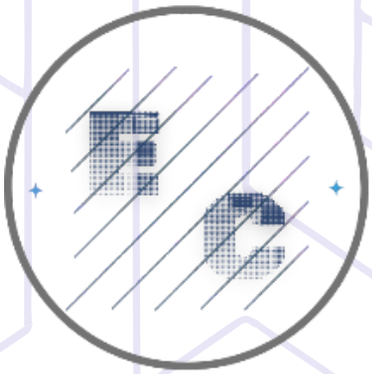
UNIVERSIDAD DE SONSONATE
USO
SCIENTIAE ET BONIS ARTIBUS



REVISTA MULTIDISCIPLINAR EPISTEMOLOGÍA DE LAS CIENCIAS

Universidad de Sonsonate

Volumen 3, Número 3

Julio - Septiembre, 2026
Edición Especial

CROSSREF PREFIX DOI: 10.71112

ISSN: 3061-7812, www.omniscens.com

Revista Multidisciplinar Epistemología de las Ciencias

Volumen 3, Número 3
Julio-Septiembre 2026
Edición Especial

Publicación trimestral
Hecho en México

La Revista Multidisciplinar Epistemología de las Ciencias acepta publicaciones de cualquier área del conocimiento, promoviendo una plataforma inclusiva para la discusión y análisis de los fundamentos epistemológicos en diversas disciplinas. La revista invita a investigadores y profesionales de campos como las ciencias naturales, sociales, humanísticas, tecnológicas y de la salud, entre otros, a contribuir con artículos originales, revisiones, estudios de caso y ensayos teóricos. Con su enfoque multidisciplinario, busca fomentar el diálogo y la reflexión sobre las metodologías, teorías y prácticas que sustentan el avance del conocimiento científico en todas las áreas.

Contacto principal: admin@omniscens.com

9773061781003

Cintillo legal

Revista Multidisciplinar Epistemología de las Ciencias Vol. 3, Núm. 3, julio-septiembre 2026, edición especial, es una publicación trimestral editada por el Dr. Moises Ake Uc, C. 51 #221 x 16B , Las Brisas, Mérida, Yucatán, México, C.P. 97144 , Tel. 9993556027, Web: https://www.omniscens.com, admin@omniscens.com, Editor responsable: Dr. Moises Ake Uc. Reserva de Derechos al Uso Exclusivo No. 04-2024-121717181700-102, ISSN: 3061-7812, ambos otorgados por el Instituto Nacional del Derecho de Autor (INDAUTOR). Responsable de la última actualización de este número, Dr. Moises Ake Uc, fecha de última modificación, 1 julio 2026.

Revista Multidisciplinar Epistemología de las Ciencias

Volumen 3, Número 3, 2026, julio-septiembre, Edición Especial

DOI: https://doi.org/10.71112/fdzkm898

LEVENBERG-MARQUARDT: UN ENFOQUE HÍBRIDO PARA LA MEJORA DEL RENDIMIENTO EN REDES NEURONALES

LEVENBERG-MARQUARDT: A HYBRID APPROACH FOR IMPROVING PERFORMANCE IN NEURAL NETWORKS

Ernesto Antonio Morales Rodríguez

El Salvador

DOI: https://doi.org/10.71112/fdzkm898

Levenberg-Marquardt: Un enfoque híbrido para la mejora del rendimiento en redes neuronales

Levenberg-Marquardt: A hybrid approach for improving performance in neural networks

Ernesto Antonio Morales Rodríguez[a,*]

ernesto.morales@usonsonate.edu.sv

https://orcid.org/0009-0004-1934-7204

*Autor de correspondencia: ernesto.morales@usonsonate.edu.sv, [a]Universidad de Sonsonate, El Salvador

RESUMEN

El presente trabajo analiza la aplicación del algoritmo de Levenberg–Marquardt (LM) como método de optimización para el entrenamiento de redes neuronales multicapa. Este enfoque combina la rapidez del método de Gauss–Newton con la estabilidad del gradiente descendente, ajustando dinámicamente un parámetro de amortiguamiento que regula su comportamiento. Se desarrolló un marco teórico que explica su formulación matemática a partir de la expansión de Taylor, el cálculo del gradiente y el Jacobiano, y su adaptación al aprendizaje supervisado. Los resultados experimentales demuestran que LM mejora significativamente la velocidad de convergencia y la precisión del modelo respecto a métodos de primer orden, logrando un error cuadrático medio notablemente menor. Esto confirma su efectividad para optimizar funciones altamente no lineales y complejas en redes neuronales.

Palabras clave: Levenberg–Marquardt; Redes Neuronales; Optimización; Convergencia; Aprendizaje Supervisado.

DOI: https://doi.org/10.71112/fdzkm898

ABSTRACT

This work analyzes the application of the Levenberg–Marquardt (LM) algorithm as an optimization method for training multilayer neural networks. This approach combines the speed of the Gauss–Newton method with the stability of gradient descent, dynamically adjusting a damping parameter that regulates its behavior. A theoretical framework was developed to explain its mathematical formulation based on the Taylor expansion, the computation of the gradient and the Jacobian, and its adaptation to supervised learning. Experimental results show that LM significantly improves both the convergence speed and the accuracy of the model compared to first-order methods, achieving a noticeably lower mean squared error. This confirms its effectiveness in optimizing highly nonlinear and complex functions in neural networks.

Keywords: Levenberg–Marquardt; Neural Networks; Optimization; Convergence; Supervised Learning

Recibido: 16 febrero 2026 | Aceptado: 30 junio 2026 | Publicado1 julio 2026

INTRODUCCIÓN

El entrenamiento eficiente de redes neuronales artificiales constituye un aspecto fundamental en el ámbito del aprendizaje automático, especialmente en problemas que involucran funciones de error altamente no lineales y espacios de parámetros de gran complejidad. Los métodos tradicionales de optimización, como el descenso por gradiente, presentan limitaciones inherentes relacionadas con la lentitud en la convergencia, la sensibilidad a la tasa de aprendizaje y la propensión a quedar atrapados en mínimos locales, lo que puede comprometer la estabilidad del proceso de aprendizaje (A. Ranganath, 2024).

