Applied Mathematics & Optimization (2021) 84:2557–2586
https://doi.org/10.1007/s00245-020-09718-8

Backtracking Gradient Descent Method and Some Applications in Large Scale Optimisation. Part 2: Algorithms and Experiments

Tuyen Trung Truong[1] · **Hang-Tuan Nguyen**[2]

Published online: 6 September 2020

Abstract
In this paper, we provide new results and algorithms (including backtracking versions of Nesterov accelerated gradient and Momentum) which are more applicable to large scale optimisation as in Deep Neural Networks. We also demonstrate that Backtracking Gradient Descent (Backtracking GD) can obtain good upper bound estimates for local Lipschitz constants for the gradient, and that the convergence rate of Backtracking GD is similar to that in classical work of Armijo. Experiments with datasets CIFAR10 and CIFAR100 on various popular architectures verify a heuristic argument that Backtracking GD stabilises to a finite union of sequences constructed from Standard GD for the mini-batch practice, and show that our new algorithms (while automatically fine tuning learning rates) perform better than current state-of-the-art methods such as Adam, Adagrad, Adadelta, RMSProp, Momentum and Nesterov accelerated gradient. To help readers avoiding the confusion between heuristics and more rigorously justified algorithms, we also provide a review of the current state of convergence results for gradient descent methods. Accompanying source codes are available on GitHub.

Keywords Automation of learning rates · Backtracking · Deep neural networks · Random dynamical systems · Global convergence · Gradient descent · Image classification · Iterative optimisation · Large scale optimisation · Local minimum

Mathematics Subject Classification 65Kxx · 68Txx · 49Mxx · 68Uxx

✉ Tuyen Trung Truong
tuyentt@math.uio.no

Hang-Tuan Nguyen
hnguyen@axon.com

[1] Matematikk Institut, Universitetet i Oslo, Blindern 0851, Oslo, Norway

[2] Axon AI Research, Seattle, WA, USA

1 Introduction

In this section, we provide a non-technical overview of the important role and current practices of Gradient Descent methods (GD) in optimisation, in particular in large scale optimisation as in Deep Neural Networks (DNN), and some new features of our main results in this paper.

One special feature of the modern society is the need of solving large scale optimisation problems quickly, stably, efficiently and reproducibly. One exemplar for this is the development of Deep Learning, which has obtained spectacular achievements recently. Among the most famous novelties, one can mention Alpha Go (the first computer program that ever won human players, and in fact the best human players, in the Game of Go) and new developments in self-driving cars. One important tool in Deep Learning is DNN.

Roughly speaking, a DNN is a parametrised family $f(x, \theta)$, a composition of very simple maps, aiming to approximate well a feature y. By choosing an explicit metric to measure the difference between the prediction by the DNN $f(x, \theta)$ and the "ground truth" y over a training set I, one gets a specific value of θ as one global minimum of an associated optimisation problem. The special feature of this optimisation problem is that it is large scale and non-convex. Here we illustrate with the task of recognising hand written images from the dataset MNIST [26], which is an immense challenge for the old paradigm of rule-based learning, but is considered only a simple exercise for DNN. Below are some images from MNIST (see Fig. 1). Some of these pictures are challenging even for a human being to classify.

The DNN used for this MNIST task is regarded as "small", and needs "only" $11,935$ parameters. Modern state of the art DNN can easily have millions of parameters. With this big size of optimisation problem, the only tool one could rely on are numerical optimisation algorithms, serving to arrive closely to good local minima.

As great as it is, there are many serious concerns about the current practices in Deep Learning such as it is very easily fooled and is still not safe. Since optimisation methods are an important part of the practice, improving optimisation methods promises to improving the performance of Deep Learning. A numerical algorithm, to be universally used in realistic applications such as in Deep Learning, should have good theoretical justification (such as assurance of convergence to critical points and avoidance of saddle points) as well as easy to implement and run in large scale optimisation, and as such will help towards resolving the current deficits in the practices in Deep Learning mentioned above. Our paper contributes to such algorithms. In this paper, we will work with real functions which are continuously differentiable. This is the most general class that current techniques can be used in solving non-convex optimisation problems, and therefore is flexible enough to adapt to many kinds of realistic applications.

1.1 A Brief Introduction to Gradient Descent Methods

Among many others, gradient descent (GD)—introduced in 1847 by Cauchy [10]—is one of the oldest, most well-known and most effective numerical optimisation methods. As suggested by the name, GD uses gradients of the functions which, by the chain

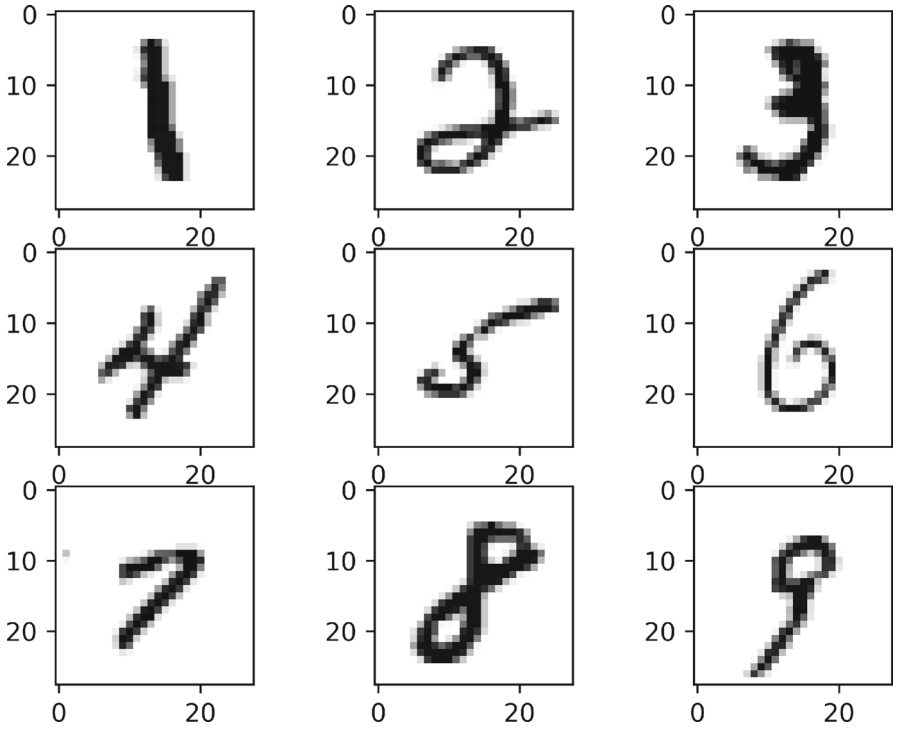


Fig. 1 Some samples from the MNIST hand-written digit dataset . (Source: [26])

rule in multiple calculus, make computations with compositions of simple functions as in DNN possible. (Second or higher order differential methods, such as Newton's, are more costly to implement on computers. Moreover, while Newton's method has better convergence rate than GD in the local setting—that is when we start close to a minimum point—no general result on global convergence to a local minimum has been proven for Newton's method, and the method can diverge to infinity even for functions with compact sublevels or converge to a saddle point.) The general form for this approximation method GD is as follows. Let $f : \mathbb{R}^m \to \mathbb{R}$ be a C^1 function which we would like to minimise. Denote by $\nabla f(x)$ the gradient (or derivative) of f. We start from a *random* point $z_0 \in \mathbb{R}^m$, and then construct iteratively

$$z_{n+1} = z_n - \delta_n \nabla f(z_n), \tag{1}$$

where $\delta_n > 0$ are appropriately chosen. In the literature, it is common to call δ_n learning rates or step sizes. The hope is that z_n will converge to a (global) minimum. The intuition is taken from the familiar picture one obtains when f is a convex function. Note that z_0 being random is important here: In Nesterov [30], one can find a function in 2 variables and a specific choice of the point z_0 for which any sequence as in (1), if converges to a critical point at all, can only converge to a saddle point.

The most known GD method is Standard GD, where we choose $\delta_n = \delta_0$ for all n. Hence, we start with a (randomly chosen) point $z_0 \in \mathbb{R}^m$ and define the sequence

$$z_{n+1} = z_n - \delta_0 \nabla f(z_n). \tag{2}$$

In reality, when doing actual computation on computers, one can only obtain the values for gradients and other quantities within some errors. The inexact version of GD, which is more suitable to realistic applications such as DNN, is as follows:

$$z_{n+1} = z_n - \delta_n v_n, \tag{3}$$

where we assume that v_n is not too far from the gradient direction $\nabla f(z_n)$. There are many ways to specify the condition of "not too far" (see e.g. [5,7]), here in this paper we use the following common version: there are $A_1, A_2 > 0$ and $1 \geq \mu > 0$ such that for all n

$$A_1 ||\nabla f(z_n)|| \leq ||v_n|| \leq A_2 ||\nabla f(z_n)||, \tag{4}$$

$$< \nabla f(z_n), v_n > \geq \mu ||\nabla f(z_n)|| \times ||v_n||. \tag{5}$$

When $\mu = 1$ we have that v_n is parallel to $\nabla f(z_n)$ for all n, and thus recover the general scheme for GD in (1). The geometric meaning of (5) is that the cosine of the angles between $\nabla f(z_n)$ and v_n are positive and uniformly bounded away from 0.

Remark If the starting point is not random, then the limit point (if exists) may be a saddle point. One such example (by Nesterov, mentioned above) is $f(x_1, x_2) = 2x_1^2 + x_2^4 - x_2^2$, with the starting point $z_0 = (1, 0)$, here $v_n = \nabla f(z_n)$ and $\delta_n > 0$ is arbitrary.

For ease of later reference, we denote by $C_L^{1,1}$ the set of C^1 functions f whose gradient is globally Lipschitz with Lipschitz constant L, that is $||\nabla f(x) - \nabla f(y)|| \leq L||x - y||$ for all $x, y \in \mathbb{R}^m$. It is worthy to remark that the class $C_L^{1,1}$ is not preserved under small perturbations, even under the common technique of L^q regularisations (or compensation) in DNN, for $q \neq 2$.

1.2 What is the State-of-the-Art for Convergence of GD Methods?

From its origin, it has been an important theoretical task for GD methods (as well as other iterative methods such as Newton's) to ensure that the sequence $\{x_n\}$ in (1) *converges* to a (global) *minimum* point. We now argue that from a practical viewpoint, theoretical guarantee for convergence of the proposed algorithms are also as important. For example, in practical DNN, even when one uses one of the iterative methods to find minima, one cannot run the algorithm infinitely and must stop after a certain finite time, say j_0. Then one will use the trained value z_{j_0} to make predictions for new data. If convergence for the sequence $\{z_n\}$ is guaranteed, then it is reasonable that next time when one (or another person) runs the training again (or when having new training data incorporated), even if one does not stop at the same time j_0, one will

still receive similar predictions, and hence one has reproducibility and stability. On the same token, if convergence is not guaranteed, then one can expect very different results for different experiments.

While the research in GD is very extensive and has been conducted nearly 200 years (see classical works [2,11,12,16,37], and also some modern presentations [5,7, 8,32,38]), we can say that results prior to our work concern mostly on the convergence of the sequence $\{\nabla f(z_n)\}$ to 0. One can see easily that this is equivalent to having only that any *cluster* point of $\{z_n\}$ is a critical point of f, and it is not equivalent to (and weaker than) having that $\{z_n\}$ itself converges. One reason to explain is that the previous work mostly concern the case of Standard GD, which does not have good convergence property, and which can be cumbersome in many situations: for example, if the function is not $C^{1,1}_L$, or if the Lipschitz constant L is not known, or if one works in a general manifold. In contrast, our recent works [44,46,47] show that Backtracking GD has very good convergence properties and is very flexible to apply to many settings, as well as works well and saves time in practical applications. So far, one can say that Backtracking GD is the best theoretically guaranteed method among all GD methods and other iterative methods such as Newton's. More details will be presented in Sect. 2.

