Estudio Comparativo de Métodos de Optimización: Gradiente Descendente, Regresión Lineal y Regularización

Belén Götz e Ion Harteker

Universidad de San Andrés, Buenos Aires, Argentina

2do Semestre 2024

Resumen

Este informe aborda el análisis de métodos numéricos aplicados a la optimización y la regresión, con un enfoque en el descenso por gradiente y técnicas complementarias. En primer lugar, se optimiza la función de Rosenbrock utilizando tanto el descenso por gradiente como el método de Newton, analizando las tasas de convergencia, el impacto de las tasas de aprendizaje y la sensibilidad frente a las condiciones iniciales. Luego, se aplica el descenso por gradiente a un problema de regresión lineal utilizando el dataset 'California Housing', comparando los resultados obtenidos mediante pseudoinversa y regresión de Ridge.

1. Introducción

El método de descenso por gradiente, o 'gradient descent' en inglés, es una de las herramientas numéricas fundamentales en la disciplina del Machine Learning. Este algoritmo se utiliza para minimizar de manera iterativa una función diferenciable, avanzando siempre en la dirección de mayor decrecimiento[1].

Aunque el descenso por gradiente se aplica en diversas áreas, como el comercio, la estadística y la computación experimental, su importancia en esta investigación radica en su rol fundamental dentro del campo del Machine Learning. Este método se utiliza ampliamente para minimizar la función de costo en modelos de inteligencia artificial, permitiendo evaluar el desempeño ('*performance*') de un modelo y ajustar parámetros, pesos o variables en consecuencia. Un ejemplo destacado de su aplicación en este campo es el trabajo *"Learning representations by back-propagating errors" (1986)*[2].

En este análisis, el descenso por gradiente se evalúa utilizando la función de Rosenbrock, una función conocida por su desafío en problemas de optimización. Esta función no convexa, caracterizada por un valle angosto y curvado, es ideal para probar y comparar diferentes algoritmos de optimización debido a la complejidad de su topología[3].

La segunda parte del experimento se centra en el método de regresión lineal, una herramienta estadística empleada para analizar la relación entre una variable dependiente Y, por ejemplo

el precio de una casa en California, y una o más variables independientes X, como la ubicación, representada en un formato numérico apto para el análisis. El objetivo del análisis es predecir con precisión valores desconocidos para Y, utilizando datos nuevos. Esto se logra determinando la línea (o el hiperplano en dimensiones superiores) que mejor se ajusta a los datos de entrenamiento. En esta investigación, el ajuste se evalúa minimizando una métrica de error, específicamente el error cuadrático medio (ECM), sobre el conjunto de datos de prueba.

La regresión lineal, ampliamente utilizada y vinculada a los inicios del Machine Learning y la inteligencia artificial, es un modelo predictivo básico que permite identificar relaciones lineales entre variables. Además, sirve como base para métodos más avanzados y complejos que hoy en día son utilizados en la industria. Un ejemplo práctico de su aplicación se encuentra en *"Predicting Students' Academic Performance Using Regression Analysis"*[4], donde se analiza el rendimiento académico de los estudiantes a partir de múltiples variables.

Un aspecto crucial de la regresión lineal es su flexibilidad matemática. Dado que su objetivo es encontrar un hiperplano con el menor error posible, el enfoque no se limita a cómo se obtiene dicho hiperplano, sino a garantizar que minimice el error. Por esta razón, en esta investigación se exploran tres métodos para construir el modelo de regresión: mediante gradiente descendente, pseudoinversa y regularización L_2. Finalmente, estos métodos se comparan para determinar cuál resulta más efectivo.

2. Marco Teórico

2.1. Descenso por Gradiente

El método de descenso por gradiente es una técnica utilizada para optimizar funciones, es decir, encontrar su mínimo. En el contexto de esta investigación, su aplicación se centra en la función de Rosenbrock. El descenso por gradiente es, además, un método muy popular para minimizar funciones de costo.

La expresión matemática que describe el descenso por gradiente es la siguiente:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \nabla f(\mathbf{x}_k) \tag{1}$$

En esta expresión:

- $\mathbf{x}_k$ representa el vector de parámetros en la k-ésima iteración.
- η es la tasa de aprendizaje (o tamaño de paso), que determina cuánto se avanza en la dirección del gradiente.
- $\nabla f(\mathbf{x}_k)$ es el gradiente de $f(\mathbf{x})$ evaluado en $\mathbf{x}_k$. Este gradiente es un vector que apunta en la dirección de máxima pendiente de f. Para una función $f(\mathbf{x})$ con n variables, se define como $\nabla f(\mathbf{x}_k) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right]^T$, evaluado en $\mathbf{x}_k$[5].

El método de descenso por gradiente requiere ciertas condiciones para ser aplicado de manera efectiva. En primer lugar, la función a optimizar debe ser diferenciable, ya sea de forma analítica o computacional, para que el gradiente pueda calcularse según la ecuación 1. Además, es crucial elegir una tasa de aprendizaje adecuada: lo suficientemente grande para asegurar una convergencia rápida, pero no tan grande que cause la omisión del mínimo durante alguna iteración[5]. Por último, es fundamental considerar el

contexto del problema; Si la función tiene múltiples mínimos locales o es concava, el algoritmo podría no alcanzar la solución global deseada.

Existen varios criterios de tolerancia para detener las iteraciones del método de descenso por gradiente:

- Cuando el gradiente cumple $\|\nabla f(\mathbf{x}_k)\| < \epsilon$, donde ϵ es un número positivo pequeño, indicando que el algoritmo está cerca de un punto estacionario.
- Cuando el cambio en el valor de la función es pequeño, $|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| < \delta$, lo que sugiere que el valor de la función ya no está cambiando significativamente.
- Cuando el número de iteraciones alcanza un límite predefinido, $k_{\max}$, para evitar bucles infinitos si no se ha alcanzado la convergencia[6].

