A Study of Gradient Descent and Stochastic Gradient Descent Methods in Optimization Problems.

Flores Vásquez, Abraham Alejandro
Universidad Centroamericana ”Jose Simeón Cañas”
00067323@uca.edu.sv

Morales Pineda, Alexander Efraín
Universidad Centroamericana ”Jose Simeón Cañas”
00024123@uca.edu.sv

Iraheta Monterrosa, Diego Alejandro
Universidad Centroamericana ”Jose Simeón Cañas”
00041923@uca.edu.sv

Tovar Jovel, Cesar Isaac
Universidad Centroamericana ”Jose Simeón Cañas”
00016023@uca.edu.sv

Abstract

***Abstract*—This trial aims to analyze and compare two fundamental techniques in supervised learning: multivariate linear regression and multiclass logistic regression. For the linear regression case, the Gradient Descent method is implemented to fit a predictive model on continuous variables. In contrast, for multiclass logistic regression, the Stochastic Gradient Descent (SGD) method is employed because of its computational efficiency when handling large volumes of data. Both implementations include preprocessing, training, evaluation, and result visualization processes, all documented using appropriate computational tools. The study seeks not only to understand the numerical behavior of both algorithms but also to assess their performance on various real-world datasets.**

Key words

***Index Terms*—Multivariate linear regression, multiclass logistic regression, Gradient Descent, Stochastic Gradient Descent, machine learning, numerical optimization.**

I. Introducción

In the context of supervised machine learning, regression and classification methods are fundamental tools for prediction and data-driven decision-making. Multivariate linear regression enables modeling of relationships between multiple independent variables and a continuous dependent variable, and is widely used in quantitative prediction problems [4]. To fit the model parameters, the Gradient Descent method is implemented, which iteratively optimizes a cost function through steps proportional to the negative gradient.

On the other hand, when the goal is to assign inputs to discrete categories, classification techniques such as logistic regression are employed. Specifically, for problems involving multiple classes, multiclass logistic regression emerges as a natural extension of the binary model [5]. In this study, this model is implemented using the Stochastic Gradient Descent (SGD) method, which offers significant computational advantages by updating model parameters based on individual samples or small subsets of data.

Both methods share common numerical foundations but differ in their objectives, cost functions, and optimization strategies. The purpose of this study is to explore these methodologies both theoretically and computationally, analyzing their behavior, convergence, advantages, and limitations, as well as their applicability in various real-world scenarios.

II. Mathematical Preliminaries

A. Taylor Expansion for Multivariable Functions

The Taylor expansion allows the approximation of a differentiable function in the neighborhood of a given point [6]. For a function $L : \mathbb{R}^n \to \mathbb{R}$, the first-order expansion around a point θ is expressed as:

$$L(\theta + \Delta\theta) \approx L(\theta) + \nabla L(\theta)^T \Delta\theta$$

where:

- $\nabla L(\theta)$ is the gradient vector, which contains the partial derivatives of L with respect to each component of θ.
- $\Delta\theta$ is the vector representing a small change in the parameters.
- The transpose ^{T} is used to perform the dot product between the vectors.

This approximation is fundamental in optimization algorithms, as it describes how the function changes around a point and helps determine the direction of the next step.

B. Gradient: Definition and Interpretation

The gradient of a function $L : \mathbb{R}^n \to \mathbb{R}$ is defined as:

$$\nabla L(\theta) = \left(\frac{\partial L}{\partial \theta_1}, \frac{\partial L}{\partial \theta_2}, \ldots, \frac{\partial L}{\partial \theta_n} \right)^T$$

Each component corresponds to the partial derivative with respect to one of the parameters. Geometrically, the gradient points in the direction of the greatest increase of the function at a given point. This property is essential for descent-based methods, as minimizing the function involves moving in the opposite direction of the gradient [7].

