficiency.

The problem formulation that we consider is the basic unconstrained optimization problem

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

where $f\colon \mathbb{R}^d \to \mathbb{R}$ is a differentiable function. Throughout the paper we assume that (1) has a solution and we denote its optimal value by f_*.

The simplest and most known approach to this problem is the gradient descent method (GD), whose origin can be traced back to Cauchy (Cauchy, 1847; Lemaréchal, 2012). Although it is probably the oldest optimization method, it continues to play a central role in modern algorithmic theory and applications. Its definition can be written in a mere one line,

$$x^{k+1} = x^k - \lambda\nabla f(x^k), \qquad k \geq 0, \tag{2}$$

where $x^0 \in \mathbb{R}^d$ is arbitrary and $\lambda > 0$. Under assumptions that f is convex and L–smooth[1], that is

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall x, y, \tag{3}$$

one can show that GD with $\lambda \in (0, \frac{2}{L})$ converges to an optimal solution (Polyak, 1963). Moreover, with $\lambda = \frac{1}{L}$ the convergence rate (Drori & Teboulle, 2014) is

$$f(x^k) - f_* \leq \frac{L\|x^0 - x^*\|^2}{2(2k+1)}, \tag{4}$$

where x^* is any solution of (1). Note that this bound is not improvable (Drori & Teboulle, 2014).

We identify four important challenges that limit the applications of gradient descent even in the convex case:

1. GD is not general: many functions do not satisfy (3) globally.
2. GD is not a free lunch: one needs to guess λ, potentially trying many values before a success.
3. GD is not robust: failing to provide $\lambda < \frac{2}{L}$ may lead to divergence.
4. GD is slow: even if L is finite, it might be arbitrarily larger than local smoothness.

1.1. Related work

Certain ways to address some of the issues above already exist in the literature. They include line search, adaptive Polyak's stepsize, mirror descent, dual preconditioning, and

*Equal contribution [1]EPFL, Lausanne, Switzerland [2]KAUST, Thuwal, Saudi Arabia. Correspondence to: Yura Malitsky <y.malitsky@gmail.com>, Konstantin Mishchenko <konsta.mish@gmail.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

[1]Alternatively, we will say that ∇f is L-Lipschitz.

stepsize estimation for subgradient methods. We discuss them one by one below, in a process reminiscent of cutting off Hydra's limbs: if one issue is fixed, two others take its place.

The most practical and generic solution to the aforementioned issues is known as line search (or backtracking). This direction of research started from the seminal works (Goldstein, 1962) and (Armijo) and continues to attract attention, see (Bello Cruz & Nghia, 2016; Salzo, 2017) and references therein. In general, at each iteration the line search executes another subroutine with additional evaluations of ∇f and/or f until some condition is met. Obviously, this makes each iteration more expensive.

At the same time, the famous Polyak's stepsize (Polyak, 1969) stands out as a very fast alternative to gradient descent. Furthermore, it does not depend on the global smoothness constant and uses the current gradient to estimate the geometry. The formula might look deceitfully simple, $\lambda_k = \frac{f(x^k)-f_*}{\|\nabla f(x^k)\|^2}$, but there is a catch: it is rarely possible to know f_*. This method, again, requires the user to guess f_*. What is more, with λ it was fine to underestimate it by a factor of 10, but the guess for f_* must be tight, otherwise it has to be reestimated later (Hazan & Kakade, 2019).

Seemingly no issue is present in the Barzilai-Borwein stepsize. Motivated by the quasi-Newton schemes, (Barzilai & Borwein, 1988) suggested using steps

$$\lambda_k = \frac{\langle x^k - x^{k-1}, \nabla f(x^k) - \nabla f(x^{k-1})\rangle}{\|\nabla f(x^k) - \nabla f(x^{k-1})\|^2}.$$

Alas, the convergence results regarding this choice of λ_k are very limited and the only known case where it provably works is quadratic problems (Raydan, 1993; Dai & Liao, 2002). In general it may not work even for smooth strongly convex functions, see the counterexample in (Burdakov et al., 2019).

Other more interesting ways to deal with non-Lipschitzness of ∇f use the problem structure. One such method, proposed in (Birnbaum et al., 2011) and further developed in (Bauschke et al., 2016), shows that the mirror descent method (Nemirovsky & Yudin, 1983), which is another extension of GD, can be used with a fixed stepsize, whenever f satisfies a certain generalization of (3). In addition, (Maddison et al., 2019) proposed the dual preconditioning method—another refined version of GD. Similarly to the former technique, it also goes beyond the standard smoothness assumption of f, but in a different way. Unfortunately, these two simple and elegant approaches cannot resolve all issues yet. First, not many functions fulfill respective generalized conditions. And secondly, both methods still get us back to the problem of not knowing the allowed range of stepsizes.

A whole branch of optimization considers adaptive extensions of GD that deal with functions whose (sub)gradients are bounded. Probably the earliest work in that direction was written by (Shor, 1962). He showed that the method

$$x^{k+1} = x^k - \lambda_k \frac{g^k}{\|g^k\|},$$

where $g^k \in \partial f(x^k)$ is a subgradient, converges for properly chosen sequences (λ_k), see, e.g., Section 3.2.3 in (Nesterov, 2013a). Moreover, λ_k requires no knowledge about the function whatsoever.

Similar methods that work in online setting such as Adagrad (Duchi et al., 2011; McMahan & Streeter, 2010) received a lot of attention in recent years and remain an active topic of research (Ward et al., 2019). Methods similar to Adagrad—Adam (Kingma & Ba, 2014; Reddi et al., 2018), RMSprop (Tieleman & Hinton, 2012) and Adadelta (Zeiler, 2012)—remain state-of-the-art for training neural networks. The corresponding objective is usually neither smooth nor convex, and the theory often assumes Lipschitzness of the function rather than of the gradients. Therefore, this direction of research is mostly orthogonal to ours, although we do compare with some of these methods in our neural networks experiment.

We also note that without momentum Adam and RMSprop reduce to signSGD (Bernstein et al., 2018), which is known to be non-convergent for arbitrary stepsizes on a simple quadratic problem (Karimireddy et al., 2019).

In a close relation to ours is the recent work (Malitsky, 2019), where there was proposed an adaptive golden ratio algorithm for monotone variational inequalities. As it solves a more general problem, it does not exploit the structure of (1) and, as most variational inequality methods, has a more conservative update. Although the method estimates the smoothness, it still requires an upper bound on the stepsize as input.