En este experimento, se utiliza el criterio de freno $\|\nabla f(\mathbf{x}_k)\| < \epsilon$ para identificar puntos cercanos a un estacionario. Además, se establece un límite máximo de iteraciones para evitar ciclos infinitos en caso de que la convergencia no ocurra.

2.1.1. Tasa de Aprendizaje sobre Mesetas

El método de gradiente descendente, debido a su flexibilidad matemática, permite implementar variaciones que se adaptan a problemas específicos. Una de estas modificaciones es el ajuste dinámico de la tasa de aprendizaje en presencia de mesetas. Este enfoque consiste en reducir la tasa de aprendizaje a una fracción de su valor original cuando el valor de la función objetivo se incrementa respecto al paso anterior (si el valor de la función en el paso $k+1$ supera el valor en el paso k).

En el algoritmo de gradiente descendente con ajuste de la tasa de aprendizaje en mesetas, la tasa de aprendizaje se modifica según la siguiente fórmula:

$$\eta_{k+1} = \eta_k \times \beta$$

donde $\beta < 1$ representa el factor de reducción, y en este caso, se utiliza $\beta = 0.5$.

La condición para aplicar la reducción de la tasa de aprendizaje se expresa de la siguiente manera:

$$f(x_{k+1}) > f(x_k) \quad \Rightarrow \quad \eta = \eta \times 0.5 \qquad (2)$$

De esta manera, el método garantiza estabilidad en regiones donde el gradiente genera actualizaciones grandes, como ocurre en valles estrechos o cerca de mesetas. Esta adaptación es particularmente útil para funciones como la de Rosenbrock, cuyo paisaje presenta retos significativos debido a la combinación de regiones empinadas y planas.

2.2. Función de Rosenbrock

La función de Rosenbrock, mencionada previamente en 1 y 2, es una función no convexa que se utiliza frecuentemente como desafío de prueba para comparar métodos de optimización. Esta función presenta un valle angosto y curvado, lo que la hace ideal para evaluar algoritmos de optimización. La función se define como:

$$f(x, y) = (a - x)^2 + b(y - x^2)^2 \qquad (3)$$

Para este experimento, se consideran las constantes $a = 1$ y $b = 100$.

2.3. Método de Newton-Raphson con la matriz Hessiana

El método de Newton-Raphson es un algoritmo iterativo diseñado para encontrar raíces de funciones. En el contexto de problemas de optimización, se aplica para localizar puntos críticos

de una función $f(x)$, donde el gradiente $\nabla f(x)$ se anula.

En problemas de varias dimensiones, el método utiliza tanto el gradiente como la matriz Hessiana de la función objetivo. La matriz Hessiana, denotada como $H(x)$, es una matriz de derivadas parciales de segundo orden que describe la curvatura de la función.

A partir de un punto inicial $x_0 \in \mathbb{R}^n$, las iteraciones del método se definen como:

$$x_{k+1} = x_k - H(x_k)^{-1}\nabla f(x_k), \tag{4}$$

donde x_k representa el punto actual en la iteración k, $\nabla f(x_k)$ es el gradiente de $f(x)$ evaluado en x_k, $H(x_k)$ es la matriz Hessiana de $f(x)$ en x_k, y $H(x_k)^{-1}\nabla f(x_k)$ es el desplazamiento hacia el punto crítico.

La matriz Hessiana $H(x)$ se define como:

$$H(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}.$$

El método de Newton-Raphson tiene una tasa de convergencia cuadrática cerca de un mínimo local, siempre que la matriz Hessiana sea positiva definida en dicho punto. Esto asegura que el punto encontrado sea un mínimo y no un punto de silla o un máximo local [7].

Sin embargo, el método presenta ciertos desafíos. El cálculo de la matriz Hessiana y su inversión puede ser computacionalmente costoso, especialmente en problemas de alta dimensionalidad. Además, el condicionamiento de la matriz Hessiana es crucial: si esta no es positiva definida [8], el algoritmo puede no converger o incluso diverger. Asimismo, el método puede converger a puntos de silla, lo que implica que el resultado no sea el mínimo deseado.

A pesar de estas limitaciones, el método de Newton-Raphson es una herramienta poderosa para problemas donde la precisión y la rapidez de convergencia son críticas.

2.4. Error Cuadrático Medio y Regresión Lineal

La regresión lineal es un método estadístico que modela la relación entre una variable dependiente y (la que queremos predecir) y una o más variables independientes $x_1, x_2, \ldots, x_p$ (las variables predictoras). En su forma más simple, denominada regresión lineal simple, esta relación se expresa como:

$$y = \beta_0 + \beta_1 x + \varepsilon,$$

donde:

- β_0: intercepto o término independiente del modelo.
- β_1: coeficiente o pendiente que mide la influencia de x sobre y.
- ε: error o residuo que captura las diferencias entre los valores reales y los predichos por el modelo.

En la regresión lineal múltiple, que incluye varias variables independientes, el modelo se extiende a:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon.$$

El objetivo principal es encontrar los coeficientes $\beta_0, \beta_1, \ldots, \beta_p$ que minimicen el error y describan mejor los datos [9].

El **Error Cuadrático Medio** (ECM) es una métrica clave para evaluar la calidad de los modelos de regresión lineal. Se define como el promedio de los cuadrados de las diferencias entre los valores observados (y_i) y los predichos ($\hat{y}_i$):

$$\text{ECM} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

donde:

- y_i: valores reales de la variable dependiente.
- $\hat{y}_i$: valores predichos por el modelo, expresados como $\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}$.
- n: número de observaciones.

El ECM es también la función objetivo de la regresión lineal: el modelo busca minimizar este error ajustando los coeficientes $\boldsymbol{\beta}$ mediante el método de mínimos cuadrados[10]:

$$\min_{\boldsymbol{\beta}} \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\beta})^2,$$

donde $\mathbf{x}_i$ es el vector de variables independientes para cada observación i.