1.3 What is New About this Paper?

This paper is extracted and developed from the more applied part (mainly, Sects. 3.1, 3.2 and 3.3 and Sect. 4, together with part of Sect. 3.4) of the preprint [47], together with the preprints [45,48]. We concentrate on algorithms with good theoretical justification and work well practically on large scale optimisation such as in DNN. The more theoretical part of Truong and Nguyen [47] will be published separately.

- We define and establish convergence of Backtracking versions of NAG and Momentum (MMT). This is helpful, since NAG and MMT help to avoid bad minima.
- We present a heuristic argument, based on Backtracking GD, to explain why Standard GD itself works so effectively as observed in practice. A practical implication of this heuristic argument is that even though it may seem counterintuitive when one first thinks about it—the running time of Backtracking GD (in particular Two-way Backtracking GD) is not too much longer than the running time of Standard GD (where manual fine-tuning or grid choice of learning rates is common practice).
- We define a new algorithm called Two-way Backtracking GD, by looking for $\delta(x_{n+1})$ starting not from δ_0 but only from $\delta(x_n)$, and then increase or decrease the learning rate depending on whether Armijo's condition is satisfied. This helps to save time.
- In common versions of GD so far, learning rates δ_n are usually bounded from above, which may make the process to converge to bad critical points or to slow down the convergence, since when z_n is close to a critical point z the change $z_{n+1} - z_n$ is bounded by $||\nabla f(x_n)||$ which is small. We will define an Unbounded Backtracking version where learning rates, for example, are bounded by $||\nabla f(x_n)||^{-\gamma}$ for some constant $0 < \gamma < 1$ when $||\nabla f(x_n)||$ becomes small so that Armijo's

condition is satisfied, and show that the good properties we see above are preserved. We illustrate (both theoretically and experimentally) that the performance of Unbounded Backtracking GD can be sharply different (compared to the usual Backtracking GD) depending on whether $\{z_n\}$ converges to a non-degenerate or degenerate critical point.
– We then make some other improvements (including a hybrid between Backtracking GD and Standard GD) on the algorithms, with sufficient theoretical justifications, and implement into DNN. Our algorithm is automatic, and do not use manual fine-tuning or grid search. Plus, on CIFAR10 and CIFAR100 on several common DNN architectures, its performance is better than popular algorithms such as Adam, SGD, Momentum, and so on.
– We also note that while in this paper we concern only C^1 functions, the results presented here are applicable also to more general functions. In fact, as discussed in the more theoretical part of Truong and Nguyen [47], the results can be applied for continuous functions whose gradient is not globally defined, provided we can use the Fundamental Theorem of Calculus, such as under the more general setting of Lebesgue's integration and differentiation. For example, this is the case if f is continuous on $\mathbb{R}^k$, and there is a closed set $A \subset \mathbb{R}^k$ of Lebesgue measure 0 so that f is C^1 on $\mathbb{R}^k \setminus A$, and ∇f is locally bounded near A.
– In real life applications, it is important that an algorithm is as most theoretically justified, reliable, reproducible and explainable as possible, besides good experimental results. The main message of this paper is that, even though now not popular, it is worth to test and implement more Backtracking GD.

Remark After both versions of the preprint [47] were posted on arXiv, a preprint by other authors [49] was posted where similar implementations in DNN and experimental results have been obtained and where convergence for a version of Backtracking GD for the mini-batch setting (under the assumption that the cost function f is strongly convex and is a sum of convex functions f_i whose gradients are in $C^{1,1}_{L_i}$, and that f_i has only one critical point which is the same as the global minimum of f). Note that the mini-batch setting considered in Vaswani et al. [49] is not the same as the usual setting of Stochastic GD.

Plan of the paper In Sect. 2, we provide a brief review of main theoretical results in Truong and Nguyen [47] which we will base on to develop new algorithms, together with major convergence results in previous literature by other authors. In Sect. 3, we propose new methods for large scale optimisation together with convergence results and proofs, as well as some heuristic arguments. In Sect. 4 we present experimental results. Accompanying source codes for the experiments in Sect. 4 are available at the following GitHub link Nguyen [31]. Section 5 presents conclusions and some directions for future work.

2 Overview and Comparison of Previous Results

The first subsection recalls our recent general convergence result for Backtracking GD which we will base on to develop new algorithms and results in Sect. 3. The second

subsection provides a brief review of previous work on convergence of GD methods, to help readers avoiding the confusion between heuristics and more rigorously justified algorithms.

2.1 A General Convergence Result for Backtracking GD

We recall first the so-called Armijo's condition, for some $0 < \alpha < 1$ and some $x, y \in \mathbb{R}^m$ (the main case of interest is when $y = x - \sigma \nabla f(x)$):

$$f(y) - f(x) \leq \alpha < \nabla f(x), y - x >, \tag{6}$$

here $< ., . >$ is the standard inner product in $\mathbb{R}^m$.

Now we introduce inexact Backtracking GD applicable for all C^1 functions $f : \mathbb{R}^m \to \mathbb{R}$, aiming for realistic applications such as DNN.

Inexact Backtracking GD Fix $A_1, A_2, \delta_0 > 0$; $1 \geq \mu > 0$ and $1 > \alpha, \beta > 0$. We start with a point z_0 and define the sequence $\{z_n\}$ by the following procedure. At step n:

(i) Choose a vector v_n satisfying $A_1||\nabla f(z_n)|| \leq ||v_n|| \leq A_2||\nabla f(z_n)||$ and $< \nabla f(z_n), v_n > \geq \mu||\nabla f(z_n)|| \times ||v_n||$. (These are the same as conditions (4) and (5) for Inexact GD.)

(ii) Choose δ_n to be the largest number σ among $\{\delta_0, \delta_0\beta, \delta_0\beta^2, \ldots\}$ so that

$$f(z_n - \sigma v_n) - f(z_n) \leq -\alpha\sigma < f(z_n), v_n > . \tag{7}$$

(This is Armijo's condition (6).) As in Lemma 3.1 in Truong and Nguyen [47], we can choose δ_n to be bounded on any compact set K on which ∇f is nowhere zero.

(iii) Define $z_{n+1} = z_n - \delta_n v_n$.

When $v_n = \nabla f(z_n)$ for all n, Inexact Backtracking GD reduces to standard Backtracking GD (aslo known as Armijo's rule in the literature). In this case, we can choose $A_1 = A_2 = \mu = 1$.

We recall that a point z^* is a cluster point of a sequence $\{z_n\}$ if there is a subsequence $\{z_{n_k}\}$ so that $\lim_{k\to\infty} z_{n_k} = z^*$. The sequence $\{z_n\}$ converges if and only if it has one and only one cluster point. It has been known that any cluster point of the sequence $\{z_n\}$ in the Backtracking GD is a critical point of f, see e.g. Proposition 1.2.1 in Bertsekas [5]. The following result, Theorem 2.7 in Truong and Nguyen [47], is the main theoretical result we will base on to develop new algorithms later in this paper.

Theorem 2.1 *Let $f : \mathbb{R}^m \to \mathbb{R}$ be a C^1 function and let z_n be a sequence constructed from the Inexact Backtracking GD procedure.*

(1) *Either* $\lim_{n\to\infty} f(z_n) = -\infty$ *or* $\lim_{n\to\infty} ||z_{n+1} - z_n|| = 0$.

(2) *Assume that f has at most countably many critical points. Then either* $\lim_{n\to\infty} ||z_n|| = \infty$ *or $\{z_n\}$ converges to a critical point of f.*

(3) *More generally, assume that the set of critical points of f contains a bounded connected component A. Let B be the set of cluster points of the sequence $\{z_n\}$. If $B \cap A \neq \emptyset$, then B is connected and $B \subset A$.*

The assumption in part 2 of Theorem 2.1 is satisfied by all Morse functions. We recall that a C^2 function f is Morse if all of its critical points are non-degenerate. This condition means that whenever $x^* \in \mathbb{R}^m$ is a critical point of f, then the Hessian matrix $\nabla^2 f(x^*) = (\partial^2 f/\partial x_i \partial x_j)_{i,j=1,\ldots,m}(x^*)$ is invertible. All critical points of a Morse function are isolated, and hence there are at most countably many of them. Moreover, note that Morse functions are dense. In fact, given any C^2 function g, the function $f(x) = g(x) + < a, x >$ is Morse for a outside a set of Lebesgue's measure 0, by Sard's lemma. More strongly, by using transversality results, it can be shown that the set of all Morse functions is preserved under small perturbations.

The statement of (3) here is equivalent to that of (4) of Theorem 2.1 in Truong and Nguyen [47], but stated in a form more convenient to apply.

In case f is real analytic (or more generally satisfies the Losjasiewicz gradient condition), then without the assumption that f has at most countably many critical points, [1] also showed that the sequence $\{z_n\}$ either diverges to infinity or converges. However, the real analytic assumption is quite restrictive.

In practical applications, we would like the sequence $\{z_n\}$ to converge to a minimum point. It has been shown in Dauphin et al. [13] via experiments that for cost functions appearing in DNN the ratio between minima and other types of critical points becomes exponentially small when the dimension m increases, which illustrates a theoretical result for generic functions Bray and Dean [9]. Which leads to the question: Would in most cases GD converge to a minimum? We will see that because it is indeed a *descent* method, Backtracking GD answers the above question in affirmative in a certain sense. For the sake of clarity, we formalise the notion of "a non-minimum critical point".

Generalised saddle point Let f be a C^1 function and let z_∞ be a critical point of f. Assume that f is C^2 near z_∞. We say that z_∞ is a generalised saddle point of f if the Hessian $\nabla^2 f(z_\infty)$ has at least one *negative* eigenvalue.

If z_∞ is a non-degenerate critical point of f, then it is a minimum if and only if all eigenvalues of the Hessian $\nabla^2 f(z_\infty)$ are positive. Hence in this case we see that a critical point of f is a minimum if and only if it is not a generalised saddle point. See next subsection for a discussion about avoidance of saddle points.

2.2 Comparison to Previous Work

In this subsection we provide a review of convergence results known in the literature. This is a brief version of Sect. 3.4 in Truong and Nguyen [47].

In classical work, starting from the influential paper [2], to prove convergence for both Standard GD and Backtracking GD, it is assumed that f is in $C^{1,1}_L$, the learning rate δ is of order $1/L$, f has compact sublevels (that is, all the sets $\{x : f(x) \leq b\}$ for $b \in \mathbb{R}$ are compact), and the set of critical points of f is bounded and has at most countably many elements. See e.g. Proposition 12.6.1 in Lange [25], and see e.g. Appendix C.12 in Helmke and Moore [19] for analog results for gradient flows (solutions to $x'(t) = -\nabla f(x(t))$). Both the assumptions that f is in $C^{1,1}_L$ and δ is small enough are necessary for the conclusion of Proposition 12.6.1 in Lange [25], even for very simple functions, as shown by Examples 2.14 (for functions of the form $f(x) = |x|^{1+\gamma}$, where $0 < \gamma < 1$ is a rational number) and 2.15 (for functions which

are smooth versions of $f(x) = |x|$) in Truong and Nguyen [47]. In contrast, for these examples, Backtracking GD always converges to the global minimum 0.