Contribution. We propose a new version of GD that at no cost resolves all aforementioned issues. The idea is simple, and it is surprising that it has not been yet discovered. In each iteration we choose λ_k as a certain approximation of the inverse local Lipschitz constant. With such a choice, we prove that convexity and local smoothness of f are sufficient for convergence of iterates with the complexity $\mathcal{O}(1/k)$ for $f(x^k) - f_*$ in the worst case.

Discussion. Let us now briefly discuss why we believe that proofs based on monotonicity and global smoothness lead to slower methods.

Gradient descent is by far not a recent method, so there have been obtained optimal rates of convergence. However, we argue that adaptive methods require rethinking optimality of the stepsizes. Take as an example a simple quadratic

problem, $f(x,y) = \frac{1}{2}x^2 + \frac{\delta}{2}y^2$, where $\delta \ll 1$. Clearly, the smoothness constant of this problem is equal to $L = 1$ and the strong convexity one is $\mu = \delta$. If we run GD from an arbitrary point (x^0, y^0) with the "optimal" stepsize $\lambda = \frac{1}{L} = 1$, then one iteration of GD gives us $(x^1, y^1) = (0, (1-\delta)y^0)$, and similarly $(x^k, y^k) = (0, (1-\delta)^k y^0)$. Evidently for δ small enough it will take a long time to converge to the solution $(0,0)$. Instead GD would converge in two iterations if it adjusts its step after the first iteration to $\lambda = \frac{1}{\delta}$.

Nevertheless, all existing analyses of the gradient descent with L-smooth f use stepsizes bounded by $2/L$. Besides, functional analysis gives

$$f(x^{k+1}) \leq f(x^k) - \lambda\Big(1 - \frac{\lambda L}{2}\Big)\|\nabla f(x^k)\|^2,$$

from which $1/L$ can be seen as the "optimal" stepsize. Alternatively, we can assume that f is μ-strongly convex, and the analysis in norms gives

$$\begin{aligned}\|x^{k+1} - x^*\|^2 \leq &\Big(1 - 2\frac{\lambda L\mu}{L+\mu}\Big)\|x^k - x^*\|^2 \\ &- \lambda\Big(\frac{2}{L+\mu} - \lambda\Big)\|\nabla f(x^k)\|^2,\end{aligned}$$

whence the "optimal" step is $\frac{2}{L+\mu}$.

Finally, line search procedures use some certain type of monotonicity, for instance ensuring that $f(x^{k+1}) \leq f(x^k) - c\|\nabla f(x^k)\|^2$ for some $c > 0$. We break with this tradition and merely ask for convergence in the end.

2. Main part

2.1. Local smoothness of f

Recall that a mapping is *locally Lipschitz* if it is Lipschitz over any compact set of its domain. A function f with (locally) Lipschitz gradient ∇f is called (locally) smooth. It is natural to ask whether some interesting functions are smooth locally, but not globally.

It turns out there is no shortage of examples, most prominently among highly nonlinear functions. In $\mathbb{R}$, they include $x \mapsto \exp(x)$, $\log(x)$, $\tan(x)$, x^p, for $p > 2$, etc. More generally, they include any twice differentiable f, since $\nabla^2 f(x)$, as a continuous mapping, is bounded over any bounded set $\mathcal{C}$. In this case, we have that ∇f is Lipschitz on $\mathcal{C}$, due to the mean value inequality

$$\|\nabla f(x) - \nabla f(y)\| \leq \max_{z\in\mathcal{C}} \|\nabla^2 f(z)\|\|x-y\|, \quad \forall x, y \in \mathcal{C}.$$

Algorithm 1 that we propose is just a slight modification of GD. The quick explanation why local Lipschitzness of ∇f does not cause us any problems, unlike most other methods, lies in the way we prove its convergence. Whenever the stepsize λ_k satisfies two inequalities[2]

$$\begin{cases}\lambda_k^2 & \leq (1+\theta_{k-1})\lambda_{k-1}^2, \\ \lambda_k & \leq \frac{\|x^k - x^{k-1}\|}{2\|\nabla f(x^k) - \nabla f(x^{k-1})\|},\end{cases}$$

independently of the properties of f (apart from convexity), we can show that the iterates (x^k) remain bounded. Here and everywhere else we use the convention $1/0 = +\infty$, so if $\nabla f(x^k) - \nabla f(x^{k-1}) = 0$, the second inequality can be ignored. In the first iteration it might happen that $\lambda_1 = \min\{+\infty\}$, in this case we suppose that any choice of $\lambda_1 > 0$ is possible.

Algorithm 1 Adaptive gradient descent

Input: $x^0 \in \mathbb{R}^d$, $\lambda_0 > 0$, $\theta_0 = +\infty$
$x^1 = x^0 - \lambda_0 \nabla f(x^0)$
for $k = 1, 2, \ldots$ **do**
  $\lambda_k = \min\Big\{\sqrt{1+\theta_{k-1}}\lambda_{k-1}, \frac{\|x^k - x^{k-1}\|}{2\|\nabla f(x^k) - \nabla f(x^{k-1})\|}\Big\}$
  $x^{k+1} = x^k - \lambda_k \nabla f(x^k)$
  $\theta_k = \frac{\lambda_k}{\lambda_{k-1}}$
end for

Although Algorithm 1 needs x^0 and λ_0 as input, this is not an issue as one can simply fix $x^0 = 0$ and $\lambda_0 = 10^{-10}$. Equipped with a tiny λ_0, we ensure that x^1 will be close enough to x^0 and likely will give a good estimate for λ_1. Otherwise, this has no influence on further steps.

2.2. Analysis without descent

It is now time to show our main contribution, the new analysis technique. The tools that we are going to use are the well-known Cauchy-Schwarz and convexity inequalities. In addition, our methods are related to potential functions (Taylor & Bach, 2019), which is a powerful tool for producing tight bounds for GD.

Another divergence from the common practice is that our main lemma includes not only x^{k+1} and x^k, but also x^{k-1}. This can be seen as a two-step analysis, while the majority of optimization methods have one-step bounds. However, as we want to adapt to the local geometry of our objective, it is rather natural to have two terms to capture the change in the gradients.

Now, it is time to derive a characteristic inequality for a specific Lyapunov energy.