C. Optimality Concepts: Minima and Maxima

In optimization, a point θ^* is called a *critical point* if it satisfies:

$$\nabla L(\theta^*) = 0$$

This point can be:

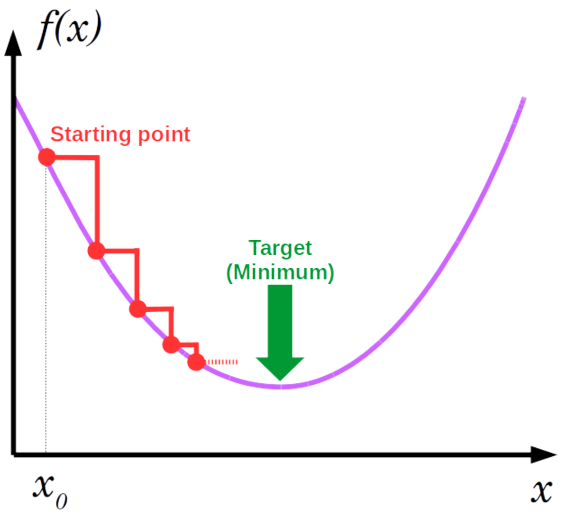
- A local minimum, if the function takes greater or equal values in a neighborhood around θ^*.
- A local maximum, if the function takes smaller or equal values in a neighborhood around θ^*.
- A saddle point, if it is neither a maximum nor a minimum.

In the case of convex functions, any local minimum is also a global minimum, which significantly simplifies optimization problems by eliminating the possibility of multiple local minimums.

III. Gradient Descent Method

Gradient Descent is a fundamental method, as it serves as the foundation for virtually all training algorithms in Machine Learning models, including Neural Networks, Convolutional Networks, Recurrent Networks, and Transformer architectures.

This optimization algorithm enables the automatic discovery of a function's minimum by leveraging the gradient (or derivative) of that function, which guides the algorithm to progressively approach the optimal minimum [9].



A. Method Derivation

Let $L(\theta) : \mathbb{R}^n \to \mathbb{R}$ be a differentiable cost function we aim to minimize, where $\theta \in \mathbb{R}^n$ represents the parameter vector of the model. The function $L(\theta)$ measures the error or the discrepancy between the model's predictions and the actual values.

Gradient Descent relies on a first-order approximation using the Taylor expansion for multivariable functions:

$$L(\theta + \Delta\theta) \approx L(\theta) + \nabla L(\theta)^T \Delta\theta$$

Here, $\nabla L(\theta)$ is the gradient vector of the cost function, containing the partial derivatives with respect to the parameters θ, and $\Delta\theta$ is the change applied to those parameters. The symbol T denotes the **transpose** of the gradient vector. Since both $\nabla L(\theta)$ and $\Delta\theta$ are column vectors, the transpose converts the gradient into a row vector, allowing the dot product to be computed:

$$\nabla L(\theta)^T \Delta\theta \in \mathbb{R}$$

To minimize the function, $\Delta\theta$ is chosen as the direction opposite to the gradient, since the gradient points in the direction of greatest increase of the function. Thus, we define:

$$\Delta\theta = -\alpha \nabla L(\theta)$$

where $\alpha > 0$ is the **learning rate**, which determines the step size.

Substituting this into the update rule yields:

$$\theta^{(k+1)} = \theta^{(k)} - \alpha \nabla L(\theta^{(k)})$$

This process is repeated iteratively until the gradient becomes sufficiently close to zero or a maximum number of iterations is reached, with the aim of approaching a local or global minimum of the cost function.

B. Convergence Rate Proof

Let us assume that the cost function $L : \mathbb{R}^n \to \mathbb{R}$ is convex and that its gradient is L_g-Lipschitz continuous. This condition implies that the variations in the gradient are bounded proportionally to the variations in the parameters, i.e.,

$$\|\nabla L(\theta_1) - \nabla L(\theta_2)\| \leq L_g \|\theta_1 - \theta_2\| \quad \forall \theta_1, \theta_2 \in \mathbb{R}^n.$$

This condition is required because it ensures that the function does not vary too abruptly and that the steps of the Gradient Descent method are controlled and stable. Without this property, it is not possible to guarantee a theoretical convergence rate, and the algorithm might diverge or behave unpredictably. In practical terms, Lipschitz continuity of the gradient is associated with the objective function having well-behaved curvature throughout the optimization domain.