Concerning the issue of saddle points, [27,34] proved the very strong result that for f in $C^{1,1}_L$ and $\delta < 1/L$, there exists a set $E \subset \mathbb{R}^k$ of Lebesgue measure 0 so that for $z_0 \in \mathbb{R}^k \backslash E$, if the sequence $\{z_n\}$ is constructed from Standard GD converges, then the limit is not a saddle point. The main idea is that then the map $x \mapsto x - \delta \nabla f(x)$ is a diffeomorphism, and hence we can use the Stable-Center manifold theorem in dynamical systems (cited as Theorem 4.4 in Lee et al. [27]). For to deal with the case where the set of critical points of the function is uncountable, the new idea in Panageas and Piliouras [34] is to use Lindelöff lemma that any open cover of an open subset of $\mathbb{R}^m$ has a countable subcover. Note that here the convergence of $\{z_n\}$ (more generally, one can assume only that the sequence eventually belongs to a small enough open neighbourhood of a generalised saddle point—but this assumption could be difficult to make precise in practice if one does not know details about the function) is important, otherwise one may not be able to use the Stable-Center manifolds. However, for convergence of $\{z_n\}$, one has to use results about convergence of Standard GD (such as Proposition 12.6.1 in Lange [25]), and needs to assume more, as seen from the functions $f(x) = |x|^{1+\gamma}$ in the previous paragraph. Avoidance of saddle points for diminishing learning rates scheme is also available now [35]. Theorem 2.3 in Truong and Nguyen [47] addresses an affirmative answer to a weak version of avoiding saddle points. Note also that as we mentioned in the Introduction—based on results and ideas in Truong and Nguyen [47]—new versions of Backtracking GD are proposed in Truong [44] and shown to be able to avoid saddle points, under assumptions more general than those required by Lee et al. [27], Panageas and Piliouras [34]) for Standard GD. Furthermore, extensions to the infinitely dimensional Banach space setting are available in Truong [46].

There are other variants of GD which are regarded as state-of-the-art algorithms in DNN such as MMT and NAG (mentioned in the Introduction and with more details in Sect. 3.1), Adam, Adagrad, Adadelta, and RMSProp (see an overview in Ruder [38]). Some of these variants (such as Adagrad and Adadelta) allow choosing learning rates δ_n in some complicated manners. However, as far as we know, convergence for such methods are not yet available beyond the usual setting such as in Proposition 12.6.1 in Lange [25].

Stochastic GD is the default method used to justify the use of GD in large scale optimisation and with randomness, which goes back to Robbins and Monro, see [7]. The most common version of it is to assume that we have a fixed cost function F (as in the deterministic case), but we replace the gradient $\nabla_\kappa F(x_n)$ by a random vector v_n (here the random variables are points in the dataset, see also Inexact GD), and then show the convergence in probability of the sequence of values $F(x_n)$ and of gradients $\nabla F(x_n)$ to 0 (in application in the mini-batch setting the random vector v_n will be an approximation of $\nabla_\kappa F_{I_n}(\kappa_n)$). However, the assumptions for these convergence results (for $F(x_n)$ and $\nabla F(x_n)$) to be valid still require those in the usual setting as in Proposition 12.6.1 in Lange [25], in particular requiring that $f \in C^{1,1}_L$ and the learning rate is small compared to $1/L$. In the case where there is noise, the following

additional conditions on the learning rates are needed [37]:

$$\sum_{n\geq 1}\delta_n = \infty,\ \ \sum_{n\geq 1}\delta_n^2 < \infty. \tag{8}$$

These conditions are commonly known as Diminishing learning rates in the literature, and the best result in this direction is contained in Bertsekas and Tsitsiklis [6], where the condition $\sum_{n\geq 1}\delta_n^2 < \infty$ is replaced by the weaker one: $\lim_{n\to\infty}\delta_n = 0$, but where we consider only point wise convergence for ∇f. In the general case, then (8) is needed, and it is generally not an optimal choice since if the sequence $\{x_n\}$ converges to a point at which ∇f is locally Lipschitz, then we can choose δ_n to be uniformly bigger than 0, see Subsect. 3.2 for a more detailed discussion. Convergence of $\{x_n\}$ itself is only proven when the cost function is strongly convex. By the row for SGD in Table 2 on CIFAR10 with Resnet18, one sees that when learning rates become smaller than a threshold, then the validation accuracy becomes worse, which serves as an indication that diminishing learning rates scheme could perform not very well in practice.

When using GD in large scale optimisation, even when the underlying function F is in $C_L^{1,1}$, it may be difficult to obtain a good lower bound estimate for the Lipschitz constant L. Hence it can be difficult to obtain a good choice for the learning rate δ_0. This is more so with the mini-batch practice. The common practice in DNN is to manually fine-tune learning rates, grid search or some other non-rigorous heuristics [32]. However, this practice is very time-consuming (especially when working with large datasets and/or complicated architectures) and depending too much on the researcher's experience. In contrast, Backtracking GD is automatic.

Remarks Recently, there are some work estimating the global Lipschitz constant L for the gradient of cost functions coming from some specific DNN architectures, see [15, 40]. Besides what mentioned in the previous paragraphs, here we note a couple more remarks. First, even if ∇f is globally Lipschitz continuous with Lipschitz constant L, the local Lipschitz constant $L(x)$ for ∇f near a critical point x may be much smaller than L, and hence using the learning rate δ_0 in the order of $1/L$ may be too small compared to what allowed $1/L(x)$. To this end, we note that Backtracking GD can obtain good upper estimates for the local Lipschitz constant $L(x)$, see Sect. 3.2. Beyond this, recall that a DNN is only an attempt to approximate an unknown function, which may not be in $C_L^{1,1}$, it is wise to work with as most general cost function as possible.

Wolfe's method Wolfe's method is very close to Backtracking GD, originating from Wolfe [51]. In the modern literature, Wolfe's conditions are the following

$$\begin{aligned} f(z_n - \delta_n v_n) - f(z_n) &\leq -c_1\delta_n < \nabla f(z_n), v_n >,\\ < \nabla f(z_n - \delta_n v_n), v_n > &\leq c_2 < \nabla f(z_n), v_n >, \end{aligned}$$

for some fixed constants $1 > c_2 > c_1 > 0$. The first condition is exactly Armijo's theorem (condition (iii) in Wolfe's paper). The second condition is only a half of

condition (iv) in Wolfe's paper. It has been shown that if f is a C^1 function which is *bounded from below*, then a positive δ_n can be chosen to satisfy these Wolfe's conditions. Moreover, if f is in $C_L^{1,1}$ and v_n satisfies condition (i) in Inexact GD, then a result by G. Zoutendijk shows the convergence of $\nabla f(z_n)$ to 0. For more details, the readers can consult [33]. Therefore, under the assumptions mentioned above (that is f is in $C_L^{1,1}$, bounded from below, and v_n satisfies condition (i) in Inexact GD), by combining with Remark 2.2 in Truong and Nguyen [47], we can prove conclusions of Theorem 2.1 with Armijo's rule replaced by Wolfe's conditions. As far as we know, this convergence result has not been known before our work. Recently, Wolfe's conditions are implemented in DNN [29]. The paper [4] shows that Wolfe's method fails even for such simple functions as $f(x_1, \ldots, x_k) = a|x_1| + \sum_{j\geq 2} x_j$, if the constant a is big enough.

3 New Gradient Descent Algorithms in Large Scale Optimisation

Here we present new algorithms, aiming at better implementations and performances in large scale optimisation, and prove their theoretical properties.

3.1 Backtracking Versions of MMT and NAG

While saddle points are in general not a problem for both Standard GD and Backtracking GD (see Sect. 2), it is not rare for these algorithms to converge to bad local minima. MMT and NAG (see e.g. [38]) are popular methods designed to avoid bad local minima.

For the standard version of MMT, we fix two numbers $\gamma, \delta > 0$, choose two initial points $z_0, v_{-1} \in \mathbb{R}^m$ and use the following update rule:

$$\begin{aligned} v_n &= \gamma v_{n-1} + \delta \nabla f(z_n), \\ z_{n+1} &= z_n - v_n. \end{aligned}$$

The standard version of NAG is a small modification of MMT: we fix again two numbers $\gamma, \delta > 0$, choose two initial points $z_0, v_{-1} \in \mathbb{R}^m$, and use the update rule:

$$\begin{aligned} v_n &= \gamma v_{n-1} + \delta \nabla f(z_n - \gamma v_{n-1}), \\ z_{n+1} &= z_n - v_n. \end{aligned}$$

If $\gamma = 0$, both MMT and NAG reduce to Standard GD. While observed to work quite well in practice, the convergence of these methods are not proven for functions which are not in $C_L^{1,1}$ or not convex. For example, the proof for convergence of NAG in Sect. 2.2 in Nesterov [30] requires that the function f is in $C_L^{1,1}$ and is moreover strongly convex. (For this class of functions, it is proven that NAG achieves the best possible convergence rate among all gradient descent methods.) Therefore, it is seen that convergence results for these methods require even stronger assumptions than that of Standard GD, see Subsect. 2.2.

Here, inspired by the Inexact Backtracking GD, we propose the following backtracking versions of MMT and NAG, whose convergence can be proven for more general functions. As far as we know, these backtracking versions are new.

Backtracking MMT We fix $0 < A_1 < 1 < A_2$; $\delta_0, \gamma_0 > 0$; $1 \geq \mu > 0$ and $1 > \alpha, \beta > 0$, and choose initial points $z_0, v_{-1} \in \mathbb{R}^m$. We construct sequences $\{v_n\}$ and $\{z_n\}$ by the following update rule:

$$v_n = \gamma_n v_{n-1} + \delta_n \nabla f(z_n),$$
$$z_{n+1} = z_n - v_n.$$

Here the values γ_n and δ_n are chosen in a two-step induction as follows.

Algorithm for Backtracking MMT:

Step 1 Choose $\sigma = 1$, $\gamma' = \gamma_0$, $\delta' = \delta_0$ and $v_n = \gamma' v_{n-1} + \delta' \nabla f(z_n)$ and $z_{n+1} = z_n - \sigma v_n$.

Step 2 If condition (i) for Inexact Backtracking GD is not satisfied, update $\gamma' \leftarrow \gamma'\beta$ and update v_n and z_{n+1} correspondingly.

Step 3 Iterate Step 2 until condition (i) for Inexact Backtracking GD is satisfied. At that stage, obtain γ'_n and δ'_n.

Step 4 If condition (ii) in Inexact Backtracking GD is not satisfied, update $\sigma \leftarrow \sigma\beta$ and update v_n and z_{n+1} correspondingly.

Step 5 Iterate Step 4 until condition (ii) in Inexact Backtracking GD is satisfied. At that stage, obtain σ_n. Then choose $\gamma_n = \sigma_n \gamma'_n$ and $\delta_n = \sigma_n \delta'_n$.

Backtracking NAG The update rule is similar to that for Backtracking MMT. We need only to take care that the term γ appear also in the argument for ∇f.

If $\gamma_0 = 0$, both Backtracking MMT and Backtracking NAG reduce to Backtracking GD. We have the following result for the convergence of these versions.