2.5. Solución por Pseudo-Inversa

2.5.1. Descomposición en Valores Singulares (SVD)[11]

La descomposición en valores singulares (SVD) es una técnica del álgebra lineal que factoriza una matriz A como:

$$A = U\Sigma V^{\top}, \tag{5}$$

donde:

- U: matriz ortogonal de autovectores de $AA^{\top}$.
- V: matriz ortogonal de autovectores de $A^{\top}A$.
- Σ: matriz diagonal con los valores singulares de A en la diagonal principal (también representada como S).

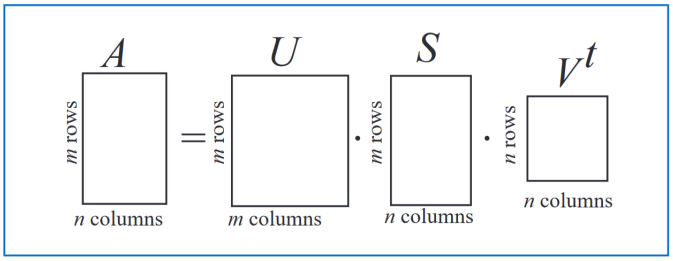

Figura 1: La descomposición en valores singulares (SVD Full).

La matriz Σ se construye a partir de los valores singulares s_i, que son las raíces cuadradas de los autovalores de $A^{\top}A$, organizados en orden decreciente. En la **descomposición SVD económica**, Σ tiene dimensiones reducidas y toma la forma:

$$\Sigma = \begin{bmatrix} s_1 & 0 & \cdots & 0 \\ 0 & s_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & s_n \end{bmatrix}$$

En la **SVD completa (full)**, la matriz Σ se extiende con filas o columnas adicionales de ceros para mantener las dimensiones originales de la matriz A, como se ilustra en la Figura 7.

Esta técnica se utiliza en diversas aplicaciones, como la compresión de datos, el ajuste por cuadrados mínimos y el análisis de datos [12].

2.5.2. Aplicación de SVD en Regresión Lineal

En la regresión lineal, la pseudo-inversa es una herramienta para resolver problemas de

mínimos cuadrados al encontrar el vector de coeficientes w que minimiza el ECM. La solución general es:

$$w = (X^\top X)^{-1} X^\top y, \tag{6}$$

donde:

- $X^\top X$: matriz de correlaciones entre las variables predictoras.
- $(X^\top X)^{-1}$: inversa ajustada para encontrar los coeficientes óptimos.
- $X^\top y$: proyección de los valores objetivo y en el espacio de características X.

Sin embargo, calcular directamente $(X^\top X)^{-1}$ puede ser numéricamente inestable si la matriz no es de rango completo. Para evitar este problema, se utiliza la decomposición por SVD [13]:

$$X = U \Sigma V^\top,$$

donde la pseudo-inversa de X se define como:

$$X^+ = V \Sigma^+ U^\top,$$

y Σ^+ se obtiene remplazando cada valor singular σ_i no nulo en Σ por su recíproco $1/\sigma_i$.

La solución para w con pseudo-inversa es entonces:

$$w = X^+ y = V \Sigma^+ U^\top y.$$

Esta formulación permite resolver el ECM de manera eficiente y estable, evitando problemas numéricos asociados al cálculo directo de la inversa, especialmente cuando X no es de rango completo.

2.6. Solución por Gradiente Descendiente

Para resolver el problema de regresión lineal, se pueden aplicar los conceptos descritos previamente en la sección de gradiente descendiente (2.1). El objetivo del método es aproximarse al mínimo de una función, por lo que al tomar el Error Cuadrático Medio (ECM) como la función objetivo a minimizar, el problema de regresión lineal se aborda de manera efectiva. Dado que el dataset utilizado no presenta una alta complejidad, es computacionalmente factible calcular el gradiente necesario para aplicar este método.

La formulación del problema se expresa como:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla \text{ECM}(\mathbf{w}_t),$$

donde:

- η: tasa de aprendizaje, que controla el tamaño de los pasos hacia el mínimo.
- $\nabla \text{ECM}(\mathbf{w}) = -\frac{2}{n} \mathbf{X}^\top (\mathbf{y} - \mathbf{X}\mathbf{w})$: gradiente del ECM con respecto a $\mathbf{w}$.

Este planteamiento permite encontrar iterativamente el vector de coeficientes que minimiza el ECM, garantizando una solución eficiente y adecuada para los datos disponibles.

2.7. Regularización L2 o Regresión de Ridge

La regularización L_2, también conocida como regresión de Ridge, es una técnica empleada para prevenir el sobreajuste en modelos de aprendizaje automático. Penaliza los coeficientes grandes del modelo añadiendo un término proporcional al cuadrado de su magnitud en la función de pérdida.

En la regresión de Ridge, la función objetivo modifica el ECM original añadiendo un término de regularización:

$$\text{Pérdida} = \text{ECM} + \lambda \sum_{j=1}^{p} \beta_j^2, \tag{7}$$

donde:

- $\text{ECM} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$: mide el ajuste del modelo a los datos.
- λ: hiperparámetro que controla la fuerza de la regularización.
- β_j: coeficientes del modelo.
- $\lambda \sum_{j=1}^{p} \beta_j^2$: término de penalización que reduce la magnitud de los coeficientes para evitar un modelo excesivamente complejo.

El efecto de esta regularización es evitar que el modelo asigne demasiada importancia a una sola característica, distribuyendo los pesos de manera más uniforme y reduciendo la complejidad. Esto es especialmente relevante en problemas en donde el modelo puede sobre-ajustarse a los datos de entrenamiento y no generalizar correctamente a datos nuevos[14].

En el contexto de esta investigación, la función objetivo con regularización L_2 se define como:

$$\text{ECM}_\lambda(\mathbf{w}) = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2,$$

donde:

- $\lambda > 0$: controla la penalización de coeficientes grandes.
- $\|\mathbf{w}\|_2^2 = \mathbf{w}^\top\mathbf{w}$: norma L_2 al cuadrado del vector de coeficientes.