Under this condition, it can be shown that Gradient Descent with a learning rate $0 < \alpha < \frac{2}{L_g}$ satisfies the following inequality at each iteration:

$$L(\theta^{(k+1)}) - L(\theta^*) \leq (1 - \alpha\mu)[L(\theta^{(k)}) - L(\theta^*)]$$

where:

- θ^* is the global optimum,
- μ is the strong convexity constant (if applicable).

In the general case (without strong convexity), the convergence is sublinear, and the following rate is obtained:

$$L(\theta^{(k)}) - L(\theta^*) = \mathcal{O}\left(\frac{1}{k}\right).$$

However, when L is convex and strongly convex, the convergence is linear:

$$\|\theta^{(k)} - \theta^*\| \leq C\rho^k$$

with $0 < \rho < 1$ and C a positive constant dependent on the initial condition.

This means that the distance to the optimum decreases by approximately a constant factor in each iteration, which characterizes **linear convergence**. For this reason, Gradient Descent is considered efficient, though it has a lower convergence order compared to methods such as Newton-Raphson, which achieve quadratic convergence near the optimum.

C. Stability Conditions

The stability of the Gradient Descent method is primarily determined by the choice of the learning rate α. For the algorithm to be stable and converge to an optimum, it is necessary for the cost function $L(\theta)$ to be convex and for its gradient to be L_g-Lipschitz continuous, as discussed earlier.

The classical stability condition is given by:

$$0 < \alpha < \frac{2}{L_g}$$

This range ensures that each update reduces the cost function. If α is too large, the algorithm may oscillate around the minimum or even diverge; whereas if α is too small, convergence may be significantly slowed down.

In practice, choosing an appropriate α often requires experimentation or the use of adaptive techniques, such as progressively decreasing the learning rate or using more advanced algorithms that dynamically adjust this value.

D. Advantages and Disadvantages

Advantages:

- **Simplicity:** The method is easy to understand and implement.
- **Scalability:** It works well with large datasets, especially when combined with stochastic variants (SGD).
- **Versatility:** It can be applied to a wide variety of convex and non-convex optimization problems.
- **Low computational demand:** It only requires the computation of the gradient, making it less expensive than second-order methods.

Disadvantages:

- **Sensitivity to the learning rate:** The choice of α is critical and can significantly affect convergence.
- **Slow convergence:** The convergence rate is linear, slower than that of second-order methods such as Newton-Raphson.
- **Local minima:** In non-convex problems, it may get stuck in local optima or saddle points.

E. Example of Use

Let us consider the quadratic function:

$$L(\theta) = (\theta - 3)^2$$

This function has a unique global minimum at $\theta = 3$. The gradient is:

$$\nabla L(\theta) = 2(\theta - 3)$$

Applying the Gradient Descent method with a learning rate $\alpha = 0.1$ and an initial value $\theta^{(0)} = 0$, we obtain the following iterations:

$$\theta^{(1)} = 0 - 0.1 \times (-6) = 0.6,$$

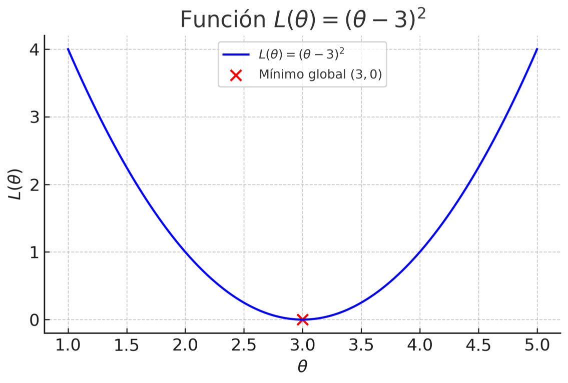
$$\theta^{(2)} = 0.6 - 0.1 \times (-4.8) = 1.08,$$

$$\theta^{(3)} = 1.08 - 0.1 \times (-3.84) = 1.464,$$

$$\theta^{(4)} = 1.464 - 0.1 \times (-3.072) = 1.771,$$

$$\theta^{(5)} = 1.771 - 0.1 \times (-2.458) = 2.016.$$

We observe how θ progressively approaches the optimal value $\theta = 3$ as the iterations progress. This simple example illustrates how the method adjusts the parameters in the direction that minimizes the cost function.