Lemma 1. Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be convex and differential and

[2] It can be shown that instead of the second condition it is enough to ask for $\lambda_k^2 \leq \frac{\|x^k - x^{k-1}\|^2}{[3\|\nabla f(x^k)\|^2 - 4\langle \nabla f(x^k), \nabla f(x^{k-1})\rangle]_+}$, where $[a]_+ \stackrel{\text{def}}{=} \max\{0, a\}$, but we prefer the option written in the main text for its simplicity.

let x^* be any solution of (1). Then for (x^k) generated by Algorithm 1 it holds

$$\|x^{k+1}-x^*\|^2+\frac{1}{2}\|x^{k+1}-x^k\|^2+2\lambda_k(1+\theta_k)(f(x^k)-f_*) \leq \|x^k-x^*\|^2+\frac{1}{2}\|x^k-x^{k-1}\|^2+2\lambda_k\theta_k(f(x^{k-1})-f_*). \tag{5}$$

Proof. Let $k \geq 1$. We start from the standard way of analyzing GD:

$$\begin{aligned}\|x^{k+1}-x^*\|^2 &= \|x^k-x^*\|^2+2\left\langle x^{k+1}-x^k, x^k-x^*\right\rangle \\ &\quad +\|x^{k+1}-x^k\|^2 \\ &= \|x^k-x^*\|^2+2\lambda_k\left\langle \nabla f(x^k), x^*-x^k\right\rangle \\ &\quad +\|x^{k+1}-x^k\|^2.\end{aligned}$$

As usually, we bound the scalar product by convexity of f:

$$2\lambda_k\left\langle \nabla f(x^k), x^*-x^k\right\rangle \leq 2\lambda_k(f_*-f(x^k)), \tag{6}$$

which gives us

$$\begin{aligned}\|x^{k+1}-x^*\|^2 \leq \|x^k-x^*\|^2 &- 2\lambda_k(f(x^k)-f_*) \\ &+\|x^{k+1}-x^k\|^2.\end{aligned} \tag{7}$$

These two steps have been repeated thousands of times, but now we continue in a completely different manner. We have precisely one "bad" term in (7), which is $\|x^{k+1}-x^k\|^2$. We will bound it using the difference of gradients:

$$\begin{aligned}\|x^{k+1}-x^k\|^2 &= 2\|x^{k+1}-x^k\|^2-\|x^{k+1}-x^k\|^2 \\ &= -2\lambda_k\langle\nabla f(x^k), x^{k+1}-x^k\rangle-\|x^{k+1}-x^k\|^2 \\ &= 2\lambda_k\langle\nabla f(x^k)-\nabla f(x^{k-1}), x^k-x^{k+1}\rangle \\ &\quad +2\lambda_k\langle\nabla f(x^{k-1}), x^k-x^{k+1}\rangle-\|x^{k+1}-x^k\|^2.\end{aligned} \tag{8}$$

Let us estimate the first two terms in the right-hand side above. First, definition of λ_k, followed by Cauchy-Schwarz and Young's inequalities, yields

$$\begin{aligned}&2\lambda_k\langle\nabla f(x^k)-\nabla f(x^{k-1}), x^k-x^{k+1}\rangle \\ &\leq 2\lambda_k\|\nabla f(x^k)-\nabla f(x^{k-1})\|\|x^k-x^{k+1}\| \\ &\leq \|x^k-x^{k-1}\|\|x^k-x^{k+1}\| \\ &\leq \frac{1}{2}\|x^k-x^{k-1}\|^2+\frac{1}{2}\|x^{k+1}-x^k\|^2.\end{aligned} \tag{9}$$

Secondly, by convexity of f,

$$\begin{aligned}2\lambda_k\langle\nabla f(x^{k-1}), x^k-x^{k+1}\rangle &= \frac{2\lambda_k}{\lambda_{k-1}}\langle x^{k-1}-x^k, x^k-x^{k+1}\rangle \\ &= 2\lambda_k\theta_k\langle x^{k-1}-x^k, \nabla f(x^k)\rangle \\ &\leq 2\lambda_k\theta_k(f(x^{k-1})-f(x^k)).\end{aligned} \tag{10}$$

Plugging (9) and (10) in (8), we obtain

$$\begin{aligned}\|x^{k+1}-x^k\|^2 \leq &\frac{1}{2}\|x^k-x^{k-1}\|^2-\frac{1}{2}\|x^{k+1}-x^k\|^2 \\ &+2\lambda_k\theta_k(f(x^{k-1})-f(x^k)).\end{aligned}$$

Finally, using the produced estimate for $\|x^{k+1}-x^k\|^2$ in (7), we deduce the desired inequality (5). □

The above lemma already might give a good hint why our method works. From inequality (5) together with condition $\lambda_k^2 \leq (1+\theta_{k-1})\lambda_{k-1}^2$, we obtain that the Lyapunov energy—the left-hand side of (5)—is decreasing. This gives us boundedness of (x^k), which is often the key ingredient for proving convergence. In the next theorem we formally state our result.

Theorem 1. *Suppose that $f\colon \mathbb{R}^d \to \mathbb{R}$ is convex with locally Lipschitz gradient ∇f. Then (x^k) generated by Algorithm 1 converges to a solution of (1) and we have that*

$$f(\hat{x}^k)-f_* \leq \frac{D}{2S_k} = \mathcal{O}\Big(\frac{1}{k}\Big),$$

where

$$\begin{aligned}\hat{x}^k &= \frac{\lambda_k(1+\theta_k)x^k+\sum_{i=1}^{k-1}w_ix^i}{S_k}, \\ w_i &= \lambda_i(1+\theta_i)-\lambda_{i+1}\theta_{i+1}, \\ S_k &= \lambda_k(1+\theta_k)+\sum_{i=1}^{k-1}w_i = \sum_{i=1}^{k}\lambda_i+\lambda_1\theta_1,\end{aligned}$$

and D is a constant that explicitly depends on the initial data and the solution set, see (11).

Our proof will consist of two parts. The first one is a straightforward application of Lemma 1, from which we derive boundedness of (x^k) and complexity result. Due to its conciseness, we provide it directly after this remark. In the second part, we prove that the whole sequence (x^k) converges to a solution. Surprisingly, this part is a bit more technical than expected, and thus we postpone it to the appendix.

Proof. (Boundedness and complexity result.)

