Gradient descent

Juan Antonio Pérez Juárez
University of Guadalajara
Guadalajara, Jalisco
juan.perez0996@alumnos.udg.mx

I. INTRODUCTION

In the rapidly evolving field of artificial intelligence (AI), optimization algorithms play a crucial role in enhancing the performance of machine learning models. Among these algorithms, gradient descent stands out as one of the most widely used techniques for minimizing loss functions, thereby improving model accuracy and efficiency. This iterative optimization method is fundamental in training various types of models, including neural networks, support vector machines, and linear regression.

Gradient descent operates by adjusting the parameters of a model in the opposite direction of the gradient of the loss function with respect to those parameters. This approach not only ensures convergence towards a local minimum but also facilitates the exploration of high-dimensional parameter spaces, which is essential in complex AI applications. The simplicity and effectiveness of gradient descent have led to its adoption across numerous domains, from computer vision to natural language processing.

However, while gradient descent is powerful, it is not without its challenges. Issues such as local minimal, saddle points, and slow convergence can hinder the optimization process. To address these challenges, various enhancements and alternatives to the standard gradient descent algorithm have been proposed, including stochastic gradient descent (SGD), mini-batch gradient descent, and adaptive learning rate methods like Adam and RMSprop.

This article aims to provide a comprehensive overview of gradient descent, exploring its theoretical foundations, practical implementations, and recent advancements. By understanding the nuances of this optimization technique, researchers and practitioners can better harness its potential to drive innovation in AI.

II. ORIGINS

Gradient descent is generally attributed to Augustin-Louis Cauchy, who first suggested it in 1847. Jacques Hadamard independently proposed a similar method in 1907. Its convergence properties for non-linear optimization problems were first studied by Haskell Curry in 1944, with the method becoming increasingly well-studied and used in the following decades.

III. DESCRIPTION

Gradient descent is based on the observation that if the multi-variable function $f(x)$ is defined and differentiable in a neighborhood of a point a, then $f(x)$ decreases fastest if one goes from a in the direction of the negative gradient of f at a, $-\nabla f(a)$, it follows that, if $a_{n+1} = a_n - n\nabla f(a_n)$.

For a small enough step size or learning rate $n \in R_+$, then $f(a_n) \geq f(a_{n+1})$ to words. The term $n\nabla f(a)$ is subtracted from a because we want to move against the gradient, toward the local minimum. With this observation in mind, one starts with a guess x_0 for a local minimum of f, and considers the sequence $x_0, x_1, \ldots$ such that

$$x_{n+1} = x_n \nabla f(x_n), n \geq 0$$

We have a monotonic sequence

$$f(x_0) \geq f(x_1) \geq f(x_2) \geq \cdots$$

So, the sequence (x_n) converges to the desired local minimum. Note that the value if the step n is allowed to change in every iteration.

It is possible to guarantee the convergence to a local minimum under certain assumptions on the function f and choices for n, those includes the sequence

$$\eta_n = \frac{\left|(\mathbf{x}_n - \mathbf{x}_{n-1})^\top [\nabla f(\mathbf{x}_n) - \nabla f(\mathbf{x}_{n-1})]\right|}{\|\nabla f(\mathbf{x}_n) - \nabla f(\mathbf{x}_{n-1})\|^2}$$

as in the Barzilai-Borwein method, or a sequence n_m satisfying the Wolfe conditions (which can be found by using line search). When the function f is convex, all local minimum are also global minimal, so in this case gradient descent can converge to the global solution.

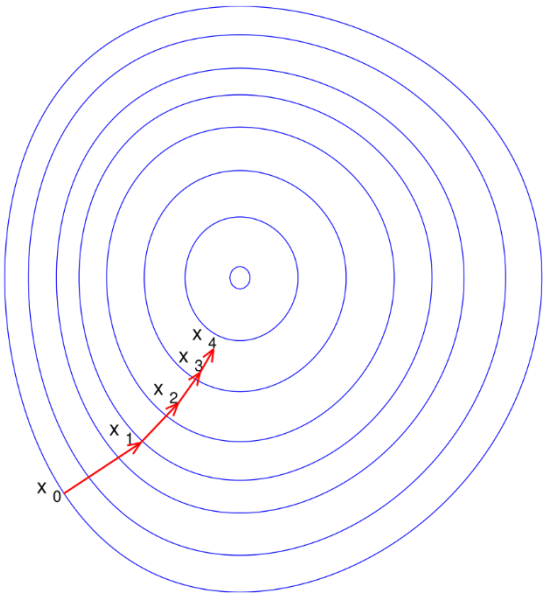

Fig. 1 f is assumed to be defined on the plane, and its graph has a bowl shape. The blue curves are the contour lines, that is, the regions on which the value of f is constant. A red arrow originating at a point shows the direction of the negative gradient at that point. Note that the (negative) gradient at a point is orthogonal to the contour line going through that

point. We see that gradient descent leads us to the bottom of the bowl, that is, to the point where the value of the function f is minimal.