Ante estas limitaciones, el algoritmo de Levenberg–Marquardt (LM) se ha consolidado como una alternativa eficiente para la optimización de redes neuronales multicapa. Este método combina la rapidez del algoritmo de Gauss–Newton con la estabilidad del gradiente descendente, introduciendo un parámetro de amortiguamiento λ que se ajusta dinámicamente durante el proceso de entrenamiento. Dicho parámetro permite controlar el tamaño del paso de actualización, logrando un equilibrio entre la exploración del espacio de soluciones y la estabilidad numérica del procedimiento (Adhirai Subramaniyam, 2019).

El enfoque de Levenberg–Marquardt puede considerarse una aproximación de segundo orden al minimizar el error cuadrático medio, sin requerir el cálculo explícito del Hessiano. Gracias a esta característica, el algoritmo presenta una convergencia más rápida y precisa en comparación con los métodos de primer orden, lo que lo hace especialmente adecuado para problemas de regresión no lineal, predicción y clasificación (Adrien B. Taylor J. H., 2015).

METODOLOGÍA

El algoritmo de Levenberg-Marquardt es una variación del método de Newton, que fue diseñado para minimizar funciones que son sumas de cuadrados de otras funciones no lineales. Esto resulta muy adecuado para el entrenamiento de redes neuronales donde el índice de rendimiento que es el error cuadrático medio. Este texto se centra en demostrar como el algoritmo fusiona lo mejor de los algoritmos (Newton y Gradiente Descendente) más eficientes para solución numérica, integrándolo en el entrenamiento de redes neuronales paramétricas multicapa (Adrien B. Taylor Y. D., 2021).

Ahora bien, para comprender el algoritmo hay que sentar las bases para descifrarlo. Para ello nos centraremos en las superficies de rendimiento y puntos óptimos y optimización

del rendimiento, tomando en cuenta algoritmos de solución numérica y series de potencias (Alexandru Damian, 2022).

Superficie de rendimiento y punto optimo

Aprendizaje y rendimiento

Existen varias leyes de aprendizaje diferentes que se incluyen en la categoría de aprendizaje por rendimiento o desempeño. Estas leyes de aprendizaje se distinguen por el hecho de que durante el entrenamiento los parámetros de la red (pesos y bias o sesgos) se ajustan en un esfuerzo para optimizar el rendimiento de la red (Antoine Lesage-Landry, 2020).

Índice de rendimiento

Hay dos pasos involucrados en este proceso de optimización. El primero es definir que entendemos por rendimiento. En otras palabras, debemos encontrar una medida cuantitativa del desempeño de la red, llamada índice de rendimiento, que es pequeño cuando la red funciona bien y grande cuando la red funciona mal (Aryan Mokhtari, 2017).

El segundo paso del proceso de optimización es de buscar el espacio de parámetros (ajustar los pesos y sesgos de la red) para reducir índice de rendimiento (Mou Wu, 2020).

Serie de Taylor

Supongamos que el índice de rendimiento que se quiere minimizar está representado por $F(x)$, donde x es el parámetro escalar que estamos ajustando. Supondremos que el índice de rendimiento es una función analítica (P. Baldi, 2016). Entonces puede representarse mediante su desarrollo en la *expansión de la serie de Taylor* sobre algún punto nominal cualquiera x^*:

DOI: https://doi.org/10.71112/fdzkm898

$$F(x)=F(x^*)+\frac{d}{dx}F(x)\Big|_{x=x^*}(x-x^*)+\frac{1}{2}\frac{d^2}{dx^2}F(x)\Big|_{x=x^*}(x-x^*)^2+\ldots+\frac{1}{n!}\frac{d^n}{dx^n}F(x)\Big|_{x=x^*}(x-x^*)^n+\ldots \quad (1)$$

Caso vectorial.

Por supuesto, el índice de rendimiento de la red neuronal no será una función de un escalar x. Será función de todos los parámetros de la red (pesos y bias), de los cuales puede haber un número muy grande. Por lo tanto, necesitamos extender la expansión de Taylor a funciones de muchas variables (C. Uriarte, 2024).

Consideremos la siguiente función de variables:

$$F(x)=F(x_1,x_2,\ldots,x_n) \quad (2)$$

El desarrollo de la serie de Taylor para esta función, respecto al punto x^*, seria:

$$F(x)=F(x^*)+\frac{\partial}{\partial x_1}F(x)\Big|_{x=x^*}(x_1-x_1^*)+\frac{\partial}{\partial x_2}F(x)\Big|_{x=x^*}(x_2-x_2^*)+\ldots+\frac{\partial}{\partial x_n}F(x)\Big|_{x=x^*}(x_n-x_n^*)+\frac{1}{2}\frac{\partial^2}{\partial x_1^2}F(x)\Big|_{x=x^*}(x_1-x_1^*)^2+\frac{1}{2}\frac{\partial^2}{\partial x_1\partial x_2}F(x)\Big|_{x=x^*}(x_1-x_1^*)(x_2-x_2^*)+\ldots \quad (3)$$

Esta notación se nota engorrosa y compleja, pero es más conveniente escribirla de la siguiente forma:

$$F(x)=F(x^*)+\nabla F(x)^T\Big|_{x=x^*}(x-x^*)+\frac{1}{2}(x-x^*)^T\nabla^2F(x)\Big|_{x=x^*}(x-x^*)+\ldots \quad (4)$$

Donde $\nabla F(x)$ es el gradiente, y se define como:

DOI: https://doi.org/10.71112/fdzkm898

$$\nabla F(x) = \left[\frac{\partial}{\partial x_1} F(x) \frac{\partial}{\partial x_2} F(x) ... \frac{\partial}{\partial x_n} F(x) \right]^T \tag{5}$$

Y $\nabla^2 F(x)$ es el Hessiano, y se define como:

$$\nabla^2 F(x) = \begin{bmatrix} \frac{\partial^2}{\partial x_1^2} F(x) & \frac{\partial^2}{\partial x_1 \partial x_2} F(x) & \cdots & \frac{\partial^2}{\partial x_1 \partial x_n} F(x) \\ \frac{\partial^2}{\partial x_2 \partial x_1} F(x) & \frac{\partial^2}{\partial x_2^2} F(x) & \cdots & \frac{\partial^2}{\partial x_2 \partial x_n} F(x) \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2}{\partial x_n \partial x_1} F(x) & \frac{\partial^2}{\partial x_n \partial x_2} F(x) & \cdots & \frac{\partial^2}{\partial x_n^2} F(x) \end{bmatrix} \tag{6}$$

El gradiente y el Hessiano son muy importantes para comprender las superficies de rendimiento utilizando la derivada de dirección (Chunfu Guo, 2023).

