Métodos de Newton y quasi-Newton para optimización en varias variables

Newton and Quasi-Newton Methods for Optimization in Several Variables

Josué Amador-Rojas y Marcelo Picado-Leiva

Universidad de Costa Rica

{bryan.amador, marcelo.picado}@ucr.ac.cr

27 de junio de 2026

Resumen: Este trabajo estudia el método de Newton aplicado a problemas de optimización en varias variables. Se presenta su formulación matemática a partir de la búsqueda de raíces del gradiente y se analiza el papel de la matriz hessiana en la construcción de la dirección de búsqueda. Además, se discuten algunas limitaciones del método, como la dependencia del punto inicial, la necesidad de derivadas de segundo orden y el costo asociado al cálculo del Hessiano. Para ilustrar estos conceptos se desarrollan dos casos de estudio: la optimización de la función de Rosenbrock mediante el método de Newton y el entrenamiento de una red neuronal para aproximar la función seno utilizando el optimizador L-BFGS. Los resultados muestran que el método de Newton permite comprender de forma clara el uso de información de segundo orden, mientras que L-BFGS ofrece una alternativa eficiente para problemas de mayor dimensión al aproximar la información de curvatura sin calcular explícitamente la matriz hessiana.

Palabras clave: Aprendizaje automático, L-BFGS, método de Newton, optimización, redes neuronales.

Abstract: This work studies Newton's method applied to optimization problems in several variables. Its mathematical formulation is presented from the perspective of finding roots of the gradient, and the role of the Hessian matrix in the construction of the search direction is analyzed. In addition, some limitations of the method are discussed, such as its dependence on the initial point, the need for second-order derivatives, and the computational cost associated with the Hessian. To illustrate these concepts, two case studies are developed: the optimization of the Rosenbrock function using Newton's method and the training of a neural network to approximate the sine function using the L-BFGS optimizer. The results show that Newton's method provides a clear understanding of second-order information, while L-BFGS offers an efficient alternative for higher-dimensional problems by approximating curvature information without explicitly computing the Hessian matrix.

Keywords: L-BFGS, machine learning, neural networks, Newton's method, optimization.

1. Introducción

En el análisis numérico, uno de los problemas fundamentales consiste en encontrar soluciones aproximadas a ecuaciones que no pueden resolverse de forma analítica. En este contexto, el método de Newton-Raphson destaca como una herramienta iterativa de rápida convergencia para la resolución de ecuaciones no lineales.

La extensión de este método al caso multivariable permite no solo resolver sistemas de ecuaciones, sino también abordar problemas de optimización. En particular, muchos problemas de optimización pueden reformularse como problemas de búsqueda de raíces al considerar las condiciones de optimalidad de primer orden, las cuales establecen que el gradiente de la función objetivo debe anularse.

Los problemas de optimización aparecen de manera recurrente en áreas como investigación de operaciones, inteligencia artificial y aprendizaje automático, donde frecuentemente es necesario minimizar funciones de pérdida de alta dimensión. Sin embargo, la aplicación directa del método de Newton presenta limitaciones importantes, especialmente el costo computacional asociado al cálculo e inversión de la matriz hessiana.

Con el fin de mitigar estas dificultades, surgieron métodos quasi-Newton como BFGS y L-BFGS, los cuales aproximan la información de curvatura sin calcular explícitamente el Hessiano.

El presente trabajo estudia la extensión del método de Newton al contexto de optimización multivariable, así como algunas de sus principales limitaciones y estrategias de mejora. Para ello, se desarrolla un caso de estudio dividido en dos etapas: una aplicación manual del método de Newton sobre una función de baja dimensión y el entrenamiento de una red neuronal pequeña mediante el método L-BFGS para aproximar una función matemática diferenciable.

2. Marco teórico

2.1. Método de Newton-Raphson en una variable

El método de Newton-Raphson se describe como un algoritmo iterativo utilizado para aproximar raíces de funciones no lineales $f : \mathbb{R} \to \mathbb{R}$ [1, pp. 67–71]. Dado un punto inicial x_0, el método genera una sucesión de aproximaciones mediante la iteración mostrada en la ecuación 1.

$$\begin{cases} x_0 \\ x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \end{cases} \tag{1}$$

La iteración puede justificarse mediante la expansión de Taylor de primer orden alrededor de un punto x_k:

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k).$$

Imponiendo la condición $f(x) = 0$ y despejando x, se obtiene directamente la iteración de Newton.