IV. ANALOGY

Imagine you are hiking down a foggy mountain. Your goal is to reach the lowest point in the valley, but the thick fog prevents you from seeing the entire landscape. Instead, you can only see a small area around you.

Finding the Slope: As you stand at your current position on the mountain, you can feel the slope of the ground beneath your feet. You lean forward and notice which direction feels steepest downhill. This direction indicates where you should move next.

You take a step in that downhill direction. After stepping, you stop and reassess your surroundings. Again, you feel the slope and determine the steepest direction from your new position.

You continue this process, feeling the slope and taking steps downhill, until you can no longer find a lower point. Eventually, you reach a local minimum, which may not be the absolute lowest point in the valley, but it is the lowest point you can see from your current position.

The size of your steps represents the learning rate in gradient descent. If you take large steps, you might overshoot the valley and end up climbing back up the mountain. If you take very small steps, it may take a long time to reach the valley floor.

Sometimes, you may find yourself in a situation where you reach a low point that isn't the lowest point in the entire valley (a local minimum). To find the absolute lowest point (global minimum), you might need to explore different paths or start from different locations on the mountain.

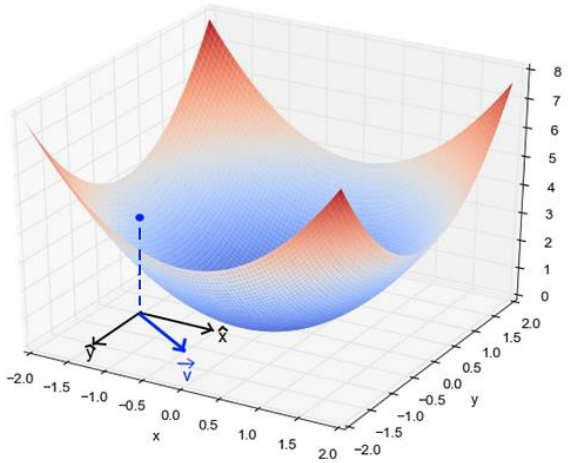

Fig 2. This 3D plot visualizes the gradient descent optimization process. The blue dot represents the initial parameter values, while the valleys and lower regions depict the areas of lower loss that the algorithm aims to converge towards. The color gradient from blue to red indicates the magnitude of the gradients, with the red regions showing steeper gradients driving the optimization downhill. This plot effectively illustrates the iterative nature of gradient descent as it navigates the complex landscape to find the optimal solution.

V. TYPES

A. Batch Gradient Descent

Batch gradient descent sums the error for each point in a training set, updating the model only after all training examples have been evaluated. This process referred to as a training epoch.

While this batching provides computation efficiency, it can still have a long processing time for large training datasets as it still needs to store all the data into memory. Batch gradient descent also usually produces a stable error gradient and convergence, but sometimes that convergence point isn't the most ideal, finding the local minimum versus the global one.

B. Stochastic Gradient Descent

Stochastic gradient descent (SGD) runs a training epoch for each example within the dataset, and it updates each training example's parameters one at a time. Since you only need to hold one training example, they are easier to store in memory. While these frequent updates can offer more detail and speed, it can result in losses in computational efficiency when compared to batch gradient descent. Its frequent updates can result in noisy gradients, but this can also be helpful in escaping the local minimum and finding the global one.

C. Mini-batch gradient descent

Mini-batch gradient descent combines concepts from both batch gradient descent and stochastic gradient descent. It splits the training dataset into small batch sizes and performs updates on each of those batches. This approach strikes a balance between the computational efficiency of batch gradient descent and the speed of stochastic gradient descent.

VI. OPERATION

1. The algorithm optimizes to minimize the model's cost function.
2. The cost function measures how well the model fits the training data and defines the difference between the predicted and actual values.
3. The cost function's gradient is the derivative with respect to the model's parameters and points in the direction of the steepest ascent.
4. The algorithm starts with an initial set of parameters and updates them in small steps to minimize the cost function.
5. In each iteration of the algorithm, it computes the gradient of the cost function with respect to each parameter.
6. The gradient tells us the direction of the steepest ascent, and by moving in the opposite direction, we can find the direction of the steepest descent.
7. The learning rate controls the step size, which determines how quickly the algorithm moves towards the minimum.
8. The process is repeated until the cost function converges to a minimum. Therefore, indicating that the model has reached the optimal set of parameters.
9. Different variations of gradient descent include batch gradient descent, stochastic gradient descent, and mini-

batch gradient descent, each with advantages and limitations.
10. Efficient implementation of gradient descent is essential for performing well in machine learning tasks. The choice of the learning rate and the number of iterations can significantly impact the algorithm's performance.