Derivada direccional

El enésimo elemento del gradiente, $\partial F(x)/\partial x_i$, es la primera derivada del índice de rendimiento F a lo largo del eje x_i . El enésimo elemento de la diagonal F de la matriz Hessiana, $\partial^2 F(x)/\partial x_i^2$, es la segunda derivada del índice de rendimiento F a lo largo del eje x_i . ¿Qué pasaría si necesitamos saber la derivada de la función en una dirección arbitraria? Sea P un vector en la dirección en la que deseamos conocer la derivada (R. B. Yunus, 2024). Esta derivada direccional puede ser calculada a partir de:

$$\frac{p^T \nabla F(x)}{\|p\|} \tag{7}$$

DOI: https://doi.org/10.71112/fdzkm898

La segunda derivada a lo largo de P también se puede calcular.

$$\frac{p^T \nabla^2 F(x) p}{\|p\|^2} \tag{8}$$

Mínimos

Recordemos que el objetivo del aprendizaje del rendimiento será optimizar el índice de rendimiento de la red. Definiremos que se entiende por punto óptimo. Supongamos que el punto óptimo es un mínimo del índice de rendimiento. Las definiciones se pueden modificar fácilmente para problemas de maximización (D. Rotondo, 2024).

Mínimo fuerte.

El punto x^* es un mínimo fuerte de x^* si un escalar $\delta > 0$ existe, tal que $F(x^*) < F(x^* + \Delta x)$ para todo Δx tal que $\delta > \|\Delta x\| > 0$.

En otras palabras, si nos alejamos de un mínimo fuerte una pequeña distancia en cualquier función, la función aumentara.

Mínimo global.

El punto x^* es un mínimo global único de $F(x)$ si $F(x^*) < F(x^* + \Delta x)$ para todo $\Delta x \neq 0$.

Para un mínimo fuerte simple, x^*, la función puede ser menor que $F(x^*)$ en algunos puntos fuera de un pequeño grupo de x^*. Por lo tanto, a esto a veces de le llama mínimo local. Para un mínimo global, la función será mayor que el punto mínimo en cualquier otro punto del espacio de parámetros (Damien Lee, 2023).

Mínimo débil.

El punto x^* es un mínimo débil de $F(x)$ si no es un mínimo fuerte, y el escalar $\delta > 0$ existe, tal que $F(x^*) \leq F(x^* + \Delta x)$ para todo Δx tal que $\delta > \|\Delta x\| > 0$.

DOI: https://doi.org/10.71112/fdzkm898

No importa en qué dirección nos alejemos del mínimo débil, la función no puede disminuir, aunque puede haber algunas direcciones en las que no cambie (Dongsheng Guo, 2017).

Condiciones necesarias para la optimización

Habiendo definido lo que ahora entendemos por punto óptimo (mínimo), pasaremos a identificar algunas condiciones que dicho punto debe de satisfacer (R. Dehghani, 2018). Usando nuevamente la expansión de Taylor para derivar estas condiciones:

$$F(x) = F(x^* + \Delta x) = F(x^*) + \nabla F(x)^T \Big|_{x=x^*} \Delta x + \frac{1}{2} \Delta x^T \nabla^2 F(x) \Big|_{x=x^*} \Delta x + \ldots \quad (9)$$

Donde

$$\Delta x = x - x^* \quad (10)$$

Condiciones de primer orden

Si $\|\Delta x\|$ es muy pequeño, entonces los términos de orden superior en la ecuación (9) será insignificante y podemos aproximar la función como:

$$F(x^* + \Delta x) \cong F(x^*) + \nabla F(x)^T \Big|_{x=x^*} \Delta x \quad (11)$$

El punto x^* es un punto mínimo candidato, lo que significa que la función debe subir (o al menos no bajar) si Δx no es cero. Para que esto suceda, el segundo término de la ecuación (11) no debería ser negativo (Shuvomoy Das Gupta, 2023). En otras palabras:

$$\nabla F(x)^T \Big|_{x=x^*} \Delta x \geq 0 \quad (12)$$

DOI: https://doi.org/10.71112/fdzkm898

Sin embargo, si el termino es positivo,

$$\nabla F(x)^T \Big|_{x=x^*} \Delta x > 0 \tag{13}$$

Entonces, esto implicaría que

$$F(x^* + \Delta x) \cong F(x^*) - \nabla F(x)^T \Big|_{x=x^*} \Delta x < F(x^*) \tag{14}$$

Pero esto es una contradicción, ya que x^* debería ser un punto mínimo. Por lo tanto, la ecuación (12) debe ser cierta y la ecuación (13) debe ser falsa, la única alternativa debe ser que:

$$\nabla F(x)^T \Big|_{x=x^*} \Delta x = 0 \tag{15}$$

Como esto debe ser cierto para cualquier Δx, tenemos

$$\nabla F(x)\Big|_{x=x^*} = 0 \tag{16}$$

Por lo tanto, el gradiente debe ser cero en un punto mínimo. Esta es una condición de primer orden, necesaria (pero no suficiente) para que x^* sea un mínimo local. Cualquier punto que satisfaga la ecuación (16) se llaman puntos estacionarios (Gleich, 2017).