Fix any x^* from the solution set of eq. (1). Telescoping inequality (5), we deduce

$$\begin{aligned}&\|x^{k+1}-x^*\|^2+\frac{1}{2}\|x^{k+1}-x^k\|^2+2\lambda_k(1+\theta_k)(f(x^k)-f_*) \\ &\quad +2\sum_{i=1}^{k-1}[\lambda_i(1+\theta_i)-\lambda_{i+1}\theta_{i+1}](f(x^i)-f_*) \\ &\leq \|x^1-x^*\|^2+\frac{1}{2}\|x^1-x^0\|^2+2\lambda_1\theta_1[f(x^0)-f_*] \stackrel{\text{def}}{=} D.\end{aligned} \tag{11}$$

Note that by definition of λ_k, the second line above is always nonnegative. Thus, the sequence (x^k) is bounded. Since ∇f is locally Lipschitz, it is Lipschitz continuous on bounded sets. It means that for the set $\mathcal{C} = \overline{\text{conv}}\{x^*, x^0, x^1, \dots\}$, which is bounded as the convex hull of bounded points, there exists $L > 0$ such that

$$\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\| \quad \forall x, y \in \mathcal{C}.$$

Clearly, $\lambda_1 = \frac{\|x^1 - x^0\|}{2\|\nabla f(x^1) - \nabla f(x^0)\|} \ge \frac{1}{2L}$, thus, by induction one can prove that $\lambda_k \ge \frac{1}{2L}$, in other words, the sequence (λ_k) is separated from zero.

Now we want to apply the Jensen's inequality for the sum of all terms $f(x^i) - f_*$ in the left-hand side of (11). Notice, that the total sum of coefficients at these terms is

$$\lambda_k(1+\theta_k) + \sum_{i=1}^{k-1}[\lambda_i(1+\theta_i) - \lambda_{i+1}\theta_{i+1}] = \sum_{i=1}^{k}\lambda_i + \lambda_1\theta_1 = S_k$$

Thus, by Jensen's inequality,

$$\frac{D}{2} \ge \frac{\text{LHS of (11)}}{2} \ge S_k(f(\hat{x}^k) - f_*),$$

where $\hat{x}^k$ is given in the statement of the theorem. By this, the first part of the proof is complete. Convergence of (x^k) to a solution is provided in the appendix. □

As we have shown that $\lambda_i \ge \frac{1}{2L}$ for all i, we have a theoretical upper bound $f(\hat{x}^k) - f_* \le \frac{DL}{k}$. Note that in practice, however, (λ_k) might be much larger than the pessimistic lower bound $\frac{1}{2L}$, which we observe in our experiments together with a faster convergence.

2.3. f is locally strongly convex

Since one of our goals is to make optimization easy to use, we believe that a good method should have state-of-the-art guarantees in various scenarios. For strongly convex functions, this means that we want to see linear convergence, which is not covered by normalized GD or online methods. In section 2.1 we have shown that Algorithm 1 matches the $\mathcal{O}(1/\varepsilon)$ complexity of GD on convex problems. Now we show that it also matches $\mathcal{O}(\frac{L}{\mu}\log\frac{1}{\varepsilon})$ complexity of GD when f is locally strongly convex. Similarly to local smoothness, we call f *locally strongly convex* if it is strongly convex over any compact set of its domain.

For proof simplicity, instead of using bound $\lambda_k \le \sqrt{1+\theta_{k-1}}\lambda_{k-1}$ as in step 4 of Algorithm 1 we will use a more conservative bound $\lambda_k \le \sqrt{1+\frac{\theta_{k-1}}{2}}\lambda_{k-1}$ (otherwise the derivation would be too technical). It is clear that with such a change Theorem 1 still holds true, so the sequence is bounded and we can rely on local smoothness and local strong convexity.

Algorithm 2 Adaptive accelerated gradient descent

Input: $x^0 \in \mathbb{R}^d$, $\lambda_0 > 0$, $\Lambda_0 > 0$, $\theta_0 = \Theta_0 = +\infty$
$y^1 = x^1 = x^0 - \lambda_0\nabla f(x^0)$
for $k = 1, 2, \dots$ **do**
  $\lambda_k = \min\left\{\sqrt{1+\frac{\theta_{k-1}}{2}}\lambda_{k-1}, \frac{\|x^k - x^{k-1}\|}{2\|\nabla f(x^k) - \nabla f(x^{k-1})\|}\right\}$
  $\Lambda_k = \min\left\{\sqrt{1+\frac{\Theta_{k-1}}{2}}\Lambda_{k-1}, \frac{\|\nabla f(x^k) - \nabla f(x^{k-1})\|}{2\|x^k - x^{k-1}\|}\right\}$
  $\beta_k = \frac{\sqrt{1/\lambda_k} - \sqrt{\Lambda_k}}{\sqrt{1/\lambda_k} + \sqrt{\Lambda_k}}$
  $y^{k+1} = x^k - \lambda_k\nabla f(x^k)$
  $x^{k+1} = y^{k+1} + \beta_k(y^{k+1} - y^k)$
  $\theta_k = \frac{\lambda_k}{\lambda_{k-1}}$, $\Theta_k = \frac{\Lambda_k}{\Lambda_{k-1}}$
end for

Theorem 2. *Suppose that $f\colon \mathbb{R}^d \to \mathbb{R}$ is locally strongly convex and ∇f is locally Lipschitz. Then (x^k) generated by Algorithm 1 (with the modification mentioned above) converges to the solution x^* of (1). The complexity to get $\|x^k - x^*\|^2 \le \varepsilon$ is $\mathcal{O}(\kappa\log\frac{1}{\varepsilon})$, where $\kappa = \frac{L}{\mu}$ and L, μ are the smoothness and strong convexity constants of f on the set $\mathcal{C} = \overline{\text{conv}}\{x^*, x^0, x^1, \dots\}$.*

We want to highlight that in our rate κ depends on the local Lipschitz and strong convexity constants L and μ, which is meaningful even when these properties are not satisfied globally. Similarly, if f is globally smooth and strongly convex, our rate is still faster as it depends on the smaller local constants.

3. Heuristics

In this section, we describe several extensions of our method. We do not have a full theory for them, but believe that they are of interest in applications.