El método presenta convergencia local bajo condiciones adecuadas de suavidad de la función y siempre que la derivada en la raíz sea distinta de cero. No obstante, su comportamiento depende fuertemente de la elección del punto inicial y requiere que $f'(x)$ exista y no se anule durante las iteraciones.

2.2. Extensión de Newton a varias variables

De acuerdo con [1, pp. 638–643], el método de Newton puede extenderse al caso multivariable para resolver sistemas no lineales de la forma:

$$F(\mathbf{x}) = \mathbf{0},$$

donde $F : \mathbb{R}^n \to \mathbb{R}^n$ es una función vectorial definida por:

$$F(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{bmatrix}.$$

La extensión del método surge de la aproximación de Taylor de primer orden para funciones vectoriales:

$$F(\mathbf{x}) \approx F(\mathbf{x}^{(k)}) + J_F(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)}),$$

donde $J_F(\mathbf{x})$ es la matriz jacobiana de F, definida como [2, p. 330]:

$$J_F(\mathbf{x}) = \left[\frac{\partial f_i}{\partial x_j}\right]_{i,j=1}^{n}.$$

Imponiendo la condición $F(\mathbf{x}) = \mathbf{0}$ sobre la aproximación lineal, se obtiene:

$$J_F(\mathbf{x}^{(k)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = -F(\mathbf{x}^{(k)}).$$

Si la matriz jacobiana es invertible, la iteración de Newton en varias variables queda dada por la ecuación 2.

$$\begin{cases} \mathbf{x}^{(0)} \\ \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - J_F(\mathbf{x}^{(k)})^{-1} F(\mathbf{x}^{(k)}) \end{cases} \tag{2}$$

En la práctica, la inversión explícita de la matriz jacobiana suele evitarse debido a su costo computacional, resolviendo en su lugar sistemas lineales equivalentes [1, p. 640].

2.3. Optimización como problema de raíces

Los problemas de optimización desempeñan un papel fundamental en áreas como investigación de operaciones e inteligencia artificial, donde frecuentemente se busca minimizar o maximizar funciones bajo condiciones no lineales [3], [4].

En el caso de una función escalar de una variable $f : \mathbb{R} \to \mathbb{R}$, una condición necesaria para que un punto x^* sea un extremo local es que:

$$f'(x^*) = 0.$$

Por lo tanto, un problema de optimización puede reformularse como un problema de búsqueda de raíces aplicado a la derivada de la función [5, p. 287]. Sustituyendo $f(x)$ por $f'(x)$ en la iteración de Newton de la ecuación 1, se obtiene la ecuación 3:

$$\begin{cases} x_0 \\ x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \end{cases} \tag{3}$$

lo que muestra que el método utiliza tanto la primera como la segunda derivada de la función original para aproximar puntos óptimos [3, pp. 513–515].

Para una función escalar multivariable $f : \mathbb{R}^n \to \mathbb{R}$, la condición necesaria de optimalidad se expresa mediante el gradiente [2, p. 317]:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix},$$

de modo que los puntos críticos satisfacen:

$$\nabla f(\mathbf{x}) = \mathbf{0}.$$

Así, el problema de optimización se transforma nuevamente en la resolución de un sistema no lineal, permitiendo aplicar el método de Newton multivariable de la ecuación 2. Sustituyendo $F(\mathbf{x})$ por $\nabla f(\mathbf{x})$, se obtiene:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - J_{\nabla f}(\mathbf{x}^{(k)})^{-1} \nabla f(\mathbf{x}^{(k)}).$$

La matriz jacobiana del gradiente corresponde a la matriz hessiana de la función [2, p. 375]:

$$H_f(\mathbf{x}) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j} \right]_{i,j=1}^{n}.$$

Por lo tanto, la iteración de Newton para optimización multivariable queda dada por:

$$\begin{cases} \mathbf{x}^{(0)} \\ \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - H_f(\mathbf{x}^{(k)})^{-1} \nabla f(\mathbf{x}^{(k)}) \end{cases} \tag{4}$$

Dado que esta formulación corresponde a una aplicación directa del método de Newton, su convergencia depende de condiciones similares a las discutidas previamente, particularmente de la suavidad de la función y de la elección de un punto inicial adecuado.