Theorem 3.1 *Let $f : \mathbb{R}^m \to \mathbb{R}$ be in C^1. Let $\{z_n\}$ be a sequence constructed by either the Backtracking MMT update rule or the Backtracking NAG update rule.*

(1) *Either* $\lim_{n\to\infty} f(z_n) = -\infty$ *or* $\lim_{n\to\infty} ||z_{n+1} - z_n|| = 0$.
(2) *Assume that f has at most countably many critical points. Then either* $\lim_{n\to\infty} ||z_n|| = \infty$ *or $\{z_n\}$ converges to a critical point of f.*
(3) *More generally, assume that all connected components of the set of critical points of f are compact. Then either* $\lim_{n\to\infty} ||z_n|| = \infty$ *or $\{z_n\}$ is bounded. Moreover, in the latter case the set of cluster points of $\{z_n\}$ is connected.*

Proof In fact, our new algorithms satisfy conditions for Inexact Backtracking GD, hence Theorem 2.1 can be applied to yield the desired conclusions (each part of Theorem 2.1 implies the corresponding part with the same numbering in Theorem 3.1). Indeed, this can be seen verbatim for parts 1 and 2. For part 3: in this case, if we do not have $\lim_{n\to\infty} ||z_n|| = \infty$, then the cluster points of $\{z_n\}$ must meet one connected component of the set of critical points of f, and by our assumption here such a component is compact. Hence, all conditions for applying part 3 in Theorem 2.1 are satisfied, and we obtain the desired conclusion. □

Through many experiments, even if the critical points of the function f are non-isolated, it seems that part 3 of Theorem 3.1 (and similarly part 3 of Theorem 2.1) can

be made stronger into saying that either $\lim_{n\to\infty} ||z_n|| = \infty$ or $\{z_n\}$ converges, just like part 2, but we have no proof yet. For theoretical support, we recall the discussion after the statement of Theorem 2.1 if f is real analytic (or more generally satisfies the Losjasiewicz gradient condition). Like in the case of Backtracking GD as in Theorem 2.1, we see that a pathological scenario—not covered by part 3 of Theorem 3.1—when $\{z_n\}$ contains both a bounded and an unbounded subsequences can only happen if either: (a) the set of critical points of f has an unbounded connected component; or (b) the learning rates δ_n are unbounded. While if there are no constraints on the learning rates then it is easy to construct such pathological examples, we will in Subsect. 3.1 show that if the learning rates are allowed to grow but constrained by $||\nabla f(x_n)||^{-1}$ and by Armijo's condition, then good properties as in Theorems 2.1 and 3.1 are preserved. Further, Theorem 3.5 puts an upper bound, at least for Morse's functions, on learning rates in Armijo's condition if we want the constructed sequence to converge. We will also argue heuristically that Armijo's condition and the backtracking manner of choosing learning rates seem to be enough to prevent the mentioned pathological scenario even for functions which are not Morse.

3.2 A Heuristic Argument for the Effectiveness of Standard GD

This subsection is developed from Section 2.4 in Truong and Nguyen [47]. We also add further remarks on rate of convergence for Backtracking GD and that Backtracking GD can obtain good upper estimates for local Lipschitz constants of the gradient ∇f, even for functions not in $C^{1,1}_L$.

Let us consider a function $f : \mathbb{R}^m \to \mathbb{R}$ of class C^1, a number $\delta_0 > 0$ and a point $z_0 \in \mathbb{R}^m$. For simplicity, assume also that f has only at most countably many critical points (which is the generic case), and ∇f is locally Lipschitz at every critical point of f (a reasonable assumption). By Theorem 2.1, if the sequence $\{z_n\}$ in (7) does not contain a sequence diverging to ∞, then it will converge to a critical point z_∞ of f. That ∇f is locally Lipschitz near z_∞ implies that $\delta(f, \delta_0, z) > 0$ uniformly near z_∞. Therefore, since $\delta(f, \delta_0, z)$ takes values in the discrete set $\{\beta^n \delta_0 : \ n = 0, 1, 2, \ldots\}$, it is contained in a finite set. (*Note* The argument until this point is not heuristic, but rigorous.) Hence the number $\delta_\infty := \inf_{n=1,2,\ldots} \delta_n > 0$. Therefore, the Standard GD with learning rate $\delta_0 = \delta_\infty$ should converge.

Our experiments with the data set CIFAR10 and CIFAR100 and various different architectures, more details in the next subsection and Sect. 4, show that this argument is also verified for the practice of using mini-batches in DNN. If $\nabla F_h(\kappa_h)$ approximates well $\nabla F(\kappa_h)$ and $\delta(F_h, \delta_0, \kappa_h)$ approximates well $\delta(F, \delta_0, \alpha_h)$, then Theorem 2.1 can be applied to justify the use of (Inexact) Backtracking GD in DNN.

Convergence rate If ∇f is locally Lipschitz continuous (but not necessarily globally Lipschitz continuous, for example f can be C^2), then when the sequence $\{x_n\}$ converges to a critical point x^*, it follows that for learning rates obtained from Backtracking GD we have $\delta(x_n) \geq \beta\alpha/L(x^*)$ and hence we obtain the same convergence rate as in the classical paper [2].

Upper bound estimates for local Lipschitz constants Assume as in the previous paragraph that ∇f is locally Lipschitz continuous and the sequence $\{x_n\}$ converges

to a critical point x^*. Then it is intuitive (and confirmed by experiments on simple functions) that actually $\delta(x_n)$ is in the order of $\alpha/L(x^*)$, and hence $L(x^*)$ is in the order of $1/\delta(x_n)$. See also Theorem 3.5 for a close relation between the Hessian of the function at a non-degenerate critical point and learning rates in Armijo's condition.

3.3 Two-Way Backtracking GD

In this subsection we present a modification of Backtracking GD which aims to save time and computations. We know that if z_n converges to z_∞, and $\delta_0 \geq \delta_n > 0$ are positive real numbers so that $f(z_n - \delta_n \nabla f(z_n)) - f(z_n) \leq -\alpha\delta_n ||\nabla f(z_n)||^2$ for all n, then $f(z_\infty - \delta_\infty \nabla f(z_\infty)) - f(z_\infty) \leq -\alpha\delta_\infty ||\nabla f(z_\infty)||$ for $\delta_\infty = \limsup_{n\to\infty} \delta_n$. Moreover, at least for Morse functions, the previous subsection shows that all the values $\{\delta_n\}$ belong to a finite set. Therefore, intuitively we will save more time by starting for the search of the learning rate δ_n not from δ_0 as in the original Backtracking GD, but from the learning rate $\sigma = \delta_{n-1}$ of the previous step, and allowing increasing σ, and not just decreasing it, in case σ satisfies inequality (7) and still does not exceed δ_0. We call this Two-way Backtracking GD.

Algorithm for choosing $\delta(z_n)$ *in Two-way Backtracking GD:*

Step 1: choose $\sigma = \delta_{n-1}$.

Step 2: If σ does not satisfy (7), then update $\sigma \leftarrow \beta\sigma$ until Armijo's condition (7) is satisfied.

Step 3: If, on the other hand, σ satisfies (7) then:

Step 3.1: If $\sigma/\beta > \delta_0$, then STOP, and choose $\delta(z_n) = \sigma$.

Step 3.2: If, on the other hand, $\sigma/\beta \leq \delta_0$, then update $\sigma \leftarrow \sigma/\beta$.

Step 3.2.1: Iterate Step 3.2 until $\sigma/\beta > \delta_0$. Then STOP, and choose $\delta(z_n) = \sigma$.

Theorem 2.1 can be used to justify that this procedure should also provide convergence for the sequence $\{z_n\}$. In fact, we have the following result concerning convergence for Two-way Backtracking GD, under assumptions (the same as those required in Theorem 1.3 in Truong [44] about avoiding saddle points) including the case where f is in C^2 or $C^{1,1}_L$.

Theorem 3.2 *Let $f : \mathbb{R}^k \to \mathbb{R}$ be a C^1 function. Assume that there are continuous functions $r, L : \mathbb{R}^k \to (0, \infty)$ such that for all $x \in \mathbb{R}^k$, the gradient ∇f is Lipschitz continuous on $B(x, r(x))$ with Lipschitz constant $L(x)$. Then all conclusions in Theorem 2.1 are satisfied for the sequence $\{z_n\}$.*

Proof As observed in the proof of Theorem 2.1 (see statement in the previous section) in Truong and Nguyen [47], the key is to prove the following two statements:

(i) If $K \subset \mathbb{R}^k$ is a compact set, then $\inf_{n:\, z_n \in K} \delta(z_n) > 0$. This is satisfied under our assumption, since in fact it can be checked that $\delta(z_n) \geq \min\{\beta/L(z_n), \beta r(z_n)/||\nabla f(z_n)||, \delta_0\}$ and hence

$$\delta(z_n) \geq \inf_{z\in K} \min\{\beta/L(z), \beta r(z)/||\nabla f(z)||, \delta_0\}$$

for all n, and that all functions $r, L, ||\nabla f||$ are continuous, and $r, L > 0$. Then we have that any cluster point of $\{z_n\}$ is a critical point of f.

(ii) $\sup_n \delta(z_n) < \infty$. This is satisfied automatically since by construction $\delta(z_n) \leq \delta_0$ for all n. □

The next example illustrates the advantage of Two-way Backtracking GD, in the deterministic case, compared to the Standard GD and the pure Backtracking GD.

Example 3.3 (Mexican hats) The Mexican hat example in Absil et al. [1] is as follows (see Equation 2.8 therein). In polar coordinates $z = (r, \theta)$, the function has the form $f(r, \theta) = 0$ if $r \geq 1$, and when $r < 1$ it has the form

$$f(r, \theta) = \left[1 - \frac{4r^4}{4r^4 + (1 - r^2)^4} \sin\left(\theta - \frac{1}{1 - r^2}\right)\right] e^{-1/(1-r^2)}.$$

For this function, [1] showed that if we start with an initial point $z_0 = (r_0, \theta_0)$, where $\theta_0(1 - r_0^2) = 1$, then the gradient descent flow $(r(t), \theta(t))$ (solutions to $x'(t) = -\nabla f(x(t))$) neither diverges to infinity nor converges as $t \to 0$.

We have run many experiments with Standard GD, pure Backtracking GD and Two-way Backtracking GD for random choices of initial values z_0's. We found that in contrast to the case of the continuous method, all these three discrete methods *do* converge. Here is an explanation: For the continuous method, the gradient descent flow $(r(t), \theta(t))$, with an initial point z_0 on the curve $\theta(1 - r^2) = 1$ will get stuck on this curve, while for the discrete methods right after the first iterate we already escape this curve.

For this Mexican hat example, we observe that Two-way Backtracking GD works much better than Standard GD and pure Backtracking GD. In fact, for the case where $z_0 = (r_0, \theta_0)$ satisfies $\sin(\theta_0) < 0$, for a random choice of initial learning rate Standard GD needs many more iterates before we are close to the limit point than Two-way Backtracking GD, and pure Backtracking GD needs more total time to run and less stable than Two-way Backtracking GD.

In other experiments, we see that the performance of Backtracking GD and Two-way Backtracking GD are almost identical, while the time spent for Backtracking GD is about double (or more) that of time to run Two-way Backtracking GD. Thus, this confirms our intuition that Two-way Backtracking GD helps to save time.

3.4 Unbounded Backtracking GD

Here we define a version of Backtracking GD where learning rates $\{\delta_n\}$ are allowed to be unbounded, and prove the corresponding convergent results. Remark that Unbounded versions of Inexact Backtracking GD, Backtracking Momentum and Backtracking NAG can also be defined similarly, together with corresponding convergent results. We will also discuss on a more general setting where Armijo's condition is kept but with no constraints on how big the learning rates can grow.

Unbounded Backtracking GD Let f be a C^1 function. Fix $0 < \alpha, \beta < 1$ and $\delta_0 > 0$. We choose $\delta(x)$ as in the Backtracking GD procedure. Fix a function $h : (0, \infty) \to (0, \infty)$ such that $\lim_{t \to 0} th(t) = 0$. We choose $\hat{\delta}(x)$ any function satisfying

$\delta(x) \leq \hat{\delta}(x) \leq h(||\nabla f(x)||)\}$ and Armijo's condition $f(x - \hat{\delta}(x)\nabla f(x)) - f(x) \leq -\alpha\hat{\delta}(x)||\nabla f(x)||^2$, for all $x \in \mathbb{R}^k$. Choose a random point x_0. The update rule for Unbounded Backtracking GD is as follows:

$$x_{n+1} = x_n - \hat{\delta}(x)\nabla f(x).$$

We have the following result, which is a generalisation of Theorem 2.1 in Truong and Nguyen [47]. Likewise, its Inexact version is a generalisation of Theorem 2.1.