3.1. Acceleration

Suppose that f is μ-strongly convex. One version of the accelerated gradient method proposed by Nesterov (Nesterov, 2013a) is

$$y^{k+1} = x^k - \frac{1}{L}\nabla f(x^k),$$
$$x^{k+1} = y^{k+1} + \beta(y^{k+1} - y^k),$$

where $\beta = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$. Adaptive gradient descent for strongly convex f efficiently estimated $\frac{1}{2L}$ by

$$\lambda_k = \min\left\{\sqrt{1+\frac{\theta_{k-1}}{2}}\lambda_{k-1}, \frac{\|x^k - x^{k-1}\|}{2\|\nabla f(x^k) - \nabla f(x^{k-1})\|}\right\}.$$

What about the strong convexity constant μ? We know that it equals to the inverse smoothness constant of the conjugate

$f^*(y) \stackrel{\text{def}}{=} \sup_x\{\langle x, y\rangle - f(x)\}$. Thus, it is tempting to estimate this inverse constant just as we estimated inverse smoothness of f, i.e., by formula

$$\Lambda_k = \min\left\{\sqrt{1+\frac{\Theta_{k-1}}{2}}\Lambda_{k-1}, \frac{\|p^k - p^{k-1}\|}{2\|\nabla f^*(p^k) - \nabla f^*(p^{k-1})\|}\right\}$$

where p^k and p^{k-1} are some elements of the dual space and $\Theta_k = \frac{\Lambda_k}{\Lambda_{k-1}}$. A natural choice then is $p^k = \nabla f(x^k)$ since it is an element of the dual space that we use. What is its value? It is well known that $\nabla f^*(\nabla f(x)) = x$, so we come up with the update rule

$$\Lambda_k = \min\left\{\sqrt{1+\frac{\Theta_{k-1}}{2}}\Lambda_{k-1}, \frac{\|\nabla f(x^k) - \nabla f(x^{k-1})\|}{2\|x^k - x^{k-1}\|}\right\},$$

and hence we can estimate β by $\beta_k = \frac{\sqrt{1/\lambda_k}-\sqrt{\Lambda_k}}{\sqrt{1/\lambda_k}+\sqrt{\Lambda_k}}$.

We summarize our arguments in Algorithm 2. Unfortunately, we do not have any theoretical guarantees for it.

Estimating strong convexity parameter μ is important in practice. Most common approaches rely on restarting technique proposed by (Nesterov, 2013b), see also (Fercoq & Qu, 2019) and references therein. Unlike Algorithm 2, these works have theoretical guarantees, however, the methods themselves are more complicated and still require tuning of other unknown parameters.

3.2. Uniting our steps with stochastic gradients

Here we would like to discuss applications of our method to the problem

$$\min_x \mathbb{E}\left[f_\xi(x)\right],$$

where f_ξ is almost surely L-smooth and μ-strongly convex. Assume that at each iteration we get sample ξ^k to make a stochastic gradient step,

$$x^{k+1} = x^k - \lambda_k \nabla f_{\xi^k}(x^k).$$

Then, we have two ways of incorporating our stepsize into SGD. The first is to reuse $\nabla f_{\xi^k}(x^k)$ to estimate $L_k = \frac{\|\nabla f_{\xi^k}(x^k) - \nabla f_{\xi^k}(x^{k-1})\|}{\|x^k - x^{k-1}\|}$, but this would make $\lambda_k \nabla f_{\xi^k}(x^k)$ biased. Alternatively, one can use an extra sample to estimate L_k, but this is less intuitive since our goal is to estimate the curvature of the function used in the update.

We give a full description in Algorithm 3. We remark that the option with a biased estimate performed much better in our experiments with neural networks. The theorem below provides convergence guarantees for both cases, but with different assumptions.

Theorem 3. *Let f_ξ be L-smooth and μ-strongly convex almost surely. Assuming $\alpha \leq \frac{1}{2\kappa}$ and estimating L_k with ∇f_{ζ^k}, the complexity to get $\mathbb{E}\left[\|x^k - x^*\|^2\right] \leq \varepsilon$ is not worse than $\mathcal{O}\left(\frac{\kappa^2}{\varepsilon}\log\frac{\kappa}{\varepsilon}\right)$. Furthermore, if the model is overparameterized, i.e., $\nabla f_\xi(x^*) = 0$ almost surely, then one can estimate L_k with ξ^k and the complexity is $\mathcal{O}\left(\kappa^2 \log\frac{1}{\varepsilon}\right)$.*

Algorithm 3 Adaptive SGD

Input: $x^0 \in \mathbb{R}^d$, $\lambda_0 > 0$, $\theta_0 = +\infty$, ξ^0, $\alpha > 0$
$x^1 = x^0 - \lambda_0 \nabla f_{\xi^0}(x^0)$
for $k = 1, 2, \ldots$ **do**
  Sample ξ^k and optionally ζ^k
  Option I (biased): $L_k = \frac{\|\nabla f_{\xi^k}(x^k) - \nabla f_{\xi^k}(x^{k-1})\|}{\|x^k - x^{k-1}\|}$
  Option II (unbiased): $L_k = \frac{\|\nabla f_{\zeta^k}(x^k) - \nabla f_{\zeta^k}(x^{k-1})\|}{\|x^k - x^{k-1}\|}$
  $\lambda_k = \min\left\{\sqrt{1+\theta_{k-1}}\lambda_{k-1}, \frac{\alpha}{L_k}\right\}$
  $x^{k+1} = x^k - \lambda_k \nabla f_{\xi^k}(x^k)$
  $\theta_k = \frac{\lambda_k}{\lambda_{k-1}}$
end for

Note that in both cases we match the known dependency on ε up to logarithmic terms, but we get an extra κ as the price for adaptive estimation of the stepsize.

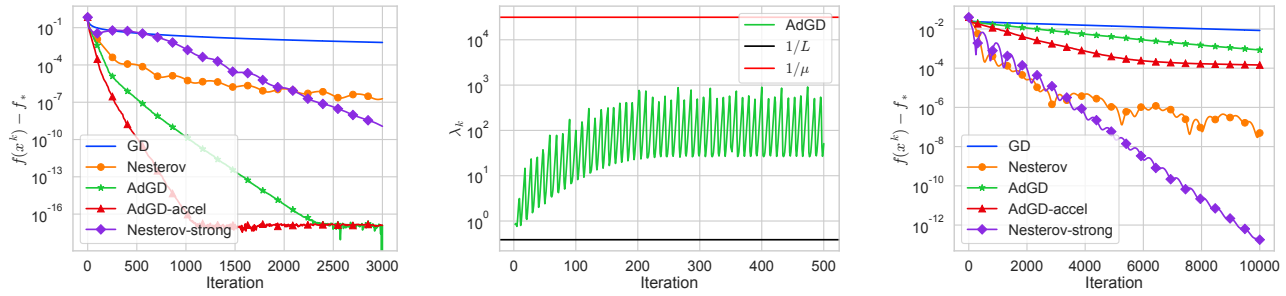
Another potential application of our techniques is estimation of decreasing stepsizes in SGD. The best known rates for SGD (Stich, 2019), are obtained using λ_k that evolves as $\mathcal{O}\left(\frac{1}{L+\mu k}\right)$. This requires estimates of both smoothness and strong convexity, which can be borrowed from the previous discussion. We leave rigorous proof of such schemes for future work.