2.4. Limitaciones del método de Newton para optimización

Dado que la formulación utilizada en optimización corresponde a una aplicación directa del método de Newton para la resolución de ecuaciones no lineales, las limitaciones del método se trasladan naturalmente al contexto de optimización.

Una de las principales dificultades del método radica en su dependencia de un punto inicial adecuado [5, p. 214]. Debido a que la convergencia es de carácter local, un punto inicial alejado de la solución puede provocar divergencia o convergencia hacia un punto no deseado.

Asimismo, el método requiere que la función objetivo sea suficientemente suave, en particular, que sus derivadas de primer y segundo orden existan y sean continuas en la región de interés [5, p. 289]. Esta condición resulta necesaria para garantizar la validez de la aproximación local utilizada por el método.

Otra limitación importante corresponde a la necesidad de que la matriz hessiana sea invertible en cada iteración. En caso contrario, no es posible resolver el sistema lineal asociado. Incluso cuando la matriz es invertible, su cálculo, almacenamiento e inversión pueden resultar computacionalmente costosos en problemas de alta dimensión. Para un problema con n variables, el almacenamiento explícito del hessiano requiere memoria del orden de $O(n^2)$, mientras que el cálculo y resolución del sistema lineal asociado pueden alcanzar costos del orden de $O(n^3)$ utilizando métodos directos tradicionales [1], [6]. En consecuencia, el método puede volverse impráctico cuando el número de variables es muy grande.

Finalmente, en el contexto de optimización, el método puede converger hacia distintos tipos de puntos críticos, ya que todos ellos satisfacen la condición necesaria de optimalidad:

$$\nabla f(\mathbf{x}) = \mathbf{0}.$$

Aunque la matriz hessiana permite clasificar dichos puntos mediante criterios de segundo orden, el método de Newton clásico no incorpora mecanismos explícitos para distinguir o evitar máximos locales y puntos de silla durante el proceso iterativo. Como consecuencia, el tipo de punto crítico al que converge depende tanto de la geometría de la función como de la elección del punto inicial.

Diversas estrategias han sido propuestas para mitigar algunas de estas limitaciones. Entre ellas destacan la regularización de la matriz hessiana mediante términos de la forma

$$H_f(\mathbf{x}) + \lambda I,$$

los cuales favorecen una curvatura más estable y reducen problemas asociados a matrices singulares o indefinidas [6, pp. 48–53]. Asimismo, pueden incorporarse estrategias de control de paso mediante factores de escala α_k, con el fin de reducir el riesgo de divergencia y mejorar la convergencia en regiones donde la aproximación local del método no resulta suficientemente precisa [6, pp. 37–41].

No obstante, incluso con estas mejoras, el costo asociado al cálculo y manipulación de la matriz hessiana continúa siendo una limitación importante en problemas de gran escala. Esta situación motivó el desarrollo de los métodos quasi-Newton, los cuales buscan aproximar la información de curvatura sin calcular explícitamente el Hessiano.

2.5. Métodos quasi-Newton

Con el fin de reducir las limitaciones computacionales asociadas al método de Newton, particularmente el cálculo e inversión explícita de la matriz hessiana, se desarrollaron los métodos quasi-Newton. Estos métodos conservan la estructura general del método de Newton, pero reemplazan la matriz hessiana por una aproximación construida iterativamente a partir de información del gradiente [5, p. 304].

De manera general, la iteración de un método quasi-Newton puede expresarse como:

$$\begin{cases} \mathbf{x}^{(0)} \\ \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - B_k \nabla f(\mathbf{x}^{(k)}) \end{cases} \tag{5}$$

donde B_k representa una aproximación de la inversa de la matriz hessiana $H_f(\mathbf{x}^{(k)})^{-1}$.

A diferencia del método de Newton clásico, los métodos quasi-Newton evitan el cálculo explícito de derivadas de segundo orden, reduciendo significativamente el costo computacional y mejorando su aplicabilidad en problemas de gran escala [6, p. 136].

2.5.1. Método BFGS

Uno de los métodos quasi-Newton más utilizados es BFGS (Broyden-Fletcher-Goldfarb-Shanno), el cual construye una aproximación iterativa de la inversa de la matriz hessiana utilizando diferencias entre gradientes consecutivos [6, pp. 136–143].