La solución analítica para este problema regularizado se expresa como:

$$\mathbf{w}_\lambda = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y},$$

donde $\mathbf{I}$ es la matriz identidad. Este enfoque equilibra la precisión del ajuste del modelo con su capacidad de generalización, lo que lo convierte en una herramienta robusta frente al sobreajuste.

A partir del marco teórico presentado, se desarrollan a continuación dos experimentos numéricos: el primero enfocado en la aplicación del método de optimización de gradiente descendente, y el segundo en el uso de este mismo método para resolver un problema de regresión lineal.

3. Optimización en 2 dimensiones

3.1. Experimentos Numéricos

El primer experimento numérico se desarrolla sobre la función de **Rosenbrock** (ecuación (3)), visualizada en la Figura (2). Esta función es una prueba clásica en problemas de optimización debido a su naturaleza no convexa y al valle estrecho que conduce al mínimo global en $(1, 1)$ como muestra la Figura (3). El objetivo es minimizar esta función en dos dimensiones utilizando dos métodos de optimización: **Gradiente Descendente** (1) y el **Método de Newton** (4).

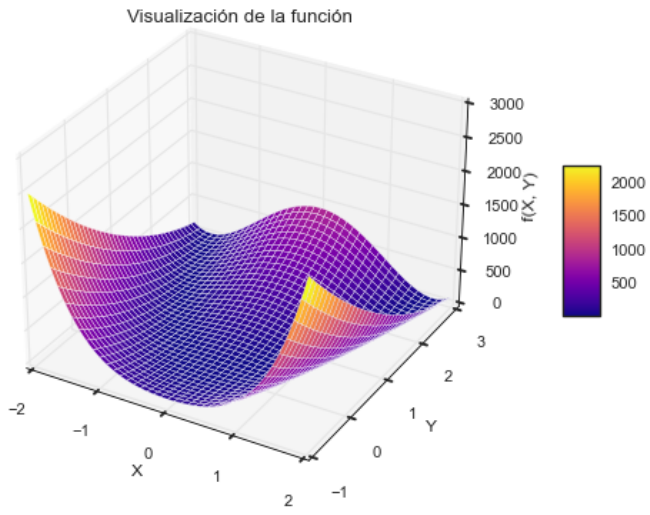

Figura 2: Función de Rosenbrock en 2 dimensiones con mapa de calor para los máximos (amarillos) y mínimos (violetas).

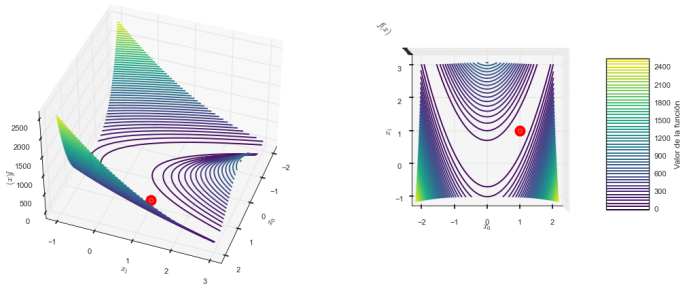

Figura 3: Visualización de la funcion de Rosenbrock en 3 dimensiones con su minimo global $(1, 1)$ marcado en Rojo.

El análisis explora el impacto de la elección de la tasa de aprendizaje (η) en el gradiente descendente, así como la sensibilidad a distintas condiciones iniciales. Adicionalmente, se comparan las trayectorias de convergencia y los errores generados por cada método.

3.1.1. Descenso por Gradiente

Para implementar el descenso por gradiente, se utiliza la función `gradient_descent()`, que toma los parámetros necesarios y aplica el método iterativo de minimización. Los parámetros requeridos son:

- `grad_f`: Gradiente de la función objetivo, representado por `grad_rosenbrock()`, calculado analíticamente.
- `learning_rate`: Tasa de aprendizaje (η), cuyo valor afecta la rapidez y estabilidad de la convergencia.
- `tol`: Tolerancia, usada como criterio de convergencia, si la norma del gradiente cae por debajo de este valor.
- `max_iter`: Número máximo de iteraciones, utilizado para evitar ciclos infinitos en caso de que no se alcance la tolerancia.

La implementación hace uso de la librería `NumPy`, en particular del módulo `linalg` para calcular la norma del gradiente. La función genera como salida la trayectoria de puntos visitados durante la optimización, lo que permite analizar cómo el algoritmo se aproxima al mínimo global.

Para visualizar los resultados, se emplea la función `plot_trajectory()`, que recibe como entrada `trajectory`, el arreglo que contiene los puntos (x, y) generados en cada iteración por `gradient_descent()`. Esta trayectoria se superpone a un gráfico de contornos de la función de Rosenbrock, mostrando cómo el algoritmo avanza paso a paso hacia el mínimo sobre las curvas de nivel.

3.1.2. Método de Newton

El Método de Newton-Raphson se implementa en la función `newton_method()`, que utiliza

tanto el gradiente como la matriz Hessiana de la función objetivo. La fórmula iterativa descrita en (4) ajusta los pasos hacia el mínimo mediante la inversión de la matriz Hessiana en cada iteración.

La matriz Hessiana se calcula analíticamente mediante la función `hessian_rosenbrock()`, que devuelve una matriz simétrica de 2×2 con las segundas derivadas parciales de la función de Rosenbrock. Este cálculo permite que el método de Newton haga uso de información de la curvatura, logrando una convergencia más rápida en la mayoría de los casos.

Aunque el método de Newton presenta ventajas en términos de velocidad de convergencia, tiene un costo computacional mayor debido a la necesidad de invertir la matriz Hessiana en cada iteración.

Se realizaron experimentos partiendo desde diferentes condiciones iniciales (x_0, y_0) para observar el impacto en las trayectorias generadas por cada método. Además, se probaron distintas tasas de aprendizaje (η) en el gradiente descendente, evaluando cómo estas afectan la rapidez y estabilidad de la convergencia.