4. Experiments

In the experiments[3], we compare our approach with the two most related methods: GD and Nesterov's accelerated method for convex functions (Nesterov, 1983). Additionally, we consider line search, Polyak step, and Barzilai-Borwein method. For neural networks we also include a comparison with SGD, SGDm and Adam.

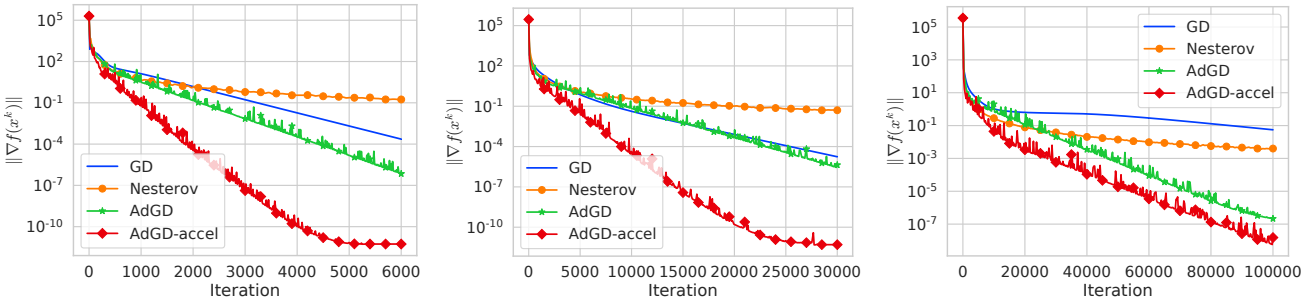
Logistic regression. The logistic loss with ℓ_2-regularization is given by $\frac{1}{n}\sum_{i=1}^n \log(1+\exp(-b_i a_i^\top x)) + \frac{\gamma}{2}\|x\|^2$, where n is the number of observations, $\gamma > 0$ is a regularization parameter, and $(a_i, b_i) \in \mathbb{R}^d \times \mathbb{R}$, $i = 1, \ldots, n$, are the observations. We use 'mushrooms' and 'covtype' datasets to run the experiments. We choose γ proportionally to $\frac{1}{n}$ as often done in practice. Since we have closed-form expressions to estimate $L = \frac{1}{4n}\|A\|^2 + \gamma$, where $A = (a_1^\top, \ldots, a_n^\top)^\top$, we used stepsize $\frac{1}{L}$ in GD and its acceleration. The results are provided in Figure 1.

[3] See https://github.com/ymalitsky/adaptive_gd



(a) Mushrooms dataset, objective (b) Mushrooms dataset, stepsize (c) Covtype dataset, objective

Figure 1: Results for the logistic regression problem.



(a) $r = 10$ (b) $r = 20$ (c) $r = 30$

Figure 2: Results for matrix factorization. The objective is neither convex nor smooth.

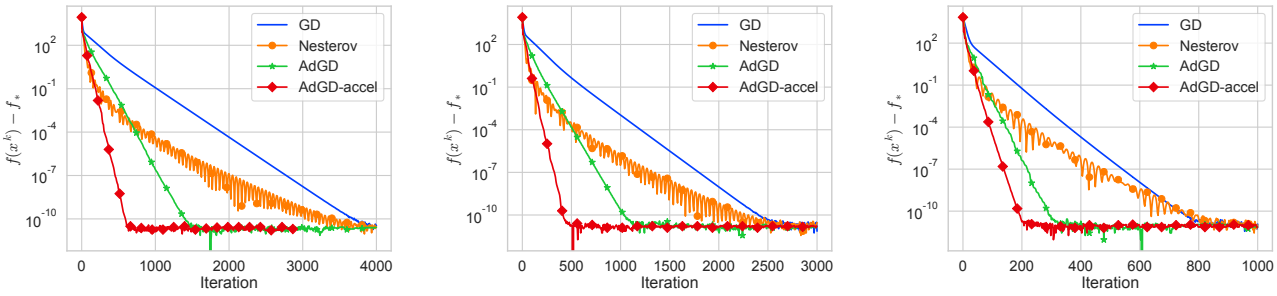
Matrix factorization. Given a matrix $A \in \mathbb{R}^{m\times n}$ and $r < \min\{m, n\}$, we want to solve $\min_{X=[U,V]} f(X) = f(U, V) = \frac{1}{2}\|UV^\top - A\|_F^2$ for $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$. It is a nonconvex problem, and the gradient ∇f is not globally Lipschitz. With some tuning, one still can apply GD and Nesterov's accelerated method, but—and we want to emphasize it—it was not a trivial thing to find the steps in practice. The steps we have chosen were almost optimal, namely, the methods did not converge if we doubled the steps. In contrast, our methods do not require any tuning, so even in this regard they are much more practical. For the experiments we used Movilens 100K dataset (Harper & Konstan, 2016) with more than million entries and several values of $r = 10,\ 20,\ 30$. All algorithms were initialized at the same point, chosen randomly. The results are presented in Figure 2.

Cubic regularization. In cubic regularization of Newton method (Nesterov & Polyak, 2006), at each iteration we need to minimize $f(x) = g^\top x + \frac{1}{2}x^\top Hx + \frac{M}{6}\|x\|^3$, where $g \in \mathbb{R}^d, H \in \mathbb{R}^{d\times d}$ and $M > 0$ are given. This objective is smooth only locally due to the cubic term, which is our motivation to consider it. g and H were the gradient and the Hessian of the logistic loss with the 'covtype' dataset, evaluated at $x = 0 \in \mathbb{R}^d$. Although the values of $M = 10,\ 20,\ 100$ led to similar results, they also required different numbers of iterations, so we present the corresponding results in Figure 3.

Barzilai-Borwein, Polyak and line searches. We have started this paper with an overview of different approaches to tackle the issue of a stepsize for GD. Now, we demonstrate some of those solutions. We again consider the ℓ_2-regularized logistic regression (same setting as before) with 'mushrooms', 'covtype', and 'w8a' datasets.