Para construir la aproximación B_k, se definen los vectores:

$$\mathbf{s}_k = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)},$$

y

$$\mathbf{y}_k = \nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)}).$$

A partir de estos vectores, BFGS actualiza iterativamente la aproximación de la inversa de la matriz hessiana incorporando información de curvatura obtenida a partir de diferencias entre gradientes consecutivos [6, pp. 136–143]. De esta manera, el método aproxima el comportamiento de Newton sin requerir el cálculo explícito de derivadas de segundo orden.

No obstante, BFGS requiere almacenar y manipular matrices densas de tamaño $n \times n$, por lo que tanto su complejidad temporal como espacial son del orden de $O(n^2)$. En consecuencia, el método puede resultar impráctico en problemas de muy alta dimensión.

2.5.2. Método L-BFGS

El método L-BFGS (Limited-memory BFGS) corresponde a una variante de BFGS diseñada para problemas de optimización de gran escala [6, pp. 176–185]. Con el fin de reducir los requerimientos computacionales de BFGS, L-BFGS evita construir y almacenar aproximaciones completas de la matriz hessiana inversa.

En su lugar, el método utiliza únicamente un número reducido de pares de vectores $(\mathbf{s}_k, \mathbf{y}_k)$ correspondientes a iteraciones recientes, definidos previamente en la sección 2.5.1. A partir de estos vectores, L-BFGS reconstruye implícitamente una aproximación de la acción de la matriz B_k sobre el gradiente, sin necesidad de calcular o almacenar explícitamente dicha matriz.

Si se utilizan únicamente m pares recientes de vectores, donde típicamente $m \ll n$, tanto la complejidad temporal como espacial del método se reducen a $O(mn)$, lo que convierte a L-BFGS en una alternativa eficiente para problemas de gran escala. Debido a estas propiedades, el método ha sido ampliamente utilizado en optimización numérica moderna y en aplicaciones de aprendizaje automático e inteligencia artificial.

3. Metodología

Con el fin de analizar el comportamiento de métodos basados en Newton en problemas de optimización, se desarrolló un caso de estudio dividido en dos etapas.

En una primera etapa, se aplica manualmente el método de Newton para optimización sobre una función de baja dimensión, con el objetivo de ilustrar explícitamente el cálculo del gradiente, la matriz hessiana y las iteraciones del método.

Posteriormente, se considera un problema de optimización de mayor dimensión correspondiente al entrenamiento de una red neuronal pequeña para aproximar una función matemática diferenciable. En este contexto, el entrenamiento se formula como un problema de minimización de una función de pérdida de error cuadrático medio (MSE), utilizando el método L-BFGS como algoritmo de optimización.

Para ambos casos se analizará el comportamiento iterativo de los métodos, considerando aspectos como convergencia, estabilidad y evolución del error.

4. Casos de estudio y resultados

4.1. Aplicación manual del método de Newton a la función de Rosenbrock

Como ejemplo ilustrativo se considera la función de Rosenbrock [7]:

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2.$$

Esta función es ampliamente utilizada como problema de prueba en optimización numérica, ya que presenta un valle estrecho que puede dificultar la convergencia de métodos iterativos. Esto se ilustra en la Fig. 1, donde el punto rojo marca el mínimo global de la función en $(1, 1)$.

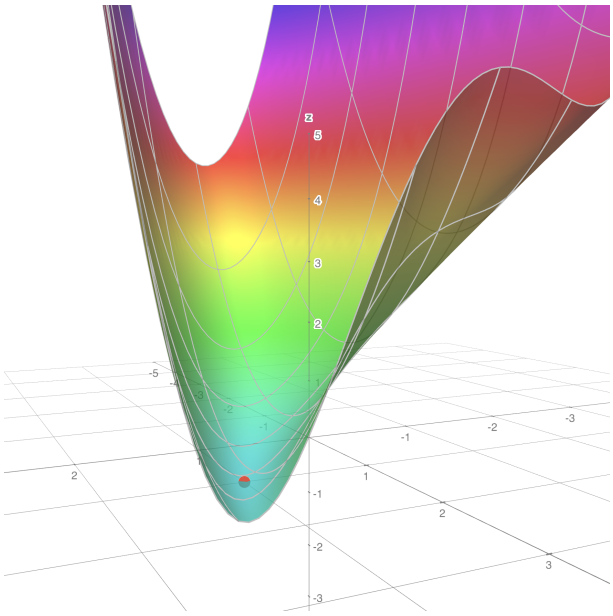

Figura 1: Visualización 3D de la función de Rosenbrock. Fuente: Math3D (math3d.org).