Condiciones de segundo orden

Supongamos que tenemos un punto estacionario x^*. Dado que el gradiente de $F(x)$ es cero en todos los puntos estacionarios, la expansión de la serie de Taylor será

$$F(x^* + \Delta x) = F(x^*) + \frac{1}{2} \Delta x^T \nabla^2 F(x)\Big|_{x=x^*} \Delta x + \ldots \tag{17}$$

Como antes se consideró solo aquellos puntos en un pequeño grupo de x^*, de modo que $\|\Delta x\|$ sea pequeño y $F(x)$ pueda aproximarse mediante los dos primeros términos de la ecuación (17). Por lo tanto, existirá un mínimo fuerte en x^* si

$$\Delta x^T \nabla^2 F(x)\Big|_{x=x^*} \Delta x > 0 \tag{18}$$

Funciones cuadráticas

Hay un tipo de índice de rendimiento que es universal: la función cuadrática. Esto es cierto porque hay muchas aplicaciones en las que aparece la función cuadrática, pero también porque muchas funciones pueden aproximarse por medio de estas funciones en grupos pequeños, especialmente cerca de puntos mínimos locales (Hawraz N. Jabbar, 2021).

Usaremos la siguiente forma general de la función cuadrática:

$$F(x) = \frac{1}{2} x^T A x + d^T x + c \tag{19}$$

Donde la matriz A es simétrica. (Si la matriz no es simétrica se puede reemplazar por la matriz simétrica que produzca $F(x)$ (Huanshui Zhang, 2024).

Para encontrar el gradiente de esta función, usaremos las siguientes propiedades útiles del gradiente:

$$\nabla(h^T x) = \nabla(x^T h) = h \tag{20}$$

Donde h es un vector constante, y:

$$\nabla x^T Q x = Q x + Q^T x = 2Qx \tag{21}$$

Ahora podemos calcular el gradiente de $F(x)$:

$$\nabla F(x) = Ax + d \tag{22}$$

Y de manera similar podemos encontrar el Hessiano:

$$\nabla^2 F(x) = A \tag{23}$$

Optimización del rendimiento

Ya que se tiene comprendido como analizar una superficie de rendimiento para determinar las condiciones que deben satisfacer los puntos óptimos, usaremos nuevamente la expansión de la serie de Taylor, en este caso para desarrollar un algoritmo que localice los puntos óptimos. Agregando a todo, cabe aclarar que cada algoritmo es iterativo (Ivan Perez Avellaneda, 2023).

Para nuestro propósito, la palabra *optimizar* significara encontrar el valor de x que minimice $F(x)$ (Shuvomoy Das Gupta, 2023). Se parte de una suposición inicial, x_0, y luego se actualiza esa suposición en etapas de acuerdo con una ecuación de la forma:

$$x_{k+1} = x_k + \alpha_k p_k \tag{24}$$

o de la forma:

$$\Delta x_k = (x_{k+1} - x_k) = \alpha_k p_k \tag{25}$$

Donde el vector p_k representa una dirección de búsqueda, y el escalar positivo α_k es la taza de aprendizaje, que determina la longitud del paso o el desplazamiento (Jaehoon Lee, 2019).

Gradiente descendente

Cuando se actualiza la estimación del punto óptimo (mínimo) usando la ecuación (24), se quiere que la función disminuya en cada iteración. En otras palabras:

$$F(x_{k+1}) < F(x_k) \tag{26}$$

DOI: https://doi.org/10.71112/fdzkm898

Aquí se genera una interrogante ¿Cómo podemos elegir una dirección, p_k, de modo que, para una tasa de aprendizaje, α_k, lo suficientemente pequeña, esta pueda ir en descenso? Podemos considerar la expansión de la serie de Taylor de primer orden (ver ecuación (4)) de $F(x)$ sobre la suposición x_k:

$$F(x_{k+1}) = F(x_k + \Delta x_k) \approx F(x_k) + g_k^T \Delta x_k \tag{27}$$

Donde g_k es el gradiente evaluado en la suposición x_k:

$$g_k \equiv \nabla F(x)\Big|_{x=x_k} \tag{28}$$

Para que $F(x_{k+1})$ sea menor que $F(x_k)$, el segundo termino en el lado derecho de la ecuación (27) debe ser negativo:

$$g_k^T \Delta x_k = \alpha_k g_k^T p_k < 0 \tag{29}$$

Ahora se selecciona un α_k que sea más pequeño, pero mayor que cero. Esto implica que:

$$g_k^T p_k < 0 \tag{30}$$

Dirección de descenso

Cualquier vector p_k que satisfaga la ecuación se llama dirección de descenso. La función debe disminuir si damos un paso lo suficientemente pequeño en esa dirección. Esto nos lleva a otra pregunta ¿Cuál es la dirección del descenso más pronunciado? (¿En qué dirección la función disminuirá más rápidamente?).

Esto ocurrirá cuando

$$g_k^T p_k \tag{31}$$

es más negativo. (Se asume que la longitud de p_k no cambia, solo su dirección). Este es un producto interno entre el gradiente y el vector de dirección. Será más negativo cuando el vector de dirección sea el gradiente. Por lo tanto, un vector que apunta en la dirección de descenso más pronunciada es

$$P_k = -g_k \tag{32}$$

Descenso más pronunciado (gradiente descendente)

Usando esto en la iteración de la ecuación (24) se produce el método del descenso más pronunciado o descenso de gradiente:

$$x_{k+1} = x_k - \alpha_k p_k \tag{33}$$

Tasa de aprendizaje.

Para el descenso de gradiente, existen dos métodos generales para determinar la *tasa de aprendizaje*, a_k. Un enfoque es minimizar el índice de rendimiento $F(x)$ con respecto a a_k en cada iteración. En este caso estamos minimizando a lo largo de una línea representada por:

$$x_k - \alpha_k p_k \tag{34}$$

Método de Newton

La derivación del algoritmo del descenso de gradiente se basó en la expansión de la serie de Taylor de primer orden. El metodo de Newton se basa en la serie de Taylor de segundo orden (Taylor Simons, 2019):

$$F(x_{k+1}) = F(x_k + \Delta x_k) \approx F(x_k) + g_k^T \Delta x_k + \frac{1}{2} \Delta x_k^T A_k \Delta x_k \tag{35}$$