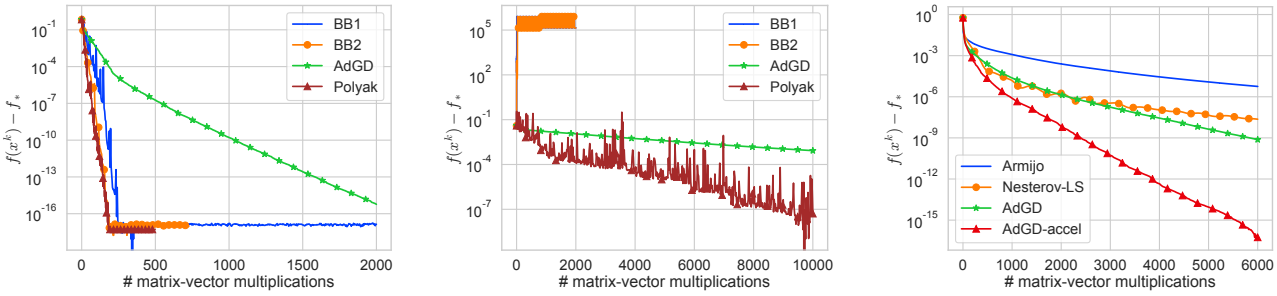
In Figure 4 (left) we see that the Barzilai-Borwein method can indeed be very fast. However, as we said before, it lacks a theoretical basis and Figure 4 (middle) illustrates this quite well. Just changing one dataset to another makes both versions of this method to diverge on a strongly convex and smooth problem. Polyak's method consistently performs well (see Figure 4 (left and middle)), however, only after it was fed with f_* that we found by running another method. Unfortunately, for logistic regression there is no way to guess this value beforehand.

Finally, line search for GD (Armijo version) and Nesterov GD (implemented as in (Nesterov, 2013b)) eliminates the need to know the stepsize, but this comes with a higher price per iteration as Figure 4 (right) shows. Actually in all our ex-



(a) $M = 10$ (b) $M = 20$ (c) $M = 100$

Figure 3: Results for the non-smooth subproblem from cubic regularization.



(a) Mushrooms dataset, objective (b) Covtype dataset, objective (c) W8a dataset, objective

Figure 4: Additional results for the logistic regression problem.

periments for logistic regression with different datasets one iteration of Armijo line search was approximately 2 times more expensive than AdGD, while line search for Nesterov GD was 4 times more expensive. We note that these observations are consistent with the theoretical derivations in (Nesterov, 2013b).

Neural networks. We use standard ResNet-18 and DenseNet-121 architectures implemented in PyTorch (Paszke et al., 2017) and train them to classify images from the Cifar10 dataset (Krizhevsky et al., 2009) with cross-entropy loss.

We use batch size 128 for all methods. For our method, we observed that $\frac{1}{L_k}$ works better than $\frac{1}{2L_k}$. We ran it with $\sqrt{1+\gamma\theta_k}$ in the other factor with values of γ from $\{1, 0.1, 0.05, 0.02, 0.01\}$ and $\gamma = 0.02$ performed the best. For reference, we provide the result for the theoretical estimate as well and value $\gamma = 0.1$ in the plot with estimated stepsizes. The results are depicted in Figures 5 and 6 and other details are provided in appendix D.

We can see that, surprisingly, our method achieves better test accuracy than SGD despite having the same train loss. At the same time, our method is significantly slower at the early stage and the results are quite noisy for the first 75 epochs. Another observation is that the smoothness estimates are very non-uniform and λ_k plummets once train loss becomes small.

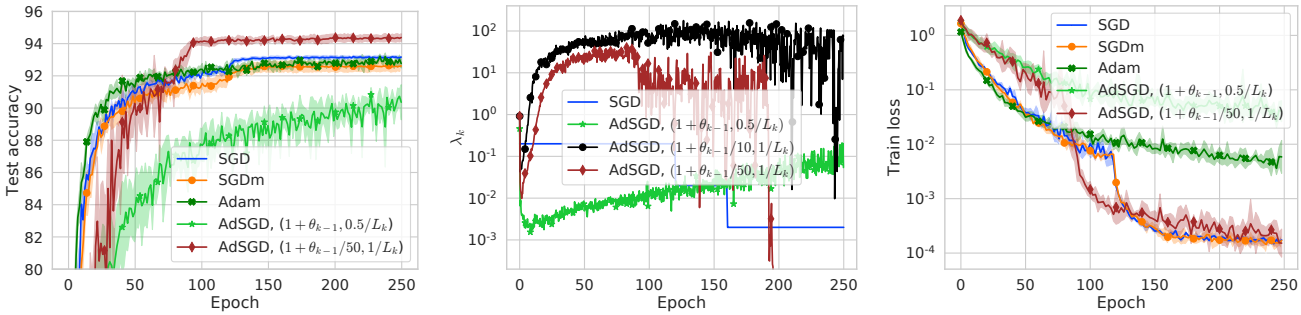
5. Perspectives

We briefly provide a few directions which we personally consider to be important and challenging.

1. **Nonconvex case.** A great challenge for us is to obtain theoretical guarantees of the proposed method in the nonconvex settings. We are not aware of any generic first-order method for nonconvex optimization that does not rely on the descent lemma (or its generalization), see, e.g., (Attouch et al., 2013).

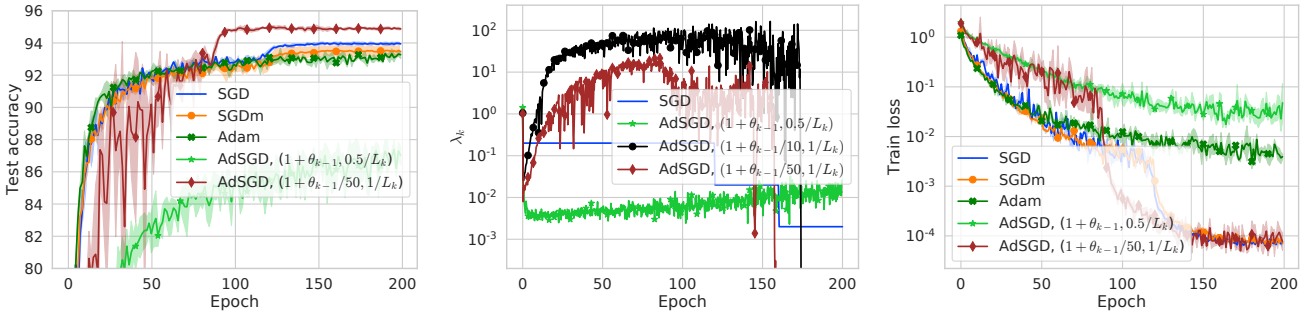
2. **Performance estimation.** In our experiments we often observed much better performance of Algorithm 1, than GD or AGD. However, the theoretical rate we can show coincides with that of GD. The challenge here is to bridge this gap and we hope that the approach pioneered by (Drori & Teboulle, 2014) and further developed in (Taylor et al., 2017; Kim & Fessler, 2016; Taylor & Bach, 2019) has a potential to do that.