El gradiente de la función f está dado por:

$$\nabla f(x, y) = \begin{bmatrix} 2(x-1) - 400x(y - x^2) \\ 200(y - x^2) \end{bmatrix}$$

y la matriz hessiana correspondiente es:

$$H_f(x, y) = \begin{bmatrix} 1200x^2 - 400y + 2 & -400x \\ -400x & 200 \end{bmatrix}.$$

Por tanto, la iteración de Newton para optimización según la ecuación 4 es:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - H_f(\mathbf{x}^{(k)})^{-1} \nabla f(\mathbf{x}^{(k)}).$$

Tomando como punto inicial:

$$\mathbf{x}^{(0)} = \begin{bmatrix} -1.2 \\ 1 \end{bmatrix},$$

se obtienen las iteraciones mostradas en el Cuadro I. En cada iteración se calcula la norma del gradiente $\|\nabla f(\mathbf{x}^{(k)})\|$, la cual constituye un criterio natural de convergencia para el método. Conforme esta cantidad se aproxima a cero, el gradiente se acerca al vector nulo y, por tanto, la solución iterativa se aproxima a un punto crítico de la función.

Cuadro I
Iteraciones del método de Newton aplicado a la función de Rosenbrock.

k	$x^{(k)}$	$y^{(k)}$	$f(\mathbf{x}^{(k)})$	$\|\nabla f(\mathbf{x}^{(k)})\|$
0	-1.2000	1.0000	24.2000	232.8677
1	-1.1753	1.3807	4.7319	4.6394
2	0.7631	-3.1750	1411.8452	1370.7898
3	0.7634	0.5828	0.0560	0.4731
4	1.0000	0.9440	0.3132	25.0274
5	1.0000	1.0000	1.85×10^{-11}	8.61×10^{-6}

Podemos ver que el método efectivamente converge hacia el mínimo global de la función, ubicado en:

$$(x^*, y^*) = (1, 1),$$

donde se cumple que:

$$\nabla f(1, 1) = \mathbf{0}.$$

Este ejemplo permite observar explícitamente el papel del gradiente y de la matriz hessiana dentro de la iteración de Newton. Asimismo, muestra que la trayectoria seguida por el método depende de la geometría local de la función y que la convergencia no necesariamente ocurre de manera monótona. La Fig. 2 muestra la trayectoria generada por el método sobre la función de Rosenbrock.

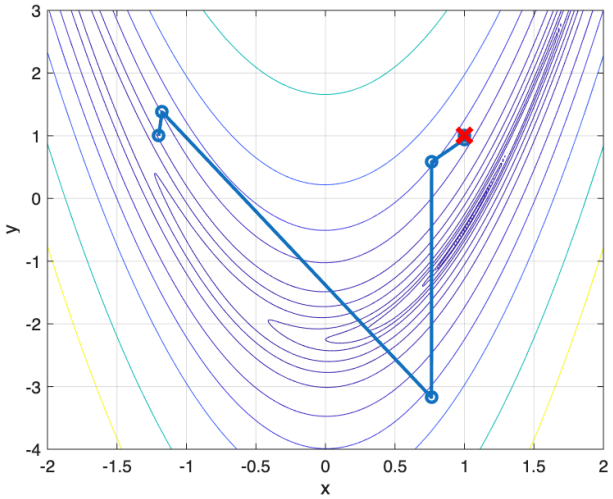

Figura 2: Trayectoria generada por el método de Newton sobre la función de Rosenbrock.

Aunque la función de Rosenbrock involucra únicamente dos variables, el ejemplo ilustra los conceptos fundamentales desarrollados en el marco teórico. En la siguiente sección se estudiará un problema de mayor escala, donde estos mismos principios se aplican al entrenamiento de una red neuronal mediante el método L-BFGS.

4.2. Entrenamiento de una red neuronal mediante L-BFGS

Como segundo caso de estudio se considera el entrenamiento de una red neuronal pequeña para aproximar la función seno en el intervalo $[0, 2\pi]$. A diferencia del caso anterior, donde el problema solo involucra dos variables, en este caso las variables de optimización corresponden a todos los pesos y sesgos de la red neuronal. Por lo tanto, el problema se formula en un espacio de mayor dimensión.