El principio detrás de este método es localizar el punto estacionario de una aproximación cuadrática de $F(x)$. Si utilizamos la ecuación (22) para tomar el gradiente de esta función cuadrática con respecto a Δx_k e igualarlo a cero, obtenemos:

$$g_k + A_k \Delta x_k = 0 \tag{36}$$

Resolviendo para Δx_k tenemos:

$$\Delta x_k = -A_k^{-1} g_k \tag{37}$$

Entonces se define el método de Newton como:

$$x_{k+1} = x_k - A_k^{-1} g_k \tag{38}$$

RESULTADOS

Algoritmo Levenberg-Marquardt

Comencemos considerando la forma del método de Newton donde el índice de rendimiento es una suma de cuadrados (Taylor Simons, 2019). Recordemos que el método de Newton para optimizar el índice de rendimiento $F(x)$ es (38) donde:

$$\begin{aligned} A_k &\equiv \nabla^2 F(x)\Big|_{x=x_k} \\ g_k &\equiv \nabla F(x)\Big|_{x=x_k} \end{aligned} \tag{39}$$

Si asumimos que $F(x)$ es una función de suma de cuadrados:

$$F(x) = \sum_{i=1}^{N} v_i^2(x) = v^T(x) v(x) \tag{40}$$

Entonces, el *j*th (j ésimo) elemento del gradiente seria:

$$\left[\nabla F(x)\right]_j \frac{\partial F(x)}{\partial x_j} = 2\sum_{i=1}^{N} v_i(x) \frac{\partial v_i(x)}{\partial x_j} \tag{41}$$

Por lo tanto, el gradiente se puede expresar en forma matricial:

$$\nabla F(x) = 2J^T(x) v(x) \tag{42}$$

DOI: https://doi.org/10.71112/fdzkm898

Donde:

$$J(x)=\begin{bmatrix} \frac{\partial v_1(x)}{\partial x_1} & \frac{\partial v_1(x)}{\partial x_2} & \cdots & \frac{\partial v_1(x)}{\partial x_n} \\ \frac{\partial v_2(x)}{\partial x_1} & \frac{\partial v_2(x)}{\partial x_2} & \cdots & \frac{\partial v_2(x)}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial v_N(x)}{\partial x_1} & \frac{\partial v_N(x)}{\partial x_2} & \cdots & \frac{\partial v_N(x)}{\partial x_n} \end{bmatrix} \tag{43}$$

La expresión anterior es la matriz Jacobiana.

A continuación, encontramos la matriz Hessiana. Los k,j elementos de la matriz serian:

$$\left[\nabla^2 F(x)\right]_{k,j}=\frac{\partial^2 F(x)}{\partial x_k \partial x_j}=2\sum_{i=1}^{N}\left\{\frac{\partial v_i(x)}{\partial x_k}\frac{\partial v_i(x)}{\partial x_j}+v_i(x)\frac{\partial^2 v_i(x)}{\partial x_k \partial x_j}\right\} \tag{44}$$

La matriz Hessiana de los elementos se puede expresar en su forma matricial:

$$\nabla^2 F(x)=2J^T(x)J(x)+2S(x) \tag{45}$$

Donde

$$S(x)=\sum_{i=1}^{N} v_i(x)\nabla^2 v_i(x) \tag{46}$$

Si se asume que $S(x)$ es pequeño, podemos aproximar la matriz Hessiana como:

$$\nabla^2 F(x)=2J^T(x)J(x)+2S(x) \tag{47}$$

Si luego se sustituye la ecuación (47) y la ecuación (42) en la ecuación (38), obtendremos el método de Gauss-Newton:

$$\begin{aligned} x_{k+1} &= x_k-\left[2J^T(x_k)J(x_k)\right]^{-1}2J^T(x_k)v(x_k) \\ &= x_k-\left[J^T(x_k)J(x_k)\right]^{-1}J^T(x_k)v(x_k) \end{aligned} \tag{48}$$

La gran ventaja del algoritmo de Gauss-Newton sobre el método estándar de Newton es que no requiere el cálculo de segundas derivadas (Jamie M. Taylor, 2022).

Un problema con el método de Gauss-Newton es que la matriz $H = J^T J$ puede no ser invertida (Weilong Liao, 2022). Este problema se puede superar utilizando la siguiente modificación de la matriz Hessiana aproximada:

$$G = H + \mu I \tag{49}$$

Para determinar cómo esta matriz puede hacerse invertible, podemos suponer que los valores y vectores propios (eigenvalues & eigenvectors) de H son $\{\lambda_1, \quad \lambda_2, \quad ..., \quad \lambda_n\}$ y $\{z_1, \quad z_2, \quad ..., \quad z_n\}$ (Yuan-Yuan Cheng, 2025). Se tiene:

$$Gz_i = [H + \mu I]z_i = Hz_i + \mu z_i = \lambda_i z_i + \mu z_i = (\lambda_i + \mu)z_i \tag{50}$$

Por lo tanto, los vectores propios de G son los mismos vectores propios de H, y los valores propios de G son $(\lambda_i + \mu)$. G puede definirse como positivo aumentando μ hasta $(\lambda_i + \mu) > 0$ para todos i, y de esa manera la matriz será invertible (Jesse Chan, 2020).

Esto nos lleva al algoritmo ***Levenberg-Marquardt***:

$$x_{k+1} = x_k - \left[J^T(x_k)J(x_k) + \mu_k I\right]^{-1} J^T(x_k)v(x_k) \tag{51}$$

O:

$$\Delta x_k = -\left[J^T(x_k)J(x_k) + \mu_k I\right]^{-1} J^T(x_k)v(x_k) \tag{52}$$

Este algoritmo tiene la característica muy útil de que, a medida que μ_k se aumenta, se acerca al algoritmo del gradiente descendente con una taza de aprendizaje muy pequeña:

$$x_{k+1} \cong x_k - \frac{1}{\mu_k} J^T(x_k)v(x_k) = x_k - \frac{1}{2\mu_k}\nabla F(x) \tag{53}$$

DOI: https://doi.org/10.71112/fdzkm898

Mientras que μ_k se reduce a cero, el algoritmo de convierte en Gauss-Newton (Yunjin Tong, 2020).