Finalmente, se calcularon y compararon los errores entre las soluciones obtenidas por ambos métodos, tanto en términos de la distancia al mínimo global como en el número de iteraciones requeridas para alcanzar una tolerancia fija ($\|\nabla f(x, y)\| < 10^{-6}$).

3.2. Resultados

Primero se exploraron diferentes valores para η (tasas de aprendizaje). Los valores seleccionados para el experimento fueron: 0.0001, 0.001, 0.01 y 0.1. Se eligieron puntos aleatorios en el espacio para observar cuántas iteraciones necesita cada tasa de aprendizaje para converger.

Como se muestra en la Tabla 1, la tasa que converge más rápido es 0.001. Se puede observar que, con las tasas de 0.01 y 0.1, no se alcanza el mínimo en menos de 500,000 iteraciones.

Dado que no todas las tasas de aprendizaje logran converger, también se experimentó con el método de **gradiente descendente con ajuste de la tasa de aprendizaje en mesetas** (2). Como se observa en la Tabla 2, todos los valores finalmente convergen, pero las tasas de aprendizaje de 0.01 y 0.1 se ajustan durante el proceso. A pesar de converger, la tasa de aprendizaje de 0.1 tarda 70,659 iteraciones en hacerlo, lo que es más del doble de las 32,690 iteraciones necesarias con una tasa de 0.01. Con esta adaptación, la tasa óptima es 0.01, que luego se ajusta a 0.00125, acercándose a la tasa de 0.001. Por lo tanto, en general y para el método de gradiente descendente estándar, la tasa de aprendizaje óptima es **0.001**.

Lo mismo puede observarse en los gráficos de la Figura (4), donde se aprecia que con $\eta = 0.001$, el error para diferentes puntos iniciales (seleccionados aleatoriamente) disminuye progresivamente a medida que se realizan más iteraciones. En contraste, con $\eta = 0.0001$, el error permanece por encima de 10^0 y se estanca, mientras que con $\eta = 0.1$ el proceso diverge directamente.

Tasa de Aprendizaje	Iteraciones	Punto Final
0.0001	327000	(0.99999888, 0.99999776)
0.001	32690	(0.99999888, 0.99999776)
0.01	No se alcanzó la convergencia.	-
0.1	No se alcanzó la convergencia.	-

Tabla 1: Resultados de la optimización para distintas tasas de aprendizaje con máxima cantidad de iteraciones = 500000.

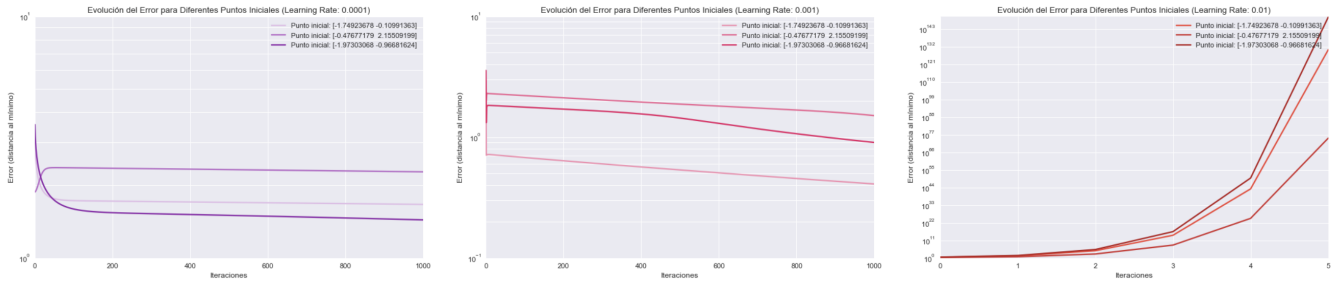

Figura 4: Convergencia del metodo de la gradiente descendente para los mismos punto de inicio pero utilizando distintas tasas de convergencia.

Tasa de Aprendizaje	Iteraciones	Punto Final	Tasa de aprendizaje final
0.0001	442311	(0.99999999 0.99999998)	0.00010
0.001	44219	(0.99999999 0.99999998)	0.00100
0.01	32690	(0.99999999 0.99999998)	0.00125
0.1	70659	(0.99999999 0.99999998)	0.00063

Tabla 2: Resultados de la optimización con gradiente descendente con ajuste de la tasa de aprendizaje en mesetas.

Dada la elección del η óptimo (0.001) se exploró el método dadas distintas condiciones iniciales. Primero, para poder visualizar el método de Gradiente Descendente se probó desde tres puntos iniciales diferentes. En las Figuras (5), (6) y (7) se ven los avances iterativos del método partiendo desde las condiciones iniciales (-1.2, 1), (0, 2.5) y (2, 2), respectivamente.

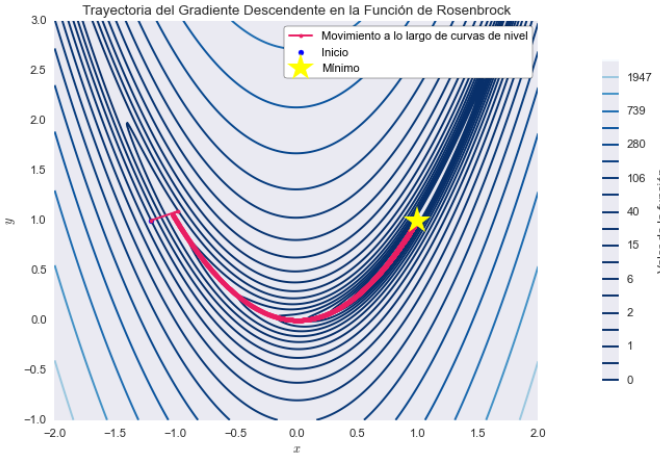

Figura 5: Trayectoria del Gradiente Descendente en la Función de Rosenbrock para (-1.2, 1) con tasa de aprendizaje = 0.001.

