UiO : Department of Mathematics
University of Oslo

Sophus B. Gullbekk

A Comparison of Gradient Descent Methods Using Armijo's Condition

MAT2000 — Project Work in Mathematics

Supervisor: Tuyen Trung Truong

2022

Abstract

We study three different variants of gradient descent, and show how they compare on the Rosenbrock function. The three variants are standard-, backtracking- and two-way backtracking gradient descent. We compare how many iterations they need, as well as how much time they use to converge for different initial learning rates δ_0. The results show that both backtracking- and two-way backtracking gradient descent converge for all δ_0, while standard gradient descent is less stable. In addition, we find that two-way backtracking gradient descent is the most time-efficient method.

1 Introduction

Gradient Descent (GD), is a well known optimization method. All GD models find a local minimum by traveling along the path of the steepest descent. The length of each step in the path towards the minimum is governed by what we refer to as the *learning rate*. A good learning rate ensures that the model converges efficiently. There are many approaches that can help you select a suitable learning rate, the simplest method is to pick a constant $\delta_0 > 0$ and use it every iteration. This is the strategy that is followed in *standard* GD (SGD). To improve SGD Armijo proposed an update rule in [Arm66] that later is referred to as *Armijo's condition*. If we use Armijo's condition to update the learning rate each iteration we get two different GD methods. The first method is *backtracking* GD (BGD), the other is *two-way* BGD. Both methods adjust the learning rate each step, to acheive a more efficient and stable algorithm than SGD.

To test the performance of the three different GD methods, we apply them to the *Rosenbrock function* given by

$$f(x, y) = (1 - x)^2 + b(y - x^2)^2, \quad b \in R, \tag{1}$$

which has a global minimum when $(x, y) = (1, 1)$. We use the Rosenbrock function as it is a common test function for optimization when $b = 100$.

In section 2 the relevant theory behind the three GD methods and Armijo's condition is introduced. In section 3 we present the relevant figures and results. Finally, the conclusions are given in section 4.

2 Theory

2.1 Gradient Descent

We start off with outlining the algorithm for SGD. Let $f : \mathbb{R}^k \to \mathbb{R}$ be a function in C^1. Next, choose an appropriate learning rate $\delta_0 > 0$ and a $x_0 \in \mathbb{R}^k$. We can now construct the sequence $\{x_n\}$ given by

$$x_{n+1} = x_n - \delta_0 \nabla f(x_n), \quad n = 1, 2, 3, \ldots, \tag{2}$$

which we refer to as SGD.

The idea behind equation 2 is that $\{x_n\}$ will converge to a local minimum, by following the path of the steepest descent given by $\nabla f(x_n)$. Hovever, as [Rud16] points out, there are a few challenges that needs to be addressed. The challenges mainly revolves around the choice of a appropriate learning rate δ_0. If the learning rate is too small the convergence rate will be exceedingly slow, and if it is too big $\{x_n\}$ might not converge at all. Moreover, if one works with non-smooth functions like $f(x) = |x|^{4/3}$, then there may be no value of learning rate δ_0 for which SGD converges to the global minimum $x = 0$, see [TN18, p. 11].

2.2 Armijo's Condition

In this section we present Armijo's condition [Arm66, p. 2] and show how it is used to construct BGD and two-way BGD. Let $f \in C^1$ and $a \in (0, 1)$. Armijo's condition tells us that there exist a $\sigma > 0$ such that

$$f(x - \sigma \nabla f(x)) - f(x) \leq -\alpha\sigma \|\nabla f(x)\|^2, \tag{3}$$

where $\|\cdot\|$ is the Euclidian norm.

Now, let $\beta \in (0, 1)$ and choose σ to be the greatest element in $\{\beta^n \delta_0\}$ that satisfies Armijo's condition (3). If we replace the learning rate δ_0 by σ in the SGD algorithm (2) we get the sequence

$$x_{n+1} = x_n - \sigma \nabla f(x_n), \quad n = 1, 2, 3, \ldots, \tag{4}$$

which is defined as BGD. This method lets us decrease the learning rate which leads to a more stable and efficient algorithm. However, it seems obvious that it should be possible to make the learning late larger as well. To do this, we introduce the last algorithm: Two-way BGD. During two-way BGD, we first check if Armijo's condition (3) is met, and if it is not, we calculate the learning rate σ according to normal BGD. However, if the learning rate σ satisfy Armijo's condition (3), we increase σ to be the smallest element in $\{\sigma/\beta^n\}$ that still satisfies the condition. This way, we can avoid both divergence caused by a too large learning rate and slow convergence caused by a too small learning rate.

3 Figures and Results

All figures and results are generated by code from [Gul22].

Figure (1) shows a visualization of the sequence generated by SGD, BGD and two-way BGD, when applied to the Rosenbrock function (1) with $b = 1$. The trivial case of $b = 1$ is chosen because it illustrates nicely how the methods move towards the solution. Table (1) includes two subtables that compare the performance of SGD, BGD and two-way BGD on the Rosenbrock function (1) with the more difficult case $b = 100$. Subtable (1a) show how many iterations the different methods need to converge to the global minimum, while subtable (1b) show how much time each method used.