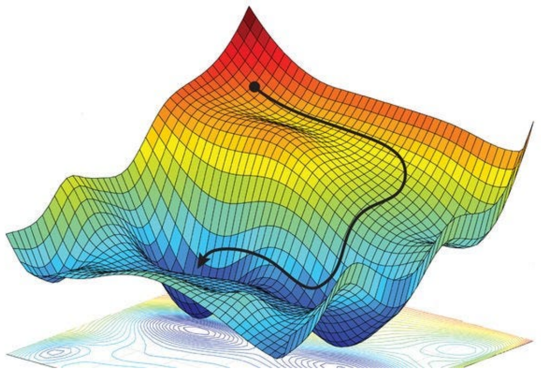


IV. Stochastic Gradient Descent (SGD)

The Stochastic Gradient Descent (SGD) method is a variant of the classic Gradient Descent algorithm designed to improve computational efficiency, especially when working with large datasets. Unlike the traditional method, which computes the gradient using the entire set of samples in each iteration, SGD updates the parameters using only one sample (or a small mini-batch) at a time. [10]

This feature makes SGD significantly faster per iteration, although it introduces noise into the convergence path, since each step may fluctuate due to the inherent variability of individual data points.

In practice, SGD is widely used to train classification models such as multiclass logistic regression, neural networks, and other machine learning algorithms.



A. *Derivation of the Method*

Let $\{(x_i, y_i)\}_{i=1}^m$ be a training set, where:

- $x_i \in \mathbb{R}^n$ is the input (feature vector),
- y_i is the corresponding class label,
- m is the total number of samples.

In the classical method, the gradient of the total cost function is computed as:

$$\nabla L(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla L_i(\theta)$$

where $L_i(\theta)$ is the loss associated with sample i.

In the stochastic method, instead of computing the full sum, a random sample (x_j, y_j) is selected, and the parameters are updated using only its gradient:

$$\theta^{(k+1)} = \theta^{(k)} - \alpha \nabla L_j(\theta^{(k)})$$

where:

- α is the learning rate,
- $j \in \{1, 2, \ldots, m\}$ is the random index of the selected sample in iteration k.

This scheme introduces noise at each step, but on average it still approximates the true gradient direction, allowing convergence to the expected minimum.

B. *Convergence Rate Proof*

The convergence analysis of Stochastic Gradient Descent (SGD) differs from the classical method due to the noisy nature of the updates. Under certain conditions, it can be shown that SGD converges on average toward an optimum.

Assume that the cost function is convex and that the stochastic gradient is an unbiased estimator of the true gradient:

$$\mathbb{E}[\nabla L_j(\theta)] = \nabla L(\theta)$$

Additionally, we require that the variance of the gradient is bounded:

$$\mathbb{E}\|\nabla L_j(\theta) - \nabla L(\theta)\|^2 \leq \sigma^2$$

Under these conditions, and using a decreasing learning rate $\alpha_k = \frac{1}{k}$, it can be shown that:

$$\mathbb{E}[L(\theta^{(k)}) - L(\theta^*)] = \mathcal{O}\left(\frac{1}{k}\right)$$

This implies that the expected convergence of SGD is **sublinear**, meaning that accuracy improves slowly as the number of iterations increases, but it remains effective for large datasets due to the speed of each individual step.

C. *Stability Conditions*

For SGD to be stable and convergent, certain conditions related to the learning rate must be met. Unlike the classical method, where a constant learning rate can be used under specific conditions, in SGD it is advisable for the rate to decrease with the number of iterations to mitigate stochastic noise.