3. **Composite minimization.** In classical first-order



(a) Test accuracy (b) Stepsize (c) Train loss

Figure 5: Results for training ResNet-18 on Cifar10. Labels for AdGD correspond to how λ_k was estimated.



(a) Test accuracy (b) Stepsize (c) Train loss

Figure 6: Results for training DenseNet-121 on Cifar10.

methods, the transition from smooth to composite minimization (Nesterov, 2013b) is rather straightforward. Unfortunately, the proposed proof of Algorithm 1 does not seem to provide any route for generalization and we hope there is some way of resolving this issue.

4. **Stochastic optimization.** The derived bounds for the stochastic case are not satisfactory and have a suboptimal dependency on κ. However, it is not clear to us whether one can extend the techniques from the deterministic analysis to improve the rate.

5. **Heuristics.** Finally, we want to have some solid ground in understanding the performance of the proposed heuristics.

Acknowledgements

Yura Malitsky wishes to thank Roman Cheplyaka for his interest in optimization that partly inspired the current work. Yura Malitsky was supported by the ONRG project N62909-17-1-2111 and HASLER project N16066.

References

Armijo, L. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, (1):1–3.

Attouch, H., Bolte, J., and Svaiter, B. F. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods. *Mathematical Programming*, 137(1-2):91–129, 2013.

Barzilai, J. and Borwein, J. M. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

Bauschke, H. H., Bolte, J., and Teboulle, M. A descent lemma beyond Lipschitz gradient continuity: first-order methods revisited and applications. *Mathematics of Operations Research*, 42(2):330–348, 2016.

Bello Cruz, J. Y. and Nghia, T. T. On the convergence of the forward–backward splitting method with linesearches. *Optimization Methods and Software*, 31(6):1209–1238, 2016.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signSGD: Compressed optimisation for non-

convex problems. In *International Conference on Machine Learning*, pp. 559–568, 2018.

Birnbaum, B., Devanur, N. R., and Xiao, L. Distributed algorithms via gradient descent for Fisher markets. In *Proceedings of the 12th ACM conference on Electronic commerce*, pp. 127–136, 2011.

Burdakov, O., Dai, Y., and Huang, N. Stabilized Barzilai-Borwein method. *Journal of Computational Mathematics*, 37(6):916–936, 2019.

Cauchy, A. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

Dai, Y.-H. and Liao, L.-Z. R-linear convergence of the Barzilai and Borwein gradient method. *IMA Journal of Numerical Analysis*, 22(1):1–10, 2002.

Drori, Y. and Teboulle, M. Performance of first-order methods for smooth convex minimization: a novel approach. *Mathematical Programming*, 145(1-2):451–482, 2014.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Fercoq, O. and Qu, Z. Adaptive restart of accelerated gradient methods under local quadratic growth condition. *IMA Journal of Numerical Analysis*, 39(4):2069–2095, 2019.

Goldstein, A. Cauchy's method of minimization. *Numerische Mathematik*, 4(1):146–150, 1962.

Harper, F. M. and Konstan, J. A. The movielens datasets: History and context. *ACM transactions on interactive intelligent systems (tiis)*, 5(4):19, 2016.

Hazan, E. and Kakade, S. Revisiting the Polyak step size. *arXiv preprint arXiv:1905.00313*, 2019.

Karimireddy, S. P., Rebjock, Q., Stich, S., and Jaggi, M. Error feedback fixes signSGD and other gradient compression schemes. In *International Conference on Machine Learning*, pp. 3252–3261, 2019.

Kim, D. and Fessler, J. A. Optimized first-order methods for smooth convex minimization. *Mathematical programming*, 159(1-2):81–107, 2016.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Lemaréchal, C. Cauchy and the gradient method. *Doc Math Extra*, 251:254, 2012.

Maddison, C. J., Paulin, D., Teh, Y. W., and Doucet, A. Dual space preconditioning for gradient descent. *arXiv preprint arXiv:1902.02257*, 2019.

Malitsky, Y. Golden ratio algorithms for variational inequalities. *Mathematical Programming*, Jul 2019. doi: 10.1007/s10107-019-01416-w.

McMahan, H. B. and Streeter, M. Adaptive bound optimization for online convex optimization. In *Proceedings of the 23rd Annual Conference on Learning Theory (COLT)*, 2010.

Nemirovsky, A. S. and Yudin, D. B. *Problem complexity and method efficiency in optimization*. John Wiley & Sons, Inc., New York, 1983.

Nesterov, Y. A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. *Doklady AN SSSR*, 269(3):543–547, 1983.

Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013a.

Nesterov, Y. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013b.

Nesterov, Y. and Polyak, B. T. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. 2017.

Polyak, B. T. Gradient methods for minimizing functionals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 3(4):643–653, 1963.

Polyak, B. T. Minimization of nonsmooth functionals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 9(3):509–521, 1969.

Raydan, M. On the Barzilai and Borwein choice of steplength for the gradient method. *IMA Journal of Numerical Analysis*, 13(3):321–326, 1993.

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

Salzo, S. The variable metric forward-backward splitting algorithm under mild differentiability assumptions. *SIAM Journal on Optimization*, 27(4):2153–2181, 2017.

Shor, N. An application of the method of gradient descent to the solution of the network transportation problem. *Materialy Naucnovo Seminara po Teoret i Priklad. Voprosam Kibernet. i Issted. Operacii, Nucnyi Sov. po Kibernet, Akad. Nauk Ukrain. SSSR, vyp*, 1:9–17, 1962.

Stich, S. U. Unified optimal analysis of the (stochastic) gradient method. *arXiv preprint arXiv:1907.04232*, 2019.

Taylor, A. and Bach, F. Stochastic first-order methods: non-asymptotic and computer-aided analyses via potential functions. In Beygelzimer, A. and Hsu, D. (eds.), *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pp. 2934–2992, Phoenix, USA, 25–28 Jun 2019. PMLR.

Taylor, A. B., Hendrickx, J. M., and Glineur, F. Smooth strongly convex interpolation and exact worst-case performance of first-order methods. *Mathematical Programming*, 161(1-2):307–345, 2017.

Tieleman, T. and Hinton, G. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

Ward, R., Wu, X., and Bottou, L. AdaGrad stepsizes: Sharp convergence over nonconvex landscapes. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6677–6686, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Zeiler, M. D. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.