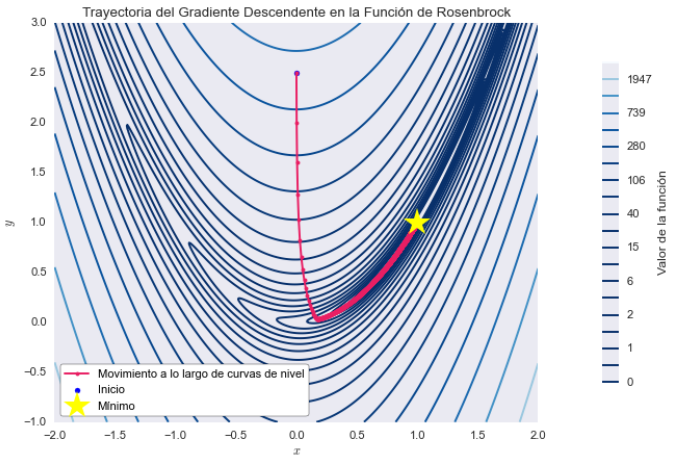

Figura 6: Trayectoria del Gradiente Descendente en la Función de Rosenbrock para (0, 2.5) con tasa de aprendizaje = 0.001.

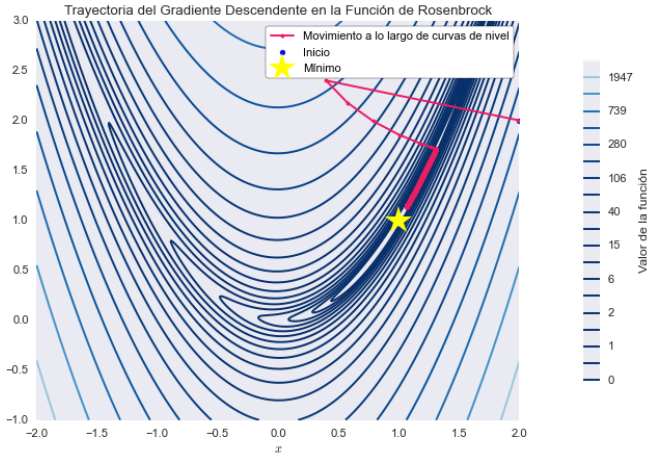

Figura 7: Trayectoria del Gradiente Descendente en la Función de Rosenbrock para (2, 2) con tasa de aprendizaje = 0.001.

Las condiciones iniciales tienen un impacto significativo en la rapidez con la que el método de gradiente descendente converge, como se observa en las figuras correspondientes. Cada pie de figura destaca cómo la ubicación inicial afecta tanto la cantidad de iteraciones necesarias para alcanzar el mínimo como el trayecto seguido en el espacio de búsqueda.

Otra observación importante es que el método de gradiente descendente se mueve ortogonalmente a las curvas de nivel de la función objetivo. Esto ocurre porque el gradiente de una función en un punto dado es perpendicular a las curvas de nivel en ese mismo punto. Al avanzar en la dirección opuesta al gradiente, el método busca reducir el valor de la función de manera más eficiente, moviéndose directamente hacia el mínimo más cercano.

Para analizar multiples recorridos simultáneamente, se seleccionaron un set de condiciones iniciales aleatorias y uno uniforme, y se visualizaron en los gráficos (8) y (9).

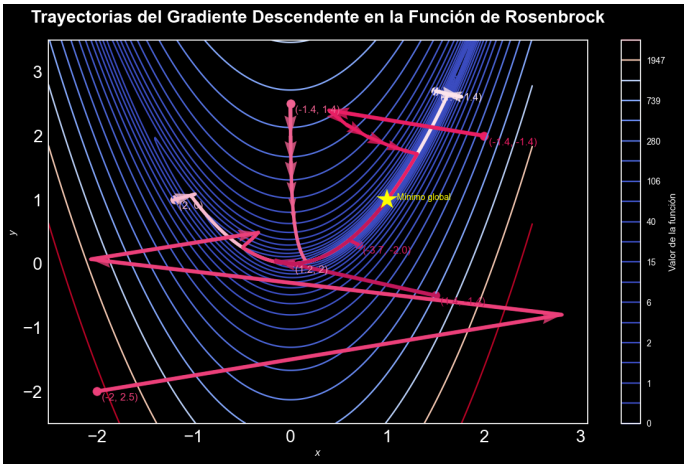

Figura 8: Aproximación del método de gradiente descendente hacia el mínimo global de la función utilizando varios puntos de inicio.

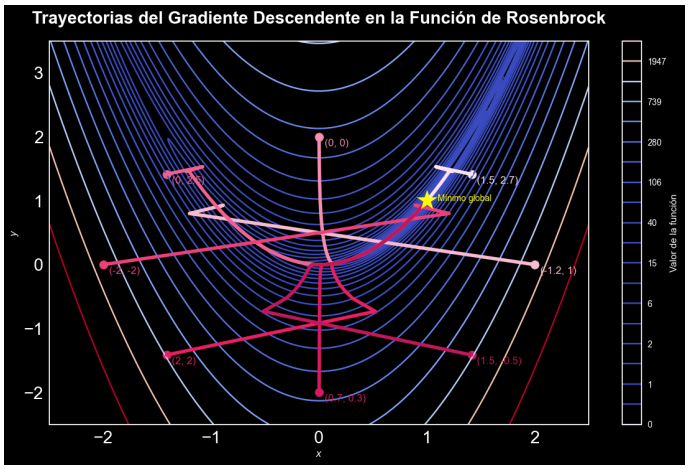

Figura 9: Evolución del metodo de la gradiente descendente hacia el minimo global tomando puntos simetricos respecto al eje x del dominio.

Con algunos puntos de partida, como $(-2, 2.5)$, los pasos del gradiente oscilan de forma más abrupta en comparación con otros puntos, como $(-1.4, 1.4)$. Sin embargo, a pesar de los recorridos diferentes, todos los puntos iniciales convergen hacia el mismo "valle", representado por las curvas de nivel más azules que rodean el mínimo global.