Theorem 3.4 *Assume that f is C^1, and $\{x_n\}$ is a sequence constructed by the Unbounded Backtracking GD procedure. Then:*

(1) Any cluster point of $\{x_n\}$ is a critical point of f.
(2) Either $\lim_{n\to\infty} f(x_n) = -\infty$ or $\lim_{n\to\infty} ||x_{n+1} - x_n|| = 0$.
(3) Assume that f has at most countably many critical points. Then either $\lim_{n\to\infty} ||x_n|| = \infty$ or $\{x_n\}$ converges to a critical point of f.
(4) More generally, assume that the set of critical points of f has a bounded connected component A. Let B be the set of cluster points of $\{x_n\}$. If $B \cap A \neq \emptyset$, then B is connected and $B \subset A$.

Proof (1) Let K be a compact set for which $\inf_{x\in K} ||\nabla f(x)|| > 0$. Then $\inf_{x\in K} \hat{\delta}(x) \geq \inf_{x\in K} \delta(x) > 0$, the latter can be shown as in Truong and Nguyen [47]. Having this property, we can prove as in Truong and Nguyen [47] (see [5]) that any cluster point of $\{x_n\}$ is a critical point of f.

(2) By Armijo's condition we have

$$f(x_{n+1}) - f(x_n) \leq -\alpha\hat{\delta}(x_n)||\nabla f(x_n)||^2$$

for all n. Hence either $\lim_{n\to\infty} f(x_n) = -\infty$ or $\lim_{n\to\infty} f(x_n)$ exists as a finite number. In that case, summing over n, we obtain the well-known estimate (see [36]):

$$\sum_n \hat{\delta}(x_n)||\nabla f(x_n)||^2 < \infty.$$

In particular, $\lim_{n\to\infty} \hat{\delta}(x_n)||\nabla f(x_n)||^2 = 0$. (In fact, the latter fact can be derived directly, using only that $\lim_{n\to\infty}(f(x_{n+1}) - f(x_n)) = 0$. This will be used in the proof of Theorem 3.8 below).

For any $\epsilon > 0$, we consider 2 sets: $C_1(\epsilon) = \{n \in \mathbb{N} : ||\nabla f(x_n)|| \leq \epsilon\}$ and $C_2(\epsilon) = \{n \in \mathbb{N} : ||\nabla f(x_n)|| > \epsilon\}$. For $n \in C_1(\epsilon)$, using the assumption that $\lim_{t\to\infty} th(t) = 0$ and that $\hat{\delta}(x_n) \leq h(||\nabla f(x_n)||)$, we obtain that

$$||x_{n+1} - x_n|| = \hat{\delta}(x_n)||\nabla f(x_n)|| \leq h(||\nabla f(x_n)||)||\nabla f(x_n)||,$$

must be small when ϵ is small.

For $n \in C_2(\epsilon)$, we have

$$||x_{n+1} - x_n|| = \hat{\delta}(x_n)||\nabla f(x_n)|| \leq \hat{\delta}(x_n)||\nabla f(x_n)||^2/\epsilon,$$

which—for a fixed $\epsilon > 0$—is small when n large enough, because $\lim_{n\to\infty} \hat{\delta}(x_n)$ $||\nabla f(x_n)||^2 = 0$.

Combining these estimates, we obtain: $\lim_{n\to\infty} ||x_{n+1} - x_n|| = 0$.

3) and 4) follows from 1) and 2) by using the real projective space $\mathbb{P}^k$ as in Truong and Nguyen [47], by using a result on convergence in compact metric spaces in [3]. □

If we assume that the sequence $\{z_n\}$ converges, then $\lim_{n\to\infty} ||z_{n+1} - z_n|| = 0$, and hence $\lim_{n\to\infty} \delta_n ||\nabla f(z_n)|| = 0$. Thus the condition $\lim_{t\to 0} th(t) = 0$ is best possible if we want the process to converge. (On the other hand, if one chooses δ_n too small, so to make the condition $\lim_{n\to\infty} \delta_n ||\nabla f(x_n)|| = 0$, then the limit point—if exists—may not be a critical point.)

If z_n is near a critical point z_∞ where the gradient is very flat, for example ∇f is Lipschitz continuous near z_∞ with a very small Lipschitz constant $L(z_\infty)$, then the update $z_{n+1} = z_n - \delta_n \nabla f(z_n)$ is very small when δ_n is bounded. However, here we can take $\hat{\delta}(z_n)$ in the order of $1/L(z_\infty)$, which is big, and make big steps and maybe can escape the point z_∞ and go to another better critical point. (Of relevance is the Capture Theorem in Bertsekas [5] which asserts that if z_0 is close enough to a local minimum and the learning rates are bounded, then the sequence $z_{n+1} = z_n - \delta_n \nabla f(z_n)$ cannot escape z_0.)

One way to make a discrete construction of Unbounded Backtracking GD is as follows. At Step n, we choose $\hat{\delta}(z_n)$ by starting with $\delta = \delta_0$. If δ does not satisfy Armijo's condition, then we reduce it by a factor of β as in the basic version of Backtracking GD. On the other hand, if δ does satisfy Armijo's condition, we multiply it by $1/\beta$ while Armijo's condition and $\delta \leq h(||\nabla f(z_n)||)$ are both still satisfied. $\hat{\delta}(z_n)$ is the final value of δ. This construction is similar to Two-way Backtracking GD, where the only differences are that in Two-way Backtracking GD we start with $\delta = \delta(z_{n-1})$, and we bound δ not by $h(||\nabla f(z_n)||)$ but δ_0. In addition to Theorem 3.2, this similarity between Two-way Backtracking GD and Unbounded Backtracking GD together with Theorem 3.4 provide more support to the convergence of Two-way Backtracking GD in general.

The condition $\delta_n(x) \leq h(||\nabla f(z_n)||)$ in Unbounded Backtracking GD is a way to constrain the learning rate, and as we explained covers all possibilities. The second condition in Wolfe's conditions also serves the same purpose of constraining the learning rate, but is not known to cover all possibilities, and currently can only work well for $C^{1,1}_L$ functions.

Next, we will present theoretical and experimental results as well as heuristic arguments which complement the discussions and results presented so far. First is a result in a more general setting which as a simple consequence roughly says that in case of convergence to a non-degenerate critical point then the behaviour of Unbounded Backtracking GD and the usual Backtracking GD are similar (and this also applies for Wolfe's method), for example when the cost function is Morse. Then experiments are presented to illustrate this result, as well as that Unbounded Backtracking GD and Backtracking GD can perform sharply different in case of convergence to a degenerate critical point. Finally, we present a heuristic argument showing that if learning rates

are chosen in the backtracking manner and so that Armijo's condition is satisfied, then the pathological scenario mentioned in Subsect. 3.1 could be prevented.

Theorem 3.5 *Fix $0 < \alpha < 1$. Choose a sequence of positive numbers δ_n and define a sequence $x_{n+1} = x_n - \delta_n \nabla f(x_n)$. Assume that Armijo's condition $f(x_{n+1}) \leq f(x_n) - \alpha\delta_n ||\nabla f(x_n)||^2$ is satisfied for all n, and that the sequence $\{x_n\}$ converges to a non-degenerate critical point x_∞. To avoid triviality, we assume moreover that $\nabla f(x_n) \neq 0$ for all n. Then for every $\epsilon > 0$, there is n_ϵ so that for all $n \geq n_\epsilon$ we have*

$$\alpha\delta_n \leq \frac{1}{2}(||\nabla^2 f(x_\infty)|| + \epsilon) \times (||\nabla^2 f(x_\infty)^{-1}|| + \epsilon)^2.$$

Proof Fix $\epsilon > 0$. We have that $\{f(x_n)\}$ decreases to $f(x_\infty)$. Hence, by Armijo's condition we have

$$0 \leq f(x_{n+1}) - f(x_\infty) \leq f(x_n) - f(x_\infty) - \alpha\delta_n ||\nabla f(x_n)||^2,$$

for all n. Therefore, for all n, we have $\alpha\delta_n ||\nabla f(x_n)||^2 \leq f(x_n) - f(x_\infty)$.

By Taylor's expansion for f near x_∞, using that f is C^2 and noting that $\nabla f(x_\infty) = 0$, we have (here $o(.)$ is the small-O notation)

$$\begin{aligned} f(x_n) - f(x_\infty) &= \frac{1}{2} < \nabla^2 f(x_\infty)(x_n - x_\infty), x_n - x_\infty > + o(||x_n - x_\infty||^2) \\ &\leq \frac{1}{2}||\nabla^2 f(x_\infty)|| \times ||x_n - x_\infty||^2 + o(||x_n - x_\infty||^2). \end{aligned}$$

Hence, if n is large enough then $f(x_n) - f(x_\infty) \leq \frac{1}{2}(||\nabla^2 f(x_\infty)|| + \epsilon) \times ||x_n - x_\infty||^2$

By Taylor's expansion for ∇f near x_∞, using again that f is C^2 and noting that $\nabla f(x_\infty) = 0$, we have

$$\nabla f(x_n) = \nabla^2 f(x_\infty)(x_n - x_\infty) + o(||x_n - x_\infty||).$$

Hence, multiplying both sides with $\nabla^2 f(x_\infty)^{-1}$, when n is large enough, we get $||x_n - x_\infty|| \leq (||\nabla^2 f(x_\infty)^{-1}|| + \epsilon)||\nabla f(x_n)||$.

Putting together all the above estimates and cancelling the term $||\nabla f(x_n)||^2$ at the end, we obtain finally:

$$\alpha\delta_n \leq \frac{1}{2}(||\nabla^2 f(x_\infty)|| + \epsilon) \times (||\nabla^2 f(x_\infty)^{-1}|| + \epsilon)^2,$$

for large enough values of n, as wanted. □

This result says roughly that in case of convergence to a non-degenerate critical point, then the performance of Unbounded Backtracking GD and of the usual Backtracking GD are similar. Since Wolfe's condition includes Armijo's condition, the same applies to Wolfe's condition. On the other hand, in case of convergence to a degenerate

critical point, then the performance of Unbounded Backtracking GD and Backtracking GD can be sharply different. Below are some experimental results illustrating this comment.

The setups are as follows. We choose $\alpha = 0.5$ for Armijo's condition.

For the usual Backtracking GD, we choose $\beta = 0.7$ and $\delta_0 = 1$.

For Unbounded Backtracking GD: we choose $\beta = 0.7$ and $\delta_0 = 1$ as in the usual Backtracking GD. We choose the function $h(t) = \delta_0$ if $t > 1$, and $h(t) = \delta_0/\sqrt{t}$ if $t \leq 1$. For the readers' convenience, we recall here the update rule for Unbounded Backtracking GD: At step n, we start with $\delta = \delta_0$. If δ does not satisfy Armijo's condition, then we reduce δ by $\delta\beta$ until it satisfies Armijo's condition, hence in this case we proceed as in the usual Backtracking GD. On the other hand, if δ does satisfy Armijo's condition, then we increase it by δ/β while both Armijo's condition and $\delta \leq h(||\nabla f(x_n)||)$ is satisfied. We choose δ_n to be the final value of δ, and update $x_{n+1} = x_n - \delta_n \nabla f(x_n)$.

We will stop when either the iterate number is 10^6 or when the gradient of the point is $\leq 10^{-10}$.

Example 3.6 We look at the function $f(x, y) = x^3 sin(1/x) + y^3 sin(1/y)$ and start from the initial point $z_0 = (4, -5)$. After 10 steps, both algorithms Backtracking GD and Unbounded Backtracking GD arrive at the same point $(0.09325947, -0.09325947)$ which is very close to a non-degenerate local minimum of the function.

Example 3.7 We look at the function $f(x, y) = x^4 + y^4$ and start from the initial point $z_0 = (0.1, 15)$. This function has a degenerate global minimum at $(0, 0)$. After 10^6 steps, Backtracking GD arrives at the point $(0.00111797, 0.00111802)$ with learning rate 1. On the other hand, only after 89 steps, Unbounded Backtracking GD already arrives at a better point $(0.00025327, 0.00025327)$ with learning rate 90544.63441298596 much bigger than 1.