Para generar los datos de entrenamiento se utilizaron 200 puntos igualmente espaciados en el intervalo $[0, 2\pi]$, con valores objetivo dados por:

$$y_i = \sin(x_i).$$

La red neuronal utilizada fue implementada en PyTorch y está compuesta por una capa de entrada, dos capas ocultas de 32 neuronas con función de activación tangente hiperbólica y una capa de salida lineal. Con esta arquitectura, el conjunto de parámetros del modelo se denota por θ.

El entrenamiento se formuló como la minimización del error cuadrático medio:

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \Big(\sin(x_i) - \hat{f}(x_i; \theta) \Big)^2 ,$$

donde $\hat{f}(x_i; \theta)$ representa la salida generada por la red neuronal para la entrada x_i.

Para minimizar esta función de pérdida se utilizó el optimizador L-BFGS disponible en PyTorch. En la implementación se empleó una tasa de aprendizaje de 0.8, búsqueda lineal Strong Wolfe, un máximo de 20 iteraciones internas por época y un total de 50 épocas de entrenamiento. Durante el proceso se registraron dos métricas principales: el valor de la función de pérdida MSE y la norma del gradiente de la función de pérdida con respecto a los parámetros del modelo.

El Cuadro II resume las métricas principales obtenidas durante el entrenamiento.

Cuadro II
Resumen del entrenamiento de la red neuronal mediante L-BFGS.

Métrica	Valor
MSE inicial	0.570946
MSE final	0.000002
Norma final del gradiente	0.000494
Épocas	50

Con el fin de observar el comportamiento iterativo del método, el Cuadro III muestra algunas épocas representativas del entrenamiento. Se observa que la función de pérdida disminuye rápidamente durante las primeras iteraciones y luego se estabiliza en un valor cercano a cero.

Cuadro III
Épocas representativas del entrenamiento mediante L-BFGS.

Época	MSE	$\|\nabla\mathcal{L}\|$
1	0.570946	0.103909
2	0.072474	0.267207
3	0.027224	0.083819
4	0.004875	0.056462
5	0.000707	0.019766
10	0.000009	0.004003
41	0.000002	0.000494
50	0.000002	0.000494

La Fig. 3 compara la función seno original con la aproximación obtenida por la red neuronal después del entrenamiento.

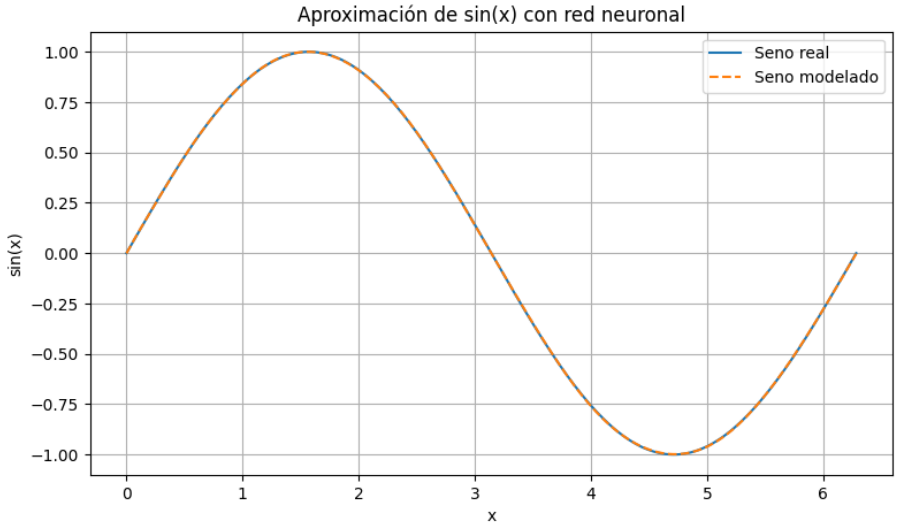

Figura 3: Comparación entre la función $\sin(x)$ y la aproximación generada por la red neuronal entrenada mediante L-BFGS.

La Fig. 4 muestra la evolución de la función de pérdida MSE a lo largo del entrenamiento.