El algoritmo comienza con μ_k estableciendo un valor pequeño (ejemplo, $\mu_k = 0.01$). Si un paso no produce un valor menor para $F(x)$, entonces el paso se repite con μ_k multiplicado con algún factor $\vartheta > 1$ (ejemplo, $\vartheta = 10$). Finalmente $F(x)$ debería disminuir, ya que estaríamos dando un paso en la dirección del gradiente descendente. Si un paso produce un valor menor para $F(x)$, entonces μ_k es dividido por ϑ para el siguiente paso, de modo que el algoritmo se acerque a Gauss-Newton, lo que debería proporcionar una convergencia más rápida. El algoritmo proporciona un buen compromiso entre la velocidad del método de Newton y la convergencia garantizada del gradiente descendente (John Taylor, 2022).

Ahora veamos cómo podemos aplicar el algoritmo al problema del entrenamiento de redes multicapa. El índice de rendimiento para el entrenamiento de redes multicapa es el error cuadrático medio. Si cada objetivo ocurre con la misma posibilidad, el error cuadrático medio es proporcional a la suma de los errores cuadráticos de los objetivos Q en el conjunto de entrenamiento:

$$\begin{aligned} F(x) &= \sum_{q=1}^{Q} \left(\mathrm{t}_q - \mathrm{a}_q\right)^T \left(\mathrm{t}_q - \mathrm{a}_q\right) \\ &= \sum_{q=1}^{Q} \mathrm{e}_q^T \mathrm{e}_q = \sum_{q=1}^{Q} \sum_{j=1}^{S^M} \left(e_{j,q}\right)^2 = \sum_{i=1}^{N} \left(v_i\right)^2 \end{aligned} \tag{54}$$

Donde $e_{j,q}$ es el j ésimo (j th) elemento del error para el q ésimo (q th) par entrada/objetivo del entrenamiento. S el número de neuronas por capa, a es la salida o resultado de la red, M el número de capas de la red (Marcus Carlsson, 2025).

La ecuación (54) es equivalente al índice de rendimiento, ecuación (40), para el cual se diseñó el algoritmo Levenberg-Marquardt. Por lo tanto, debería ser sencillo adaptar el

DOI: https://doi.org/10.71112/fdzkm898

algoritmo para el entrenamiento de la red. Resulta que esto es cierto en concepto, pero requiere cierto cuidado al resolver los detalles (Zheyuan Hu, 2023).

Comparación del Gradiente Descendente vs Levenberg-Marquardt

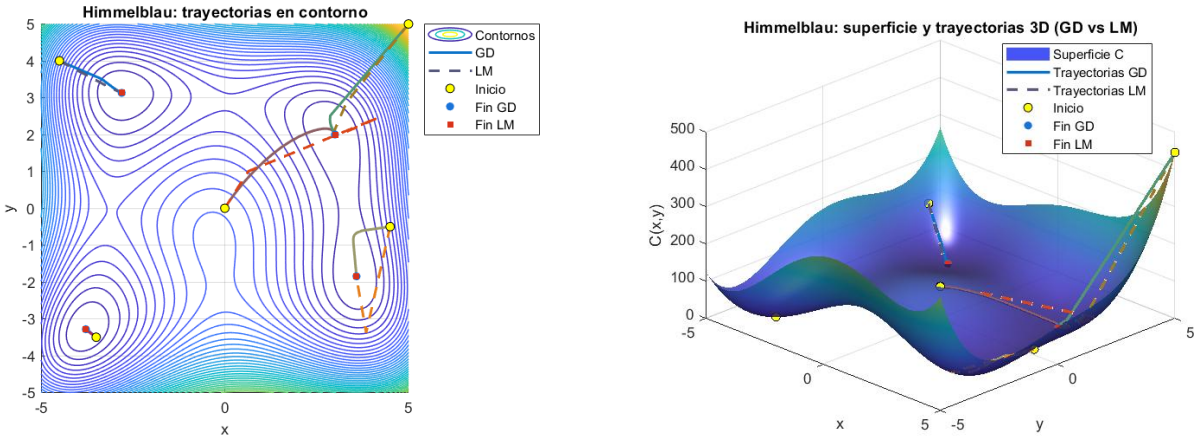
Caso 1. Función de Himmelblau

La Figura 1(a) y (b) ilustran la comparación entre los algoritmos de Descenso por Gradiente (GD) y Levenberg–Marquardt (LM) aplicados a la función de Himmelblau, una superficie clásica de prueba en optimización no lineal definida por:

$$C(x,y)=\frac{1}{2}\|r(x,y)\|^2=\frac{1}{2}\left[(x^2+y-11)^2+(x+y^2-7)^2\right] \quad (55)$$

Figura 1

Trayectorias con puntos arbitrarios con vistas en el plano superficial y tridimensional



a) b)

Los resultados demuestran que:

- LM desciende la superficie de forma casi directa hacia los valles de mínimo, aprovechando la información del Jacobiano para ajustar la dirección y magnitud del paso.
- GD, en cambio, desciende con movimientos más conservadores, perdiendo eficiencia cuando la superficie presenta curvaturas pronunciadas o zonas planas.

Ambos métodos alcanzan los mismos puntos mínimos (los cuatro valles característicos de Himmelblau), pero con un tiempo de convergencia radicalmente menor para LM (promedio de 5.8 iteraciones frente a 110.8 para GD).

Esto evidencia la capacidad del algoritmo LM para adaptar su paso mediante el parámetro de amortiguamiento λ, que le permite comportarse como Gauss–Newton cerca de la solución (rápido) y como GD cuando está lejos del óptimo (seguro).

Tabla 1

Análisis numérico de GD vs LM

Inicio	Mínimo alcanzado	Iteraciones GD	Iteraciones LM	Relación de mejora
(-4.5, 4.0)	(-2.805, 3.131)	56	5	≈ 11× más rápido
(0.0, 0.0)	(3.000, 2.000)	160	7	≈ 23× más rápido
(4.5, -0.5)	(3.584, -1.848)	152	7	≈ 22× más rápido
(-3.5, -3.5)	(-3.779, -3.283)	38	4	≈ 9× más rápido
(5.0, 5.0)	(3.000, 2.000)	148	6	≈ 24× más rápido

El análisis confirma que el método de Levenberg–Marquardt reduce drásticamente el número de iteraciones necesarias para alcanzar el mismo nivel de error, combinando la rapidez de Gauss–Newton con la estabilidad del descenso por gradiente.