Cuando los puntos iniciales están distribuidos uniformemente, se observa la misma tendencia de convergencia incluso desde posiciones opuestas. Esto se debe a la simetría (aproximada) de la función original, que garantiza trayectorias similares hacia el mínimo.

En la Figura (10) se graficaron los mismos puntos iniciales que en la Figura (9). Se puede apreciar visualmente cómo, dada esta simetría, las trayectorias convergen de forma análoga, descendiendo hacia el mínimo global desde diferentes direcciones.

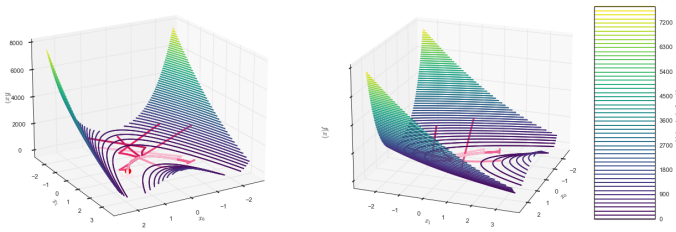

Figura 10: Gráfica tridimensional que expone la convergencia del metodo de gradiente descendente con los mismos puntos simetricos que en la Figura (9).

En ciertos experimentos, el método no logró alcanzar la convergencia dentro del límite de iteraciones máximo debido a inicializaciones lejanas al mínimo. Con las condiciones iniciales $(-12, 13)$ y $(5, 2)$ mostradas en la Figura (11), en lugar de acercarse al mínimo global $(1, 1)$, las trayectorias correspondientes a estos puntos terminan en posiciones como $(3.1617, 9.9997)$ y $(4.1860, 17.5262)$. Esto indica que no logran aproximarse a la solución en las 500000 iteraciones permitidas, evidenciando que la elección del punto de partida es crucial para la convergencia del método.

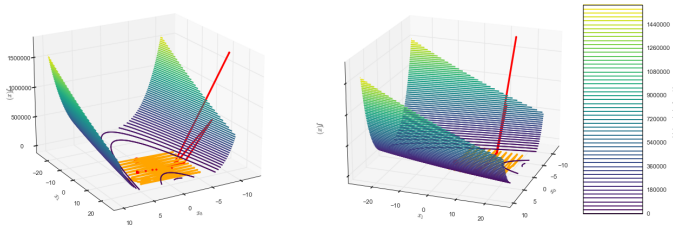

Figura 11: Gráfico de la función de Rosenbrock con puntos iniciales $(-12, 13)$ y $(5, 2)$ que no convergen al superar la maxima cantidad de iteraciones impuestas.

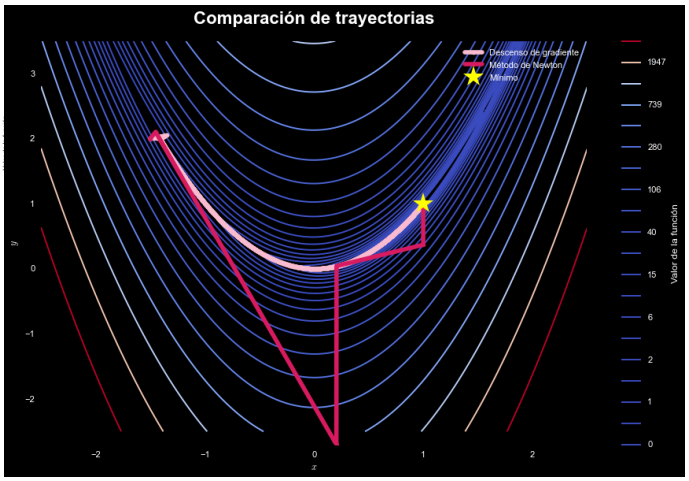

Figura 12: Comparación de trayectorias utilizando los métodos de Gradiente Descendente y Newton, ambos iniciando desde el punto $(-1.5, 2.0)$.

3.2.1. Comparación entre Gradiente Descendente y Método de Newton

El método de Newton mostró un desempeño significativamente superior en términos de convergencia: El mínimo global fue alcanzado en solo 8 iteraciones, como se muestra en la Figura (12). La precisión del método fue exacta, con un punto final de $(1.0, 1.0)$, igual al mínimo. En contraste, el gradiente descendente requirió más de 32000 iteraciones para alcanzar resultados similares, dependiendo de la tasa de aprendizaje y la condición inicial.

Para problemas como la minimización de Rosenbrock, el método de Newton es significativamente más rápido y preciso, pero requiere el cálculo y la inversa de la matriz Hessiana, lo cual puede ser computacionalmente costoso en dimensiones altas y ahí es que conviene utilizar el Gradiente Descendente, cuyo éxito depende de la elección de la tasa de aprendizaje y la proximidad al mínimo global.

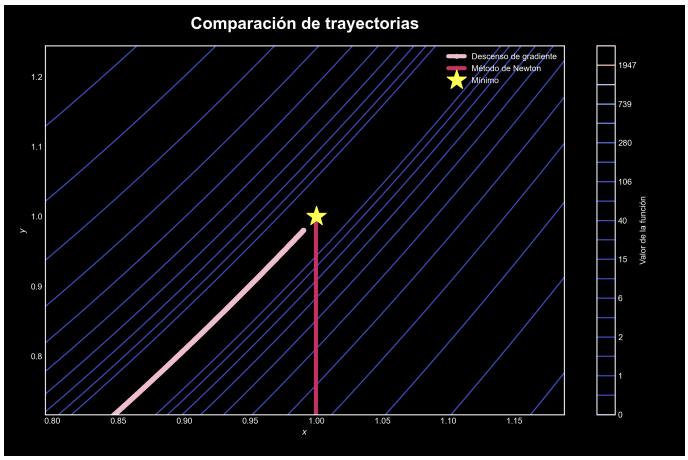

Figura 13: Comparación de las trayectorias y precisión de los métodos de Gradiente Descendente y Newton-Raphson, partiendo del punto inicial $(-1.5, 2.0)$ hacia el mínimo global $(1, 1)$ en un máximo de $10,000$ iteraciones.