Finally, we present a heuristic argument showing that Armijo's condition and backtracking manner of choosing learning rates could prevent the pathological scenario mentioned at the end of Subsect. 3.1. More precisely, we use the following update rule: it is like the update rule for the discrete version of Unbounded Backtracking GD mentioned above, except that we do not constrain δ by any function $h(||\nabla f(z_n)||)$. The pathological scenario is that the constructed sequence $\{z_n\}$ contains both a bounded and an unbounded subsequence, and the bounded subsequence converges to a critical point z_∞. Since as mentioned, modifications of Backtracking GD in Truong [44,46] can avoid saddle points, we expect that with the above update rule the sequence $\{z_n\}$ can also avoid saddle points. Then the point z_∞ is expected to be a local minimum. There is expected a small open neighbourhood U of z_∞ for which $\min_{z \in \partial U} f(z) > f(z_\infty)$. Now, the backtracking manner of choosing learning rates is expected to have this effect: if $z \in U$ is very close to z_∞, then the choice of $\delta(z)$ - since at most will be increased by β at a time and must keep the value of the function not increased—will not be enough to allow the resulting point $z - \delta(z)\nabla f(z)$ to escape U. (Since $||\nabla f(z_n)||$ is very small, it is expected that if δ' is the largest positive number so that $z_n - \delta' \nabla f(z_n)$ stays in U, then the next value δ'/β is expected to make $z_n - \delta' \nabla f(z_n)/\beta$ stay close to ∂U,

which will force $f(z_n - \delta' \nabla f(z_n)/\beta) > f(z_n)$—a condition prohibited by Armijo's condition.) Therefore, we expect that if there is a sequence $\{z_{n_j}\}$ converging to z_∞, then the whole sequence $\{z_n\}$ must be bounded, and the above pathological scenario cannot happen. It would be good if the above heuristic argument can be rigorously shown at least for C^2 cost functions.

3.5 Some Remarks About Implementation of GD in DNN

In this section we mention some issues one faces when applying GD (in particular, Backtracking GD) to the mini-batch practice in DNN, and also the actual implementations of mini-batch backtracking methods for GD, MMT and NAG.

3.5.1 Rescaling of Learning Rates

Since the cost functions obtained from different mini-batches are not exact even though they may be close, using GD iterations many times can accumulate errors and lead to explosion of errors. If we use the backtracking method for a mini-batch, the obtained learning rate is only optimal in case we continue the training with that mini-batch. Even if we take the average of the learning rates from many batches, using directly this value in GD can cause a lot of noise from covariance between batches. This phenomenon has been observed in practice, and the method of rescaling of learning rates has been proposed to prevent it, in both Standard GD [24] and Wolfe's method [29]. The main idea in these papers is that we should rescale learning rates depending on the size of mini-batches, and a popular choice is to use a linear dependence. Roughly speaking, if δ is a theoretically good learning rate for the cost function of full batch, and the ratio between the size of full batch and that of a mini-batch is $\rho = k/N$ (where $N =$ the size of full batch and $k =$ the size of a mini-batch), then this popular choice suggests to use instead the learning rate δ/ρ.

Here, we propose a new rescaling scheme that the bigger learning rate $\delta/\sqrt{\rho}$ should also work in practice. Our justification comes from the stochastic gradient update rule with Gaussian noise [23]:

$$z_{n+1} = z_n - \delta_n \left(\nabla f(z_n) + \frac{r}{\sqrt{N}} \right), \tag{9}$$

where r is a zero mean Gaussian random variable with covariance of $\nabla f(z_n)$ from whole population of samples. The updating term $\delta_n(\nabla f(z_n) + \frac{r}{\sqrt{N}})$ bears a random variable noise $\delta_n \frac{r}{\sqrt{N}}$. In case of mini-batching, this noise is $\delta_n \frac{r}{\sqrt{k}}$. If we want to maintain the noise level of mini-batch similar to that of full batch, we can simply rescale δ_n by a factor $1/\sqrt{\rho}$. One can notice that using δ/ρ in (9) has the effect of trying to make the training loss/accuracy of using small mini-batch the same as that of using larger mini-batch or full batch after each epoch. This is not essential because what we really need is fast convergence and high accuracy for stochastic optimising, not full batch imitation. A larger learning rate such as $\delta/\sqrt{\rho}$ can help to accelerate the convergence while still keep low noise level. Smith and Topin [42] demonstrated that

networks trained with large learning rates use fewer iterations, improve regularisation and obtain higher accuracies. We have checked with experiments, see below, that this new rescaling scheme works well in practice.

3.5.2 Mini-batch Backtracking Algorithms

While being automatic and having a stable behaviour, Backtracking GD is still not popular compared to Standard GD, despite the latter requiring a lot of efforts for manually fine-tuning.

One reason is that when using Backtracking GD for a mini-batch, we obtain a learning rate which is good for this mini-batch but not for the full batch, as mentioned in the previous Subsection. We have resolved this by using rescaling of learning rates.

Another reason is that Backtracking GD, needing more computations, is much slower than Standard GD. We can resolve by using Two-way Backtracking GD.

Combining the above two ideas, we arrive at a new method which we call Mini-batch Two-way Backtracking GD (MBT-GD), if we regard Backtracking GD as a learning rate finder for Standard GD. The precise procedure is as follows. We only need to apply Backtracking GD a small number of times (say, tens) at the beginning of the training process or periodically at the beginning of each epoch, for several first mini-batches, take the mean value of these obtained learning rates and rescale to achieve a good learning rate, and then switch to Standard GD. The cost for doing this is very small or negligible in the whole optimising process. We can also apply the same idea to obtain Mini-batch Two-way Backtracking MMT (MBT-MMT) and Mini-batch Two-way Backtracking NAG (MBT-NAG).

The above idea is compatible with recent research [50,52], which suggests that traditional GD as well as MMT and NAG with a good learning rate can be on par or even better than Adam and other adaptive methods in terms of convergence to better local minima. The common practice in Deep Learning to find good learning rates is by manually fine-tuning, as mentioned above. Our Mini-batch Two-way Backtracking methods can be used to automatically fine-tune learning rates. Below we describe briefly the details of our mini-batch methods MBT-GD, MBT-MMT and MBT-NAG.

For MBT-GD, we simply compute an optimal learning rate at the beginning of the training process by applying Backtracking GD to several mini-batches (e.g $20-50$ mini-batches), take the mean value of obtained learning rates with rescaling justification and use it in Standard GD for next training iterations. This method is inspired by cyclic learning rates ([41]) and fastai learning rate finder [14,28] (both require manual interference for the learning rate schedule in training model or selecting optimal learning rate), but uses instead backtracking line search method (and hence is automatic). The recommended value for the hyper parameter α is 10^{-4}, which means that we accept most of descent points but use scaling justification to reduce noise effects. This will give larger workable learning rates to accelerate speed and improve regularisation, which is good according to Smith and Topin [42]. We keep using this learning rate value for Standard GD until the training gets stuck (e.g. no loss decreasing in 5 consecutive epochs), at which time we then switch to Backtracking GD as in the beginning but now with $\alpha = 0.5$. Using larger α when being near local minima gives us learning rates of appropriate size to guarantee convergence (otherwise, if learning

rate is too big, then we can leave the critical point, see Example 2.15 in Truong and Nguyen [47]).

The above procedure roughly means that we will run Backtracking GD only a very small fraction of times, and most of the time will run the Standard GD. The following is a theoretical justification for the above procedure. To see the relevance, we note that when we use mini-batching, the mini-batch sizes are fixed by a number N. A version for Two-way Backtracking GD and Backtracking NAG and Backtracking Momentum can also be stated.

Weak-coercive A function f is weak-coercive if whenever $\{f(x_n)\}$ decreases to a finite number and $\liminf_{n\to\infty} ||\nabla f(x_n)|| > 0$, then $\{x_n\}$ is bounded.

The common activation functions in Deep Learning such as sigmoid and ReLU are both weak-coercive. If f is a function bounded from below, then its regularisations $f + \epsilon||x||^q$, for some constants $\epsilon, q > 0$, are all coercive. Note that cost functions in DNN are usually bounded from below, and the practice of regularisation is very common in DNN.

Theorem 3.8 *Let $f : \mathbb{R}^k \to \mathbb{R}$ be C^1 function. Fix a positive integer N. Given an initial point $x_0 \in \mathbb{R}^k$. Assume that we construct a sequence $x_{n+1} = x_n - \delta_n \nabla f(x_n)$ so that for every n we have $f(x_{n+1}) \leq f(x_n)$.*

(1) *Assume that for every n, either δ_n is constructed using Backtracking GD, or $\delta_n = \delta_{n-1}$, and at least one of $\delta_n, \delta_{n+1}, \ldots, \delta_{n+N}$ is updated using Backtracking GD. If f is weak-coercive, then all conclusions of Theorem 2.1 are satisfied for the sequence $\{x_n\}$.*

(2) *Fix $M > 0$. Assume that for every n, either $\delta_n = \delta_{n-1}$ or δ_n is the largest number in $\{\beta^n \delta_0 : n = 0, 1, 2, \ldots\}$ which is $\leq \min\{\delta(x_n), M/||\nabla f(x_n)||\}$ and satisfies Armijo's condition, where $\delta(x_n)$ is the one constructed from Backtracking GD. Moreover, assume that for every n, at least one among $\delta_n, \delta_{n+1}, \ldots, \delta_{n+N}$ is constructed by the latter procedure. Then all conclusions of Theorem 2.1 are satisfied for the sequence $\{x_n\}$.*

Proof (1) The proof of the theorem is similar to that of Theorem 3.2, by observing that since $\{f(x_n)\}$ is decreasing, if a subsequence $\{x_{n_k}\}$ converges, $\lim_{k\to\infty} f(x_{n_k}) > -\infty$ and δ_{n_k} are all updated by the Backtracking GD procedure, then $\lim_{k\to\infty} \nabla f(x_{n_k}) = 0$. Indeed, assume otherwise, then by passing to a subsequence, we can assume that $\liminf_{k\to\infty} ||\nabla f(x_{n_k})|| > 0$, and hence by the weak-coercivity we have that $\{x_{n_k}\}$ is bounded. Then since the learning rates δ_{n_k}'s are updated by Backtracking GD, it follows that $\lim_{k\to\infty} \nabla f(x_{n_k}) = 0$.

For every converging subsequence $\{x_{n_k}\}$, we construce another subsequence $\{x_{n'_k}\}$ where $n'_k \leq n_k$ is the largest number for which $\delta(x_{n'_k})$ is updated by Backtracking GD rule. Then passing to a subsequence again if necessary, we can as in the first paragraph to assume that

$$\lim_{k\to\infty} \nabla f(x_{n'_k}) = 0.$$

We now illustrate the proof in two cases: $n_k = n'_k + 1$ for all k, or $n_k = n'_k + 2$ for all k. The general case is similar.

Case 1 $n_k = n'_k + 1$ for all k. In this case, since $x_{n'_k}$ converges, δ_{n_k} is bounded, $\lim_{k\to\infty} \nabla f(x_{n'_k}) = 0$, and $x_{n'_k} = x_{n_k+1} = x_{n'_k} + \delta_{n'_k} \nabla f(n'_k)$, it follows that $\{x_{n'_k}\}$ converges to the same point as $\{x_{n_k}\}$ and hence

$$\lim_{k\to\infty} \nabla f(x_{n_k}) = \lim_{k\to\infty} \nabla f(x_{n'_k}) = 0.$$

Case 2 $n_k = n'_k + 2$ for all k. We will show again that $\{x_{n'_k}\}$ is bounded and then can proceed as in Case 1. Assume otherwise that $\lim_{k\to\infty} ||x_{n'_k}|| = \infty$, we will obtain a contradiction. Indeed, since $x_{n'_k+1} = x_{n'_k} - \delta_{n'_k} \nabla f(x_{n'_k})$, it follows that $\lim_{k\to\infty} ||x_{n'_k+1}|| = \infty$ also. Since $x_{n_k} = x_{n'_k+2} = x_{n'_k+1} - \delta_{n'_k+1} \nabla f(x_{n'_k+1})$ converges, it follows that $\lim_{k\to\infty} ||\nabla f(x_{n'_k+1})|| = \infty$, and hence by weak-coercivity we must have that $\{x_{n'_k 1+1}\}$ is bounded, which is the desired contradiction.