A typical condition is that the sequence $\{\alpha_k\}$ of learning rates satisfies:

- $\sum_{k=1}^{\infty} \alpha_k = \infty$ (infinite sum),
- $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$ (finite sum of squares).

For example, a common choice is $\alpha_k = \frac{1}{k}$. These conditions ensure that the method converges almost surely to the optimum under standard assumptions of convexity and bounded gradients.

In practice, it is also common to use a constant learning rate for a certain number of iterations and then progressively reduce it to stabilize convergence.

D. *Advantages and Disadvantages*

Advantages:

- **Computational efficiency:** Each update requires only one sample or a small mini-batch, significantly reducing the computational cost per iteration.
- **Scalability:** It is ideal for very large datasets where computing the full gradient is infeasible.

- **Escaping local minima:** Due to its stochastic nature, it can escape shallow local minima in non-convex problems.
- **Simplicity of implementation:** Its structure is simple and easy to implement in practice.

Disadvantages:

- **Noisy convergence:** The randomness introduces oscillations, making smooth convergence to the optimum more difficult.
- **Learning rate sensitivity:** Choosing the correct learning rate schedule is crucial and may require experimental tuning.
- **Slower convergence:** Although each iteration is fast, many more iterations may be required to reach a desired accuracy compared to the classical method.

E. Practical Example

Let us consider a binary classification problem with the logistic cost function for a single sample (x_j, y_j):

$$L_j(\theta) = -y_j \log(h_\theta(x_j)) - (1 - y_j)\log(1 - h_\theta(x_j))$$

where:

- $h_\theta(x_j) = \frac{1}{1+e^{-\theta^T x_j}}$ is the sigmoid function,
- $y_j \in \{0, 1\}$ is the class label,
- $x_j \in \mathbb{R}^n$ is the feature vector.

The gradient for a single sample is:

$$\nabla L_j(\theta) = (h_\theta(x_j) - y_j)x_j$$

Using SGD, the update for a randomly selected sample would be:

$$\theta^{(k+1)} = \theta^{(k)} - \alpha(h_\theta(x_j) - y_j)x_j$$

Numerical Example:

Suppose:

- $x_j = (1, 2)$,
- $y_j = 1$,
- $\theta^{(0)} = (0, 0)$,
- $\alpha = 0.1$.

Calculation:

1) $h_\theta(x_j) = \frac{1}{1+e^0} = 0.5$,
2) Gradient: $(0.5 - 1)(1, 2) = (-0.5, -1.0)$,
3) Update: $\theta^{(1)} = (0, 0) - 0.1 \times (-0.5, -1.0) = (0.05, 0.1)$.

This example illustrates how parameters are incrementally updated toward the optimum using just a single data sample.

V. Comparison between Gradient Descent and Stochastic Gradient Descent

The following is a comparison between the Gradient Descent (GD) method and its stochastic variant (SGD), highlighting their main characteristics and key differences:

- **Gradient computation:**
  - **GD:** Computes the full gradient using all samples in the dataset at each iteration.
  - **SGD:** Computes the gradient using only one (or a mini-batch of) randomly selected sample(s) per iteration.
- **Computational cost:**
  - **GD:** High per-iteration cost for large datasets, since it processes all data at every step.
  - **SGD:** Low per-iteration cost, ideal for very large datasets.
- **Convergence:**
  - **GD:** Stable and smooth convergence, generally faster in terms of total number of iterations.
  - **SGD:** Noisier convergence due to sample variability, but efficient in terms of computation time.
- **Stability:**
  - **GD:** Allows the use of a constant learning rate under certain conditions.
  - **SGD:** Requires a decreasing learning rate or adaptive techniques to ensure stability.
- **Applications:**
  - **GD:** Suitable for problems with small to moderate datasets where per-iteration cost is not a limitation.
  - **SGD:** The preferred option for machine learning problems with large-scale data, such as deep learning model training.

In summary, although both methods share the same optimization philosophy based on gradients, their differences in implementation and behavior make them more or less suitable depending on the context and computational needs.