VII. PLOTTING

When we have a single parameter (theta), we can plot the dependent variable cost on the y-axis and theta on the x-axis. If there are two parameters, we can go with a 3-D plot, with cost on one axis and the two parameters (thetas) along the other two axes.
It can also be visualized by using Contours. This shows a 3-D plot in two dimensions with parameters along axes and the response as a contour. The value of the response increases away from the center and has the same value as with the rings. The response is directly proportional to the distance of a point from the center (along with direction).

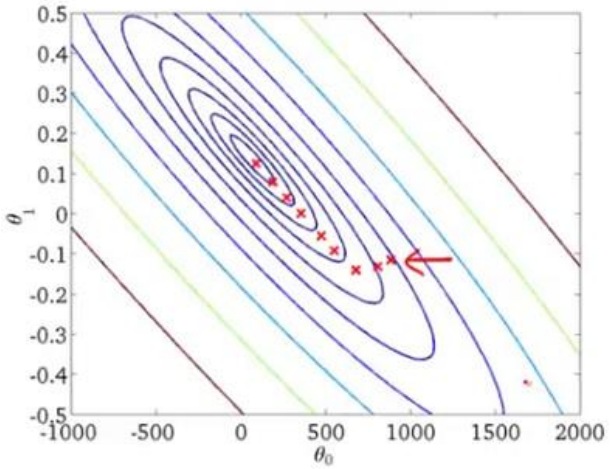

Fig 3. The contours represent level sets of the function being optimized. The red arrows indicate gradient direction, guiding the optimization towards the minimum point (red dot). This 2D visualization can help understand gradient-based optimization, but real-world problems often have higher dimensions, making visualization more challenging.

VIII. ADVANTAGES

A. Simplicity and Efficiency: Gradient descent is a relatively simple algorithm that is easy to implement and computationally efficient, especially for problems with a large number of parameters. The algorithm iteratively updates the parameters in the direction of the negative gradient, which can lead to rapid convergence.

B. Scalability: Gradient descent can be easily scaled to handle optimization problems with high-dimensional parameter spaces. The algorithm's computational complexity scales linearly with the number of parameters, making it suitable for large-scale optimization tasks.

C. Convergence Guarantees: Under certain conditions, such as convexity of the objective function, gradient descent is guaranteed to converge to the global minimum. This property makes gradient descent a reliable choice for a wide range of optimization problems.

D. Adaptability: Gradient descent can be adapted and extended to handle various types of optimization problems, including constrained optimization, stochastic optimization, and online optimization. These extensions enhance the algorithm's versatility and applicability.

E. Interpretability: The step-by-step updates of gradient descent provide insight into the optimization process, making it easier to understand and debug the algorithm's behavior. This interpretability can be valuable in many applications.

IX. CONCLUSIONS

In summary, gradient descent is a foundational algorithm in the field of artificial intelligence, serving as a critical tool for optimizing machine learning models. Its ability to efficiently minimize loss functions has made it indispensable for training a wide array of algorithms, from simple linear regression to complex deep learning architectures. As we explored, the iterative nature of gradient descent allows for effective navigation through high-dimensional parameter spaces, making it possible to achieve improved model performance.

Despite its widespread use, gradient descent is not without challenges, including issues related to convergence and the potential for getting trapped in local minima. However, ongoing research and advancements, such as adaptive learning rates and variations like stochastic and mini-batch gradient descent, continue to enhance its robustness and applicability across various domains.

As artificial intelligence continues to evolve, the significance of gradient descent remains paramount. Its role in enabling models to learn from data and make accurate predictions underscores the necessity of mastering this optimization technique. By understanding and leveraging gradient descent, researchers and practitioners can drive innovation and achieve breakthroughs in AI, ultimately paving the way for more intelligent systems that can tackle increasingly complex problems.

REFERENCES

[1] IBM, «Gradient Descent», What is gradient descent?, 17 de November de 2025. https://www.ibm.com/think/topics/gradient-descent
[2] Wikipedia contributors, "Gradient descent," *Wikipedia*, Nov. 02, 2025. https://en.wikipedia.org/wiki/Gradient_descent
[3] «Gradient Descent Explained: The Engine Behind AI Training», MEDIUM, January 2025. https://medium.com/@abhaysingh71711/gradient-descent-explained-the-engine-behind-ai-training-2d8ef6ecad6f (accessed November 202