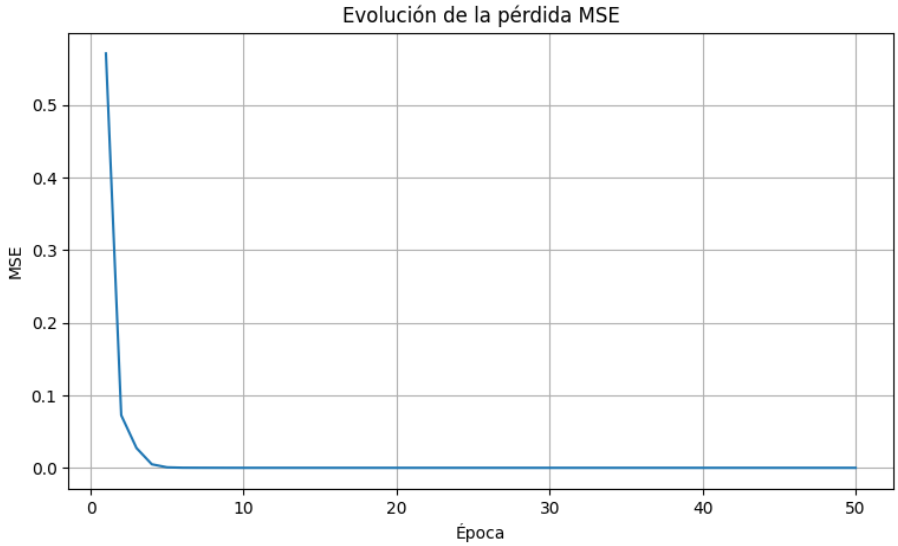

Figura 4: Evolución de la función de pérdida MSE durante el entrenamiento.

Finalmente, la Fig. 5 presenta la evolución de la norma del gradiente. Esta métrica permite relacionar el experimento con la formulación teórica de la optimización como búsqueda de raíces del gradiente, pues valores pequeños de $\|\nabla\mathcal{L}\|$ indican que el proceso se aproxima a un punto crítico de la función de pérdida.

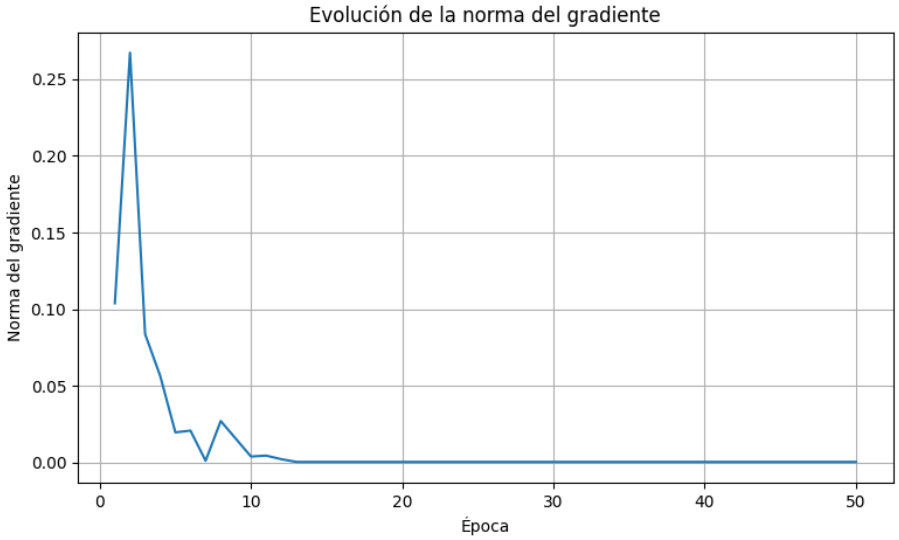

Figura 5: Evolución de la norma del gradiente durante el entrenamiento mediante L-BFGS.

5. Análisis de resultados

El primer caso de estudio permitió observar de forma explícita el funcionamiento del método de Newton para optimización en un problema de baja dimensión. A partir de la función de Rosenbrock se evidenció cómo el método utiliza simultáneamente el gradiente y la matriz hessiana para determinar la dirección de búsqueda en cada iteración. Asimismo, se observó que la convergencia no necesariamente es monótona, pues durante las primeras iteraciones el valor de la función puede aumentar antes de aproximarse al mínimo global.

En el segundo caso de estudio se trasladaron estos mismos principios al entrenamiento de una red neuronal. En este contexto, el objetivo no consiste en optimizar directamente la función $\sin(x)$, sino en minimizar una función de pérdida definida sobre los parámetros de la red. Por esta razón, el problema de optimización ocurre en el espacio de pesos y sesgos del modelo, lo que aumenta la dimensionalidad del problema en comparación con el caso manual.