(2) In this case, when we choose $\{x_{n'_k}\}$ from $\{x_{n_k}\}$ as in 1) for a convergent subsequence $\{x_{n_k}\}$, then

$$||x_{n'_k} - x_{n_k}|| \leq MN,$$

and hence the sequence $\{x_{n'_k}\}$ is bounded. Now using that $\nabla f(x)$ is a continuous function, it can be checked like in the case of Backtracking GD that if K is a compact set for which $\inf_{n:x_n\in K} ||\nabla f(x_n)|| > 0$, then $\inf_{n:x_n\in K} \delta_n > 0$. Then we can proceed like in (1). □

Remark 3.9 The assumption in (2) of Theorem 3.8 can be easily implemented in practice in DNN. Note that the condition covers all possible cases if we want the sequence $\{x_n\}$ converges, since then $\lim_{n\to\infty} \delta_n ||\nabla f(x_n)|| = 0$. It is basically the usual Backtracking GD procedure, the only difference is that here we bound the learning rate by $M/||\nabla f(x)||$ and not by δ_0. Its meaning is that when $||\nabla f(x)||$ is too big, then we should take learning rate smaller than δ_0. On the other hand, when n very big then δ_n chosen this way is the same as the learning rate chosen from the usual Backtracking GD, the reason being that then x_n is close to a critical point and hence $M/||\nabla f(x)||$ is bigger than δ_0. The assumption in (2) is like Backtracking GD with some more local inputs aside from Armijo's condition, very much like the algorithm used in Theorem 1.3 in Truong [44].

A special case of part (2) of Theorem 3.8 is when $\sup_{x\in\mathbb{R}^k} ||\nabla f(x)|| < \infty$. In this case, if we choose $M = \delta_0 \sup_{x\in\mathbb{R}^k} ||\nabla f(x)||$, then the construction of the sequence $\{x_n\}$ in (2) is the same as that in (1). This special case is very much relevant to the current DNN architectures. Indeed, the assumption that $||\nabla f||$ is bounded from above of part (3) is satisfied for common activation functions such as sigmoid, tanh, ReLU, max, softmax, linear functions and so on. Hence, by the chain rule in calculus, this assumption is also satisfied for compositions of such functions. Since a DNN architecture is currently constructed with such functions between layers, this part (3) applies for all output maps h of such DNN. If we assume for example that the output map h is also bounded, then part (3) can be applied also to the cost function of the DNN.

For MBT-MMT and MBT-NAG, we use the same method to compute the optimal learning rate, but now at the beginning of every epoch (or after some fixed iterations

of Standard GD) in order to take advantage of momentum accumulation which is the strong point of MMT and NAG. As in the case of MBT-GD, it is also recommended to use $\alpha = 10^{-4}$ in most of the training process until the training gets stuck (e.g. no loss decreasing in 5 consecutive epochs), at which time we then switch to use $\alpha = 0.5$ and turn off momentum, too. This can help accelerate the convergence at the early stages of the training and take more care about the descent near local minima at later stages of the training. This is very similar to the methods of learning rate warming up and decay, see an illustration in Subsect. 4.1. We also note that when doing experiments, we do not really follow the precise definition of Backtracking MMT and Backtracking NAG as in Sect. 2, which is complicated and which we will explore in more detail in future work. For the experiments here, we use the following simplified algorithm: fix the value of γ to 0.9 (as commonly used in practice) and choose δ_n by Backtracking GD. That is, we seek to find good learning rates in the standard MMT and NAG algorithms by using Backtracking GD. The intuition is that as Backtracking GD can find good learning rates for Standard GD, it can also find good learning rates for MMT and NAG. The experiments, see below, verify this speculation.

For the hyper-parameter β, we could use any value from 0.5 to 0.95. By the very nature of backtracking line search, it is intuitive to see that a specific choice of β does not affect too much the behaviour of Backtracking GD. For the sake of speed, we use $\beta = 0.5$ and the number of mini-batches to apply backtracking line search (at the beginning of training or each epoch) to be 20. This can help the training speed of MBT-MMT and MBT-NAG to be about $80\% - -99\%$ of the training speed of MMT and NAG, depending on the batch size. The trade-off is inexpensive since we do not need to do manually fine-tuning of learning rate (which takes a lot of time and effort).

It is worthy to note that the above settings for MBT-GD, MBT-MMT and MBT-NAG are fixed in all experiments in this section and hence all obtained results come from entirely automatic training without any human intervention.

Remark 3.10 Nowadays, in most of modern DNN, there are dropout and batch normalizations layers designed to prevent overfitting and to reduce internal covariate shift [22,43]. When using Backtracking GD for these layers, more cares are needed than usual, since non-deterministic outcomes of weights can cause unstability for our learning rate fine-tuning method. After many experiments, we find that some specific procedures can help to reduce the undesired effects of these non-deterministic outcomes. For dropout layers, we should turn them off when using Backtracking GD and turn them on again when we switch to Standard GD. For batch normalizations, we need to make sure using the training/testing flags in a consistent way to obtain the right values for Condition (7) and avoid causing non-deterministic and unstable.

4 Experimental Results

In this section we illustrate the effectiveness of the new methods with experiments on the benchmark CIFAR10 and CIFAR100 (image) datasets, using various state-of-the-art DNN models Resnet18 [17], MobileNetV2 [39], SENet [20], PreActResnet18 [18] and Densenet121 [21], showing that—while being automatic—they are better than

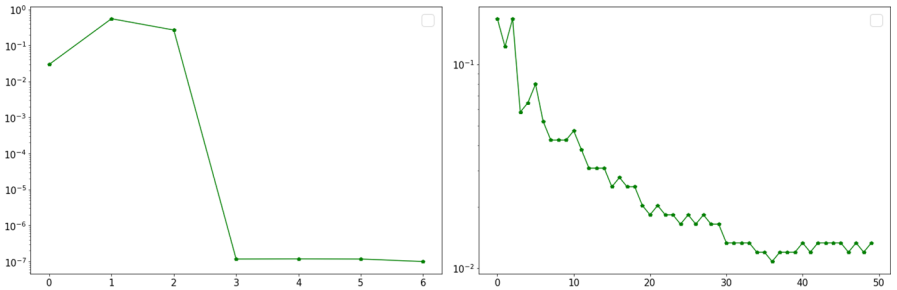


Fig. 2 Learning rate attenuation using Two-way Backtracking GD on (a) Mexican hat function (b) Resnet18 on a dataset contains 500 samples of CIFAR10 (full batch)

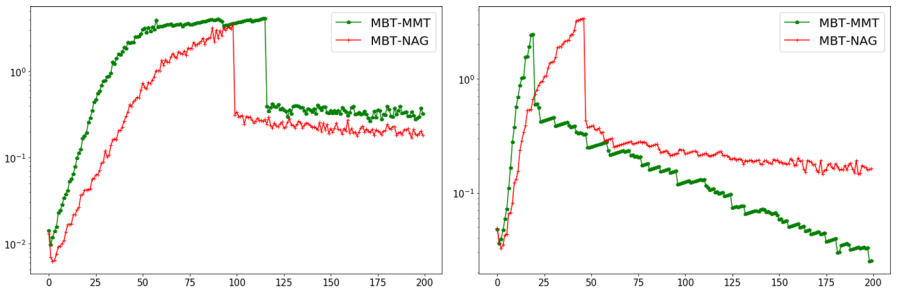
current state-of-the-art methods such as MMT, NAG, Adagrad, Adadelta, RMSProp, Adam and Adamax, see [38]. We also verify the heuristic argument proposed in Sect. 3.

4.1 Experiment 1: Behaviour of Learning Rates for Full Batch

In this experiment we check the heuristic argument in Subsect. 3.2 for a single cost function. We do experiments with Two-way Backtracking GD for two cost functions: one is the Mexican hat in Example 3.3, and the other is the cost function coming from applying Resnet18 on a random set of 500 samples of CIFAR10. See Fig. 2. It appears that for the Mexican hat example, Two-way Backtracking GD stablises to Standard GD in only 6 iterates, while for CIFAR10 stabilising appears after 40 iterates.

4.2 Experiment 2: Behaviour of Learning Rates for Mini-Batches

In this experiment we check the heuristic argument in Subsect. 3.2 in the mini-batch setting. We do experiments with MBT-MMT and MBT-NAG for the model Resnet18 on CIFAR10 and CIFAR100. See Fig. 3. Obtained learning rates behave similarly to the common scheme of learning rate warming up and decay in Deep Learning. The significant decrease of learning rates in the figure is a consequence of changing α from $1e{-}4$ to 0.5, see description in Section 3.5.2 for detail. The experiment also indicates that MBT-NAG stabilises better.



(a) CIFAR10 **(b)** CIFAR100

Fig. 3 Learning rate attenuation using Two-way Backtracking GD in the mini-batch setting on Resnet18 on **a** CIFAR10, **b** CIFAR100

Table 1 Stability of Averagely-optimal learning rate obtained from MBT-GD across 9 different starting learning rates (LR) ranging from 10^{-6} to 100 and 7 different batch sizes from 12 to 800

LR	100	10	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
12	0.0035	0.0037	0.0037	0.0040	0.0046	0.0040	0.0038	0.0036	0.0038
25	0.0050	0.0051	0.0051	0.0058	0.0049	0.0048	0.0052	0.0057	0.0044
50	0.0067	0.0067	0.0065	0.0060	0.0067	0.0061	0.0066	0.0070	0.0075
100	0.0111	0.0101	0.0093	0.0104	0.0095	0.0098	0.0098	0.0099	0.0085
200	0.0140	0.0143	0.0137	0.0147	0.0125	0.0130	0.0135	0.0122	0.0126
400	0.0159	0.0155	0.0167	0.0153	0.0143	0.0174	0.0164	0.0166	0.0154
800	0.0153	0.0161	0.0181	0.0188	0.0170	0.0190	0.0205	0.0154	0.0167

Applied using Resnet18 on CIFAR10. ($\alpha = 10^{-4}$, $\beta = 0.5$)

4.3 Experiment 3: Stability of Learning Rate Finding Using Backtracking Line Search

In this experiment, we apply MBT-GD to the network Resnet18 on the dataset CIFAR10, across 9 different starting learning rates (from 10^{-6} to 100) and 7 different batch sizes (from 12 to 800), see Table 1. With any batch size in the range, using the rough grid $\beta = 0.5$ and only 20 random mini-batches, despite the huge differences between starting learning rates, the obtained averagely-optimal learning rates stabilise into very close values. This demonstrates that our method works robustly to find a good learning rate representing the whole training data.