En este contexto sobre redes neuronales, este comportamiento se traduce en una optimización más eficiente del error de entrenamiento, una convergencia más estable y una mejor generalización del modelo.

Caso 2. Entrenamiento de red neuronal y ajuste de función

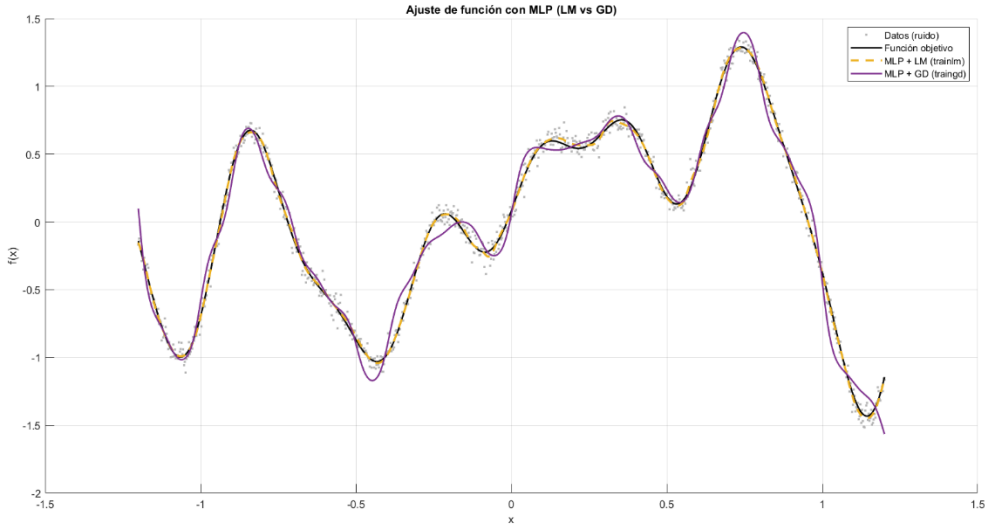
La Figura 2 muestra el ajuste de una red neuronal MLP de una sola capa oculta (25 neuronas) para una función altamente no lineal y oscilatoria definida como:

DOI: https://doi.org/10.71112/fdzkm898

$$f(x) = xsen(10x) + 0.5sen(3.2x) + 0.2sen(20x) + 0.3e^{-8(x-0.4)^2} \quad (56)$$

Figura 2

Ajuste de función oscilatoria altamente no lineal



El modelo entrenado con el algoritmo Levenberg–Marquardt (LM) logra reproducir con gran precisión la forma general y los detalles finos de la función objetivo, incluso en regiones de alta curvatura y comportamiento oscilatorio, mientras que el modelo basado en Gradiente Descendente (GD) alcanza un ajuste aceptable, aunque presenta ligeras desviaciones en las transiciones rápidas y una tendencia a suavizar los picos y valles característicos de la señal. Esta diferencia se explica porque LM ajusta los pesos neuronales incorporando información de segundo orden mediante la aproximación $(J^T J + \lambda I)$, lo que le permite realizar pasos de corrección más amplios y estables en direcciones de menor curvatura del error. En contraste, GD se fundamenta únicamente en la pendiente del gradiente, lo que produce un avance más lento y dependiente de la tasa de aprendizaje seleccionada, afectando la velocidad y la estabilidad del proceso de convergencia.

DOI: https://doi.org/10.71112/fdzkm898

DISCUSIÓN

Los resultados obtenidos en las simulaciones 2D, 3D y en el entrenamiento de la red neuronal confirman la superioridad del algoritmo Levenberg–Marquardt (LM) frente al Gradiente Descendente (GD). En los contornos bidimensionales, LM mostró trayectorias más directas y rápidas hacia los mínimos, mientras que GD presentó rutas más largas y oscilatorias. En el modelo tridimensional, LM descendió con mayor estabilidad por la superficie del error, aprovechando la información de curvatura que GD ignora. De forma análoga, en el ajuste de la función compleja, LM alcanzó una convergencia más rápida y precisa, reproduciendo fielmente las variaciones del objetivo.

CONCLUSIONES

El estudio realizado demuestra que el algoritmo Levenberg–Marquardt (LM) constituye una alternativa altamente eficiente para el entrenamiento de redes neuronales multicapa y la optimización de funciones no lineales. A partir de las simulaciones 2D y 3D, así como del experimento de ajuste de una función compleja, se comprobó que LM alcanza los mínimos con trayectorias más estables y directas, reduciendo de forma significativa el número de iteraciones necesarias respecto al Gradiente Descendente (GD). Su capacidad para adaptar dinámicamente el parámetro de amortiguamiento λ le permite equilibrar la rapidez de Gauss–Newton con la estabilidad del descenso por gradiente, logrando una convergencia más rápida, precisa y menos sensible a las condiciones iniciales. En consecuencia, se concluye que LM es un método robusto y confiable para optimizar el aprendizaje supervisado, mejorando la eficiencia y la calidad del ajuste en redes neuronales aplicadas a problemas de alta complejidad.

Declaración de conflicto de interés

El autor, Ernesto Antonio Morales Rodriguez, declara no tener ningún conflicto de interés relacionado con esta investigación.

Declaración de contribución a la autoría

Ernesto Antonio Morales Rodriguez: conceptualización, curación de datos, análisis formal, adquisición de fondos, investigación, metodología, administración del proyecto, recursos, software, supervisión, validación, visualización, redacción del borrador original, revisión y edición de la redacción.

Declaración de uso de inteligencia artificial

El autor, Ernesto Antonio Morales Rodriguez, declara que utilizo la inteligencia artificial como apoyo para este artículo, y también que esta herramienta no sustituye de ninguna manera la tarea o proceso intelectual. Después de rigurosas revisiones con diferentes herramientas en la que se comprobó que no existe plagio como constan en las evidencias, el autor manifiesta y reconoce que este trabajo fue producto de un trabajo intelectual propio, que no ha sido escrito ni publicado en ninguna plataforma electrónica o de IA.