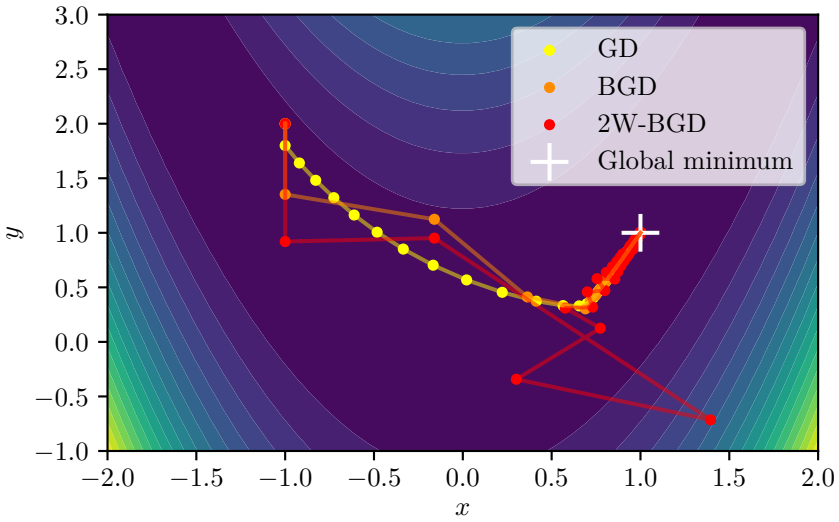

Figure 1: This figure shows how the sequences generated by SGD, BGD and two-way BGD moves towards the minimum from the initial point $(x, y) = (-1, 2)$. The methods are applied to the Rosenbrock function (1) with $b = 100$, which has a global minimum at the point (1, 1). The parameters used for SGD is $\delta_0 = 0.1$, while for BGD and two-way BGD it is $\alpha = 0.5$, $\beta = 0.6$ and $\delta_0 = 0.9$. The margin of error is $\epsilon = 10^{-7}$.

	Initial learning rate δ_0						
	10^2	10^1	10^0	10^{-1}	10^{-2}	10^{-3}	10^{-4}
SGD	∞	∞	∞	∞	∞	21030	152911
BGD	48830	32450	39400	47837	58019	38618	152911
Two-way BGD	9213	6952	10380	9345	7235	8452	9383

(a) The number of iterations used by each method to converge for various values of δ_0.

	Initial learning rate δ_0						
	10^2	10^1	10^0	10^{-1}	10^{-2}	10^{-3}	10^{-4}
SGD	N.A.	N.A	N.A	N.A	N.A.	0.350	2.523
BGD	1.179	0.762	0.935	1.169	1.370	0.908	3.590
Two-way BGD	0.440	0.410	0.477	0.477	0.373	0.378	0.416

(b) The amount of time in seconds taken to converge for various values of δ_0.

Table 1: Two tables that compare the performance of SGD, BGD and two-way BGD for various different initial learning rates δ_0, when applied to the Rosenbrock function (1) with $b = 100$. The backtracking parameters are: $\alpha = 0.5$ and $\beta = 0.5$. The margin of error is $\epsilon = 10^{-7}$.

4 Conclusions

We have described the theory behind SGD and how Armijo's condition is used to construct BGD and two-way BGD. The three different GD methods are compared on the Rosenbrock function (1) for the trivial case $b = 1$ and the more difficult case $b = 100$. the results showed that BGD and two-way BGD converged for all initial learning rates δ_0, but SGD were less stable. SGD diverged for all $\delta_0 \geq 10^{-2}$. Both BGD and SGD converged slowly for $\delta_0 = 10^{-4}$. Two-way BGD had a good performance for all tested δ_0. When comparing the time taken by each algorithm, we see a similar result; Two-way BGD performs better than BGD, and SGD diverges for most δ_0. In conclusion, we can make the heuristic argument that two-way BGD is the more stable and efficient model when applied to the Rosenbrock function.

In further works, the different models will be tested on a large scale with the help of deep neural networks.

References

[Arm66] Armijo, L. 'Minimization of functions having lipschitz continuous first partial derivatives'. eng. In: *Pacific journal of mathematics* vol. 16, no. 1 (1966), pp. 1–3.

[Gul22] Gullbekk, S. *Implementation of Gradient Descent Methods*. Feb. 2022. URL: https://github.com/sophus0505/Implementation-of-Gradient-Descent-Methods.

[Rud16] Ruder, S. 'An overview of gradient descent optimization algorithms'. In: *CoRR* vol. abs/1609.04747 (2016). arXiv: 1609.04747.

[TN18] Truong, T. T. and Nguyen, T. H. *Backtracking gradient descent method for general C^1 functions, with applications to Deep Learning*. 2018. arXiv: 1808.05160.