VI. Conclusions

This work has studied and compared two fundamental methods for numerical optimization in machine learning: Gradient Descent and Stochastic Gradient Descent (SGD). Through theoretical analysis and practical examples, it has been shown how both methods use gradient information to minimize cost functions, but with different approaches regarding the amount of data processed per iteration.

In particular, Gradient Descent was implemented to solve a Multivariate Linear Regression problem, where the objective function is smooth and convex, ensuring stable convergence to the optimal solution. On the other hand, the SGD method was used for Multiclass Logistic Regression, a more complex and common scenario in supervised classification, where SGD demonstrates its computational efficiency by handling large data volumes and performing fast updates.

The choice between one method or the other depends on the specific characteristics of the problem: Gradient Descent is more suitable for regression tasks on moderate datasets, while SGD is the better choice for multiclass classification problems in big data environments or when computational efficiency is critical.

Both methods remain fundamental pillars in modern optimization and continue to evolve through more sophisticated variants that address their limitations and improve performance.

Acknowledgments

The authors would like to express their gratitude to the Universidad Centroamericana ”José Simeón Cañas” and the Department of Mathematics for the support and training received during the completion of this academic project. The valuable guidance of the teaching staff and the resources provided for the development of this work are also gratefully acknowledged.

References

[1] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[3] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed., Springer, 2006.

[4] IBM, “Regresión lineal múltiple,” *IBM Documentation*, [Online]. Available in: https://www.ibm.com/docs/es/cognos-analytics/11.1.x?topic=tests-multiple-linear-regression

[5] Amazon Web Services, “¿Qué es la regresión logística?,” *AWS*, [Online]. Available in: https://aws.amazon.com/es/what-is/logistic-regression/

[6] OpenStax, “6.3 Series de Taylor y de Maclaurin,” *Cálculo, Volumen 2*, [Online]. Available in: https://openstax.org/books/clculo-volumen-2/pages/6-3-series-de-taylor-y-maclaurin

[7] Khan Academy, “El gradiente,” *Khan Academy*, [Online]. Available in: https://es.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/partial-derivative-and-gradient-articles/a/the-gradient

[8] UNAM, “Series de Taylor,” *PAPIME 2020, Geofísica UNAM*, [Online]. Available in: http://gmc.geofisica.unam.mx/papime2020/index.php/articulos/10-series-de-taylor

[9] Khan Academy, “¿Qué es el descenso del gradiente?,” *Khan Academy*, [Online]. Available in: https://es.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/optimizing-multivariable-functions/a/what-is-gradient-descent

[10] Scikit-learn, “Descent gradient estocástico (SGD),” *Scikit-learn (traducción)*, [Online]. Available in: https://scikit--learn-org.translate.goog/stable/modules/sgd.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

[11] Toolify, “Descenso del Gradiente Estocástico vs Descenso del Gradiente: ¿Cuál es el mejor?,” *Toolify.ai*, 4 de marzo de 2024. [Online]. Available in: https://www.toolify.ai/es/ai-news-es/descenso-del-gradiente-estocstico-vs-descenso-del-gradiente/-cual-es-el-mejor-2361431

[12] J. Stewart, *Cálculo de una variable: Trascendentes tempranas*, 8ª ed., México: Cengage Learning, 2016.

[13] G. Strang, *Introduction to Linear Algebra*, 5th ed., Wellesley-Cambridge Press, 2016.

[14] I. Goodfellow, Y. Bengio, y A. Courville, *Deep Learning*, MIT Press, 2016. [Online]. Available in: https://www.deeplearningbook.org/

[15] S. Boyd y L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004. [Online]. Available in: https://web.stanford.edu/~boyd/cvxbook/

[16] T. M. Apostol, *Calculus, Volume II: Multi-Variable Calculus and Linear Algebra with Applications*, 2nd ed., Wiley, 1969.

[17] CRAI UCA El Salvador, *Centro de Recursos para el Aprendizaje y la Investigación*, [Online]. Available in: https://crai.uca.edu.sv