Los resultados obtenidos muestran una disminución considerable de la función de pérdida. En particular, el MSE pasó de 0.570946 en la primera época a aproximadamente 0.000002 al finalizar el entrenamiento. Esto indica que la red neuronal logró aproximar adecuadamente la función seno en el intervalo considerado. De manera similar, la norma del gradiente se redujo hasta un valor final de 0.000494, lo cual sugiere que el método se acercó a un punto crítico de la función de pérdida.

También se observa que la reducción de la pérdida ocurre principalmente durante las primeras épocas. Después de alcanzar valores muy pequeños, las métricas tienden a estabilizarse, lo cual es consistente con el comportamiento esperado de un método de optimización iterativo cuando se aproxima a una solución.

En conjunto, ambos casos de estudio permiten conectar el método de Newton clásico con los métodos quasi-Newton. Mientras el ejemplo de Rosenbrock muestra explícitamente el papel del gradiente y del Hessiano, el caso de la red neuronal ilustra cómo estas ideas pueden extenderse a problemas de mayor dimensión mediante L-BFGS, evitando el cálculo explícito de la matriz hessiana.

6. Conclusiones

El método de Newton para optimización puede interpretarse como una aplicación del método de Newton para raíces, donde el objetivo consiste en encontrar puntos en los que el gradiente de

la función se anula.

El caso de la función de Rosenbrock permitió visualizar el papel del gradiente y del Hessiano en un problema de baja dimensión, así como la influencia de la geometría local de la función sobre la trayectoria de las iteraciones.

A pesar de la rápida convergencia que puede presentar el método de Newton, su dependencia del Hessiano limita su aplicación directa en problemas de mayor escala, especialmente cuando el número de variables es elevado.

El método L-BFGS constituye una alternativa práctica para problemas de optimización de mayor dimensión, ya que utiliza aproximaciones de la información de curvatura sin calcular explícitamente la matriz hessiana.

El entrenamiento de una red neuronal para aproximar la función seno permitió evidenciar una aplicación moderna de estos métodos en aprendizaje automático, mostrando una disminución significativa de la pérdida y de la norma del gradiente durante el proceso de optimización.

7. Trabajo futuro

Como trabajo futuro podrían considerarse redes neuronales de mayor complejidad, así como funciones objetivo más difíciles de aproximar. También sería relevante comparar experimentalmente el comportamiento de L-BFGS con otros optimizadores ampliamente utilizados en aprendizaje automático, como Adam o descenso por gradiente estocástico.

Además, podría extenderse el análisis a problemas de clasificación y a conjuntos de datos reales, con el fin de estudiar el comportamiento de los métodos quasi-Newton en escenarios más cercanos a aplicaciones prácticas.

8. Material complementario

Con el fin de facilitar la reproducción de los resultados presentados en este trabajo, se pone a disposición el código fuente desarrollado para ambos casos de estudio.

El repositorio del proyecto contiene la implementación del método de Newton aplicado a la función de Rosenbrock en MATLAB, así como el notebook utilizado para el entrenamiento de la red neuronal mediante el optimizador L-BFGS en PyTorch. Dicho repositorio puede consultarse en:

`https://github.com/rodamaj/newton-quasi-newton-case-studies`

Referencias

[1] R. L. Burden y J. D. Faires, *Numerical Analysis*, 9.ª ed. Brooks/Cole, Cengage Learning, 2011.

[2] T. M. Apostol, *Calculus II*. Reverté, 1984.

[3] F. S. Hillier y G. J. Lieberman, *Introducción a la Investigación de Operaciones*, 9.ª ed. McGraw-Hill, 2010.

[4] S. J. Russell y P. Norvig, *Artificial Intelligence: A Modern Approach*, 4.ª ed. Pearson, 2021, Global Edition.

[5] S. C. Chapra y R. P. Canale, *Métodos numéricos para ingenieros*, 7.ª ed. McGraw-Hill, 2014.

[6] J. Nocedal y S. J. Wright, *Numerical Optimization*, 2.ª ed. Springer, 2006.

[7] H. H. Rosenbrock, «An Automatic Method for Finding the Greatest or Least Value of a Function,» *The Computer Journal*, vol. 3, n.º 3, págs. 175-184, 1960. DOI: 10.1093/comjnl/3.3.175.