4.4 Experiment 4: Comparison of Optimisers

In this experiment we compare the performance of our methods (MBT-GD, MBT-MMT and MBT-NAG) with state-of-the-art methods, on the CIFAR10 dataset with Resnet18. See Table 2. We note that MBT-MMT and MBT-NAG usually work much better than MBT-GD, the explanation may be that MMT and NAG escape bad local minima better. Since the performance of both MBT-MMT and MBT-NAG are 1.29%

Table 2 Best validation accuracy after 200 training epochs (batch size 200) of different optimisers using different starting learning rates (MBT methods which are stable with starting learning rate only use starting learning rate 10^{-2} as default)

LR	100	10	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
SGD	10.00	89.47	91.14	*92.07*	89.83	84.70	54.41	28.35	10.00
MMT	10.00	10.00	10.00	*92.28*	91.43	90.21	85.00	54.12	28.12
NAG	10.00	10.00	10.00	*92.41*	91.74	89.86	85.03	54.37	28.04
Adagrad	10.01	81.48	90.61	88.68	*91.66*	86.72	54.66	28.64	10.00
Adadelta	91.07	92.05	*92.36*	91.83	87.59	73.05	46.46	22.39	10.00
RMSprop	10.19	10.00	10.22	89.95	91.12	*91.81*	91.47	85.19	65.87
Adam	10.00	10.00	10.00	90.69	90.62	*92.29*	91.33	85.14	66.26
Adamax	10.01	10.01	91.27	91.81	*92.26*	91.99	89.23	79.65	55.48
MBT-GD	*91.64*								
MBT-MMT	*93.70*								
MBT-NAG	**93.85**								

Italic: Best accuracy of the optimiser in each row. Bold: Best accuracy of all optimisers

Table 3 (Models and datasets) Accuracy of state-of-the-art models after 200 training epochs

Dataset	CIFAR10		CIFAR100	
Optimiser	MBT-MMT	MBT-NAG	MBT-MMT	MBT-NAG
Resnet18	93.70	93.85	68.82	70.78
PreActResnet18	93.51	93.51	71.98	71.53
MobileNetV2	93.68	91.78	69.89	70.33
SENet	93.15	93.64	69.62	70.61
DenseNet121	**94.67**	94.54	73.29	**74.51**

Bold: Best accuracy on each dataset

and 1.45% above the best performance of state-of-the-art methods, it can be asserted that our methods are better than state-of-the-art methods.

4.5 Experiment 5: Performance on Different Datasets and Models and Optimisers

In this experiment, we compare the performance of our new methods (without any change in settings and hyper-parameters) on different datasets and models. We see from Table 3 that our automatic methods work robustly with high accuracies across many different architectures, from light weight models such as MobileNetV2 to complicated architecture as DenseNet121.

5 Conclusions

In this paper we reviewed the important role of gradient descent methods in DNN and previous work on GD, concerning convergence and avoidance of saddle points for

modifications of Backtracking GD, in both finite and infinite dimensions. Then, we propose new gradient descent algorithms (including Backtracking MMT, Backtracking NAG, Two-way Backtracking GD and Unbounded Backtracking GD) in large scale optimisation. The theoretical advantage of our new methods is that convergence can be proven for most C^1 functions, and it can be also used to provide better convergence of Wolfe's method. Unlike other common versions of GD, Unbounded Backtracking GD allows learning rates to be unbounded. There is also a similarity between Unbounded Backtracking GD and Two-way Backtracking GD, and depending on whether the convergence to a non-degenerate or degenerate critical points which the performance of Unbounded Backtracking GD and the usual Backtracking GD can be similar or sharply different. We provided a heuristic argument showing that in the long run, Backtracking GD should stabilise to a finite union of Standard GD processes. We also discuss about convergence rate for Backtracking GD and that Backtracking GD can obtain good upper bounds for local Lipschitz constants of the gradient ∇f. Local conditions such as Armijo's are also easier to adapt to general settings such as manifolds than global conditions such as $C^{1,1}_L$.

Our algorithms (with sufficient theoretical justifications) provide a very good automatic fine-tuning of learning rates. In fact, experiments with the MNIST and CIFAR10 data sets show that the new methods are better than current state-of-the-art methods such as MMT, NAG, Adagrad, Adadelta, RMSProp, Adam and Adamax. The experimental results show that the new methods work very stably, and it seems that among all methods, the new method MBT-NAG usually has the best stable behaviour and performance.

There are many interesting and worthy followup questions. For example, to prove theoretical results in more general settings (such as more general functions or for constrained optimisation). The readers can see some of these in Section 5 of Truong and Nguyen [47]. Here is one question which is of foremost usefulness: To find a stochastic treatment of Backtracking GD, in the same way as Stochastic GD is for Standard GD. Note that Armijo's condition is not treated in the current version of Stochastic GD, and the fact that $\delta(f, \delta_0, x)$ is not continuous (Example 2.17 in Truong and Nguyen [47]) may make it challenging to treat. Since Backtracking GD is more adaptive to data sets, stable and automatic, we speculate that it will be useful in resolving challenges such as adversarial images.

Acknowledgements We are grateful to Geir Dahl, who pointed out relevant references, including [5]. T. T. Truong would like to thank the Mathematics in Industry Study Group meeting in 2016, for inspiration. He also appreciates very much the many discussions with Neeraj Kashyap, and thanks also Terje Kvernes for helping with technical issues. We appreciate the very useful comments and suggestions by anonymous referees. The work of the first author is partially supported by Young Research Talents grant number 300814 of Research Council of Norway, as well as some travel fund from the Trond Mohn Foundation.

Funding Open Access funding provided by University of Oslo (incl Oslo University Hospital).

References

1. Absil, P.-A., Mahony, R., Andrews, B.: Convergence of the iterates of descent methods for analytic cost functions. SIAM J. Optim. **16**(2), 531–547 (2005)
2. Armijo, L.: Minimization of functions having lipschitz continuous first partial derivatives. Pac. J. Math. **16**(1), 1–3 (1966)
3. Asic, M.D., Adamovic, D.D.: Limit points of sequences in metric spaces. Am. Math. Mon. **77**(6), 613–616 (1970)
4. Asl, A., Overton, M.L.: Analysis of the gradient method with an armijo-wolfe line search on a class of nonsmooth convex functions. Optim. Methods Softw. **35**(2), 223–242 (2020)
5. Bertsekas, D.P.: Nonlinear Programming, 2nd edn. Athena Scientific, Belmont, Massachusetts (1999)
6. Bertsekas, D.P., Tsitsiklis, J.N.: Gradient convergence in gradient methods with error. SIAM J. Optim. **10**(3), 627–642 (2006)
7. Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. SIAM Rev. **60**(2), 223–311 (2018)
8. Boyd, S., Vandenberghe, L.: Convex Optimization, 7 rev edn. Cambridge University Press, Cambridge (2009)
9. Bray, A.J., Dean, D.S.: Statistics of critical points of gaussian fields on large-dimensional spaces. Phys. Rev. Lett. **98**, 150201 (2007)
10. Cauchy, A.: Method général pour la résolution des systemes d'équations simulanées. Comptes Rendus **25**(2), 536 (1847)
11. Crockett, J.B., Chernoff, H.: Gradient methods of maximization. Pac. J. Math. **5**, 33–50 (1955)
12. Curry, H.B.: The method of steepest descent for non-linear minimization problems. Q. Appl. Math. **2**(3), 258–261 (1944)
13. Dauphin, Y.N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., Bengjo, Y.: Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. NIPS **2**, 2933–2941 (2014)
14. fastai. https://www.fast.ai/ (2020)
15. Fazlyab, M., Robey, Hassani, H., Morari, M., Pappas, G.J.: Efficient and accurate estimation of lipschitz constants for deep neural networks. arXiv:1906.04893v1 (2019)
16. Goldstein, A.A.: Cauchy's method of minimization. Numer. Math. **4**, 146–150 (1962)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CoRR, arxiv:1512.03385 (2015)
18. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: CoRR, arXiv:1603.05027 (2016)
19. Helmke, U., Moore, J.B.: Optimization and dynamical systems. Online book, 2nd edition (1996)
20. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: CoRR, arXiv:1709.01507 (2017)
21. Huang, G., Liu, Z., Weinberger, K.Q.: Densely connected convolutional networks. In: CoRR, arXiv:1608.06993 (2016)
22. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: 32nd ICML, JMLR, pp. 448–456 (2015)
23. Jastrzebski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y.Y., Storkey, A.J.: Three factors influencing minima in sgd. In: CoRR, arxiv:1711.04623 (2017)
24. Krizhevsky, A.: One weird trick for parallelizing convolutional neural networks. arXiv:1404.5997 (2014)
25. Lange, K.: Optimization. Springer Texts in Statistics, 2nd edn. Springer, New York (2013)
26. LeCun, Y., Cortes, C., Burges, C.: Mnist handwritten digit database. http://yann.lecun.com/exdb/mnist/ (2010)
27. Lee, J.D., Simchowitz, M., Jordan, M.I., Recht, B.: Gradient descent only converges to minimizers. JMRL **49**, 1–12 (2016)
28. lrfinder. https://github.com/davidtvs/pytorch-lr-finder (2018)
29. Mahrsereci, M., Hennig, P.: Probabilistic line searches for stochastic optimisation. JMLR **18**, 1–59 (2017)

30. Nesterov, Y.: Introductory Lectures on Convex Optimization : A Basic Course. Kluwer Academic Publishers, Dordrecht (2004)
31. Nguyen, T.H.: https://github.com/hank-nguyen/MBT-optimizer (2019)
32. Nielsen, M.A.: Neural Networks and Deep Learning. Determination Press, New York (2015)
33. Nocedal, J., Wright, S.J.: Numerical optimization. Springer series in operations research (1999)
34. Panageas, I., Piliouras, G.: Gradient descent only converges to minimizers: Non-isolated critical points and invariant regions. In: C. H. Papadimitrou, editor, 8th Innovations in theoretical computer science conference (ITCS 2017, pp. 2:1–2:12, Leibniz, Germany, Dagstuhl Publishing (2017)
35. Panageas, I., Piliouras, G., Wang, X.: First-order methods almost always avoid saddle points: The case of vanishing step-sizes. In: 33rd Conference on Neural Information Processing Systems (NeurIPS 2019, pp. 6474–6483, Vancouver, Canada, Dagstuhl Publishing (2019)
36. Poljak, B.T., Tsypkin, Y.Z.: Pseudogradient adaptation and training algorithms. Autom. Remote Control **12**, 83–94 (1973)
37. Robbins, H., Monro, S.: A stochastic approximation method. Ann. Math. Stat. **22**, 400–407 (1951)
38. Ruder, S.: An overview of gradient descent optimisation algorithms. arXiv:1609.04747 (2016)
39. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.: Mobilenetv2: Inverted residuals and linear bottlenecks. arXiv:1704.04861 (2017)
40. Scaman, K., Virmaux, A.: Lipschitz regularity of deep neural networks: analysis and efficient estimation. NeurIPS 2018: 32nd Conference on Neural Information Processing Systems, p. 10 (2018)
41. Smith, L.N.: Cyclical learning rates for training neural networks. arXiv:1506.01186 (2015)
42. Smith, L.N., Topin, N.: Super-convergence: very fast training of residual networks using large learning rates. CoRR. arxiv:1708.07120 (2017)
43. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. JMLR, pp. 1929–1958 (2014)
44. Truong, T.T.: Convergence to minima for the continuous version of backtracking gradient descent. arXiv:1911.04221 (2019)
45. Truong, T.T.: Backtracking gradient descent allowing unbounded learning rates. arXiv:2001.02005 (2020)
46. Truong, T.T.: Some convergent results for backtracking gradient descent method on banach spaces. arXiv:2001.05768 (2020)
47. Truong, T.T., Nguyen, T.H.: Backtracking gradient descent method for general c^1 functions with applications to deep learning. arXiv:1808.05160 (2018, 2019)
48. Truong, T.T., Nguyen, T.H.: Asymptotic behaviour of learning rates in armijo's condition. arXiv:2007.03618 (2020)
49. Vaswani, S., Mishkin, A., Laradji, I., Schmidt, M., Gidel, G., Lacoste-Julien, S.: Painless stochastic gradient: interpolation, line-search and convergence rates. arXiv:1905.09997 (2019)
50. Wilson, A.C., Roelofs, R., Stern, M., Srebro, N., Recht, B.: The marginal value of adaptive gradient methods in machine learning. arXiv:1705.08292 (2017)
51. Wolfe, P.: Convergence conditions for ascent methods. SIAM Rev. **11**(2), 226–235 (1969)
52. Zhang, J., Mitliagkas, I.: Yellowfin and the art of momentum tuning. arXiv:1706.03471