REFERENCIAS

A. Ranganath, I. R. (2024). Quasi-Adam: Accelerating Adam Using Quasi-Newton Approximations. 2024 International Conference on Machine Learning and Applications (ICMLA). https://doi.org/10.1109/icmla61862.2024.00107

Adhirai Subramaniyam, R. M. (2019). Taylor and Gradient Descent-Based Actor Critic Neural Network for the Classification of Privacy Preserved Medical Data. Big data. https://doi.org/10.1089/big.2018.0166

Adrien B. Taylor, J. H. (2015). Exact Worst-Case Performance of First-Order Methods for Composite Convex Optimization. SIAM J. Optim. https://doi.org/10.1137/16m108104x

Adrien B. Taylor, Y. D. (2021). An optimal gradient method for smooth strongly convex minimization. Mathematical Programming. https://doi.org/10.1007/s10107-022-01839-y

Alexandru Damian, E. N. (2022). Self-Stabilization: The Implicit Bias of Gradient Descent at the Edge of Stability. ArXiv. https://doi.org/10.48550/arxiv.2209.15594

Antoine Lesage-Landry, J. A. (2020). Second-Order Online Nonconvex Optimization. IEEE Transactions on Automatic Control. https://doi.org/10.1109/tac.2020.3040372

Aryan Mokhtari, Q. L. (2017). Network Newton Distributed Optimization Methods. IEEE Transactions on Signal Processing. https://doi.org/10.1109/tsp.2016.2617829

C. Uriarte, M. B. (2024). Optimizing Variational Physics-Informed Neural Networks Using Least Squares. ArXiv. https://doi.org/10.1016/j.camwa.2025.02.022

Chunfu Guo, Y. S. (2023). A combination of library search and Levenberg-Marquardt algorithm in optical scatterometry. Thin Solid Films. https://doi.org/10.1016/j.tsf.2023.139670

D. Rotondo, M. J. (2024). A Weighted Linearization Approach to Gradient Descent Optimization. 2024 European Control Conference (ECC). https://doi.org/10.23919/ecc64448.2024.10590954

Damien Lee, K. N.-H.-W. (2023). A Continual Learning Algorithm Based on Orthogonal Gradient Descent Beyond Neural Tangent Kernel Regime. IEEE Access. https://doi.org/10.1109/access.2023.3303869

Dongsheng Guo, Z.-Y. N. (2017). Novel Discrete-Time Zhang Neural Network for Time-Varying Matrix Inversion. IEEE Transactions on Systems, Man, and Cybernetics: Systems. https://doi.org/10.1109/tsmc.2017.2656941

Gleich, D. (2017). TRUST REGION METHODS. https://doi.org/10.1007/978-0-387-40065-5_4

Hawraz N. Jabbar, B. A. (2021). Two-versions of descent conjugate gradient methods for large-scale unconstrained optimization. Indonesian Journal of Electrical Engineering and Computer Science. https://doi.org/10.11591/ijeecs.v22.i3.pp1643-1649

Huanshui Zhang, H. W. (2024). Optimization methods rooted in optimal control. Sci. China Inf. Sci. https://doi.org/10.1007/s11432-024-4207-5

Ivan Perez Avellaneda, L. A. (2023). Output Reachability of Chen-Fliess series: A Newton-Raphson Approach. 2023 57th Annual Conference on Information Sciences and Systems (CISS). https://doi.org/10.1109/ciss56502.2023.10089740

Jaehoon Lee, L. X.-D. (2019). Wide neural networks of any depth evolve as linear models under gradient descent. Journal of Statistical Mechanics: Theory and Experiment. https://doi.org/10.1088/1742-5468/abc62b

Jamie M. Taylor, D. P. (2022). A Deep Fourier Residual Method for solving PDEs using Neural Networks. ArXiv. https://doi.org/10.48550/arxiv.2210.14129

Jesse Chan, C. G. (2020). Efficient computation of Jacobian matrices for entropy stable summation-by-parts schemes. J. Comput. Phys. https://doi.org/10.1016/j.jcp.2021.110701

John Taylor, W. W. (2022). Optimizing the optimizer for data driven deep neural networks and physics informed neural networks. ArXiv. https://doi.org/10.48550/arxiv.2205.07430

Marcus Carlsson, V. N. (2025). The bilinear Hessian for large scale optimization. ArXiv. https://doi.org/10.48550/arxiv.2502.03070

Mou Wu, N. X. (2020). RNN-K: A Reinforced Newton Method for Consensus-Based Distributed Optimization and Control Over Multiagent Systems. IEEE Transactions on Cybernetics. https://doi.org/10.1109/tcyb.2020.3011819

P. Baldi, K. C. (2016). Parameterized neural networks for high-energy physics. The European Physical Journal C. https://doi.org/10.1140/epjc/s10052-016-4099-4

R. B. Yunus, N. Z.-Y. (2024). An improved accelerated 3-term conjugate gradient algorithm with second-order Hessian approximation for nonlinear least-squares optimization. Journal of Mathematics and Computer Science. https://doi.org/10.22436/jmcs.036.03.02

R. Dehghani, M. M. (2018). The modified quasi-Newton methods for solving unconstrained optimization problems. International Journal of Numerical Modelling: Electronic Networks. https://doi.org/10.1002/jnm.2459

Shuvomoy Das Gupta, R. F. (2023). Nonlinear conjugate gradient methods: worst-case convergence rates via computer-assisted analyses. Mathematical Programming. https://doi.org/10.1007/s10107-024-02127-7

Taylor Simons, D.-J. L. (2019). A Review of Binarized Neural Networks. Electronics. https://doi.org/10.3390/electronics8060661

Weilong Liao, Q. Z. (2022). An SQP Algorithm for Structural Topology Optimization Based on Majorization–Minimization Method. Applied Sciences. https://doi.org/10.3390/app12136304

Yuan-Yuan Cheng, L. L.-M. (2025). Eigenvalue Blending for Projected Newton. Computer Graphics Forum. https://doi.org/10.1111/cgf.70027

Yunjin Tong, S. X. (2020). Symplectic Neural Networks in Taylor Series Form for Hamiltonian Systems. ArXiv. https://doi.org/10.1016/j.jcp.2021.110325

Zheyuan Hu, Z. S. (2023). Hutchinson Trace Estimation for High-Dimensional and High-Order Physics-Informed Neural Networks. ArXiv. https://doi.org/10.1016/j.cma.2024.116883