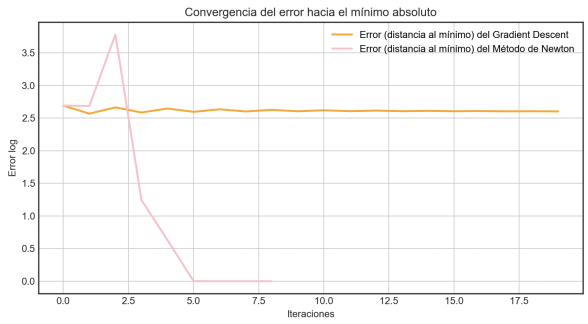

Figura 14: Convergencia del error hacia el mínimo absoluto usando el método de Gradiente Descendente y Newton-Raphson.

En la gráfica se observa que el error del método de Newton-Raphson disminuye rápidamente, estabilizándose cerca de cero antes de la octava iteración, alcanzando su valor mínimo después de aproximadamente cinco iteraciones. En contraste, el error del método de Gradiente Descendente no comienza a reducirse significativamente hasta después de varias iteraciones, lo que evidencia la lentitud con la que este método se acerca al mínimo absoluto, en comparación con Newton-Raphson.

4. Cuadrados Mínimos mediante Descenso por Gradiente

4.1. Métodos Numéricos

Esta sección aborda el problema de regresión lineal utilizando dos enfoques: la solución analítica mediante **pseudoinversa** y la solución iterativa a través del **gradiente descendente**. Además, se analiza la **regresión regularizada** mediante la técnica de Ridge Regression ((19)), que incorpora un término de penalización para mejorar la generalización.

4.1.1. Regresión Lineal: Método por Pseudoinversa

La regresión lineal por pseudoinversa se implementó utilizando varias funciones en Python para armar y evaluar el modelo. El procedimiento incluye los siguientes pasos:

(1) El preprocesamiento, que consiste en una estandarización de los datos, se aplica mediante la función `standardize_features(X_train, X_test)`. Esta función le resta la media a cada característica y la divide por su desviación estándar. Este paso asegura que todas las características estén en una escala similar, mejorando la estabilidad numérica y facilitando la convergencia.

(2) La aplicación de la pseudoinversa para hallar los pesos de los coeficientes, como se muestra en (6), se realiza con la función `computepseudoinverse(X, y)`. Aquí, X es la matriz de características y y es el vector objetivo. La librería `NumPy` es utilizada en esta implementación, particularmente `np.linalg.inv`, que calcula la inversa de la matriz.

(3) Finalmente, las funciones `predict(X, w)` y `compute_ECM(y_true, y_pred)` se utilizan para evaluar el rendimiento del modelo en los conjuntos de entrenamiento y prueba, obteniendo el Error Cuadrático Medio (ECM).

4.1.2. Regresión por Gradiente Descendente

La implementación del gradiente descendente difiere del método de pseudoinversa principalmente en su función iterativa `gradientdescent()`. Este algoritmo sigue la fórmula:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla \text{ECM}(\mathbf{w}_t),$$

donde η es la tasa de aprendizaje ajustada dinámicamente como $\eta = \frac{1}{\sigma_1^2}$, siendo σ_1 el valor singular más grande de X. Toma los parámetros:

- `tol`: Tolerancia, criterio de detención basado en la norma del gradiente.
- `max_iter`: Máximo número de iteraciones.
- `trajectory`: Registro de la trayectoria de los pesos $\mathbf{w}$ en cada iteración.
- `errors`: Historial del ECM en cada iteración.

Las funciones de estandarización, predicción y cálculo del error son reutilizadas del método de pseudoinversa. Se analizaron múltiples tasas de aprendizaje ($\frac{0.1}{\sigma_1^2}, \frac{0.01}{\sigma_1^2}, \frac{0.001}{\sigma_1^2}, \frac{0.0001}{\sigma_1^2}$) y su impacto en la convergencia.

Se trabaja con $\eta = \frac{1}{\sigma_1^2}$ para la tasa de aprendizaje principal dado que σ_1 es el valor singular más grande de X, y está asociado al autovector que corresponde a la dirección en la que el espacio de características está más "estirado". Este valor de η asegura que el gradiente descendente realice pasos más equilibrados a lo largo de las diferentes direcciones del espacio de parámetros, lo que facilita una convergencia más rápida y estable.

Si se elige una tasa de aprendizaje demasiado grande, los pasos pueden ser excesivos y causar oscilaciones o divergencia. Por el contrario, una tasa de aprendizaje demasiado pequeña puede hacer que el proceso de convergencia sea muy lento.

4.1.3. Solución por Regresión de Ridge

La regularización L_2 se implementa mediante la función `ridge_gradient_descent_with_tracking()`, que optimiza la función objetivo:

$$\text{ECM}_\lambda(\mathbf{w}) = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\|^2,$$

donde $\lambda > 0$ penaliza los coeficientes grandes, reduciendo el riesgo de sobreajuste. La tasa de aprendizaje η se ajusta dinámicamente como $\eta = \frac{1}{\sigma_1+\lambda}$, considerando tanto el valor singular más grande de X como el efecto de regularización.

Esta función devuelve los pesos $\mathbf{w}$ y el historial de pérdida en cada iteración, calculado como la suma del ECM y el término de regularización. El historial de pérdida se visualizó para analizar la convergencia y la evolución del error.

Se graficó la evolución del ECM en escala logarítmica para analizar la convergencia. Se compararon los errores finales entre los métodos analítico (pseudoinversa) y de gradiente descendente mediante gráficos de barras.

4.2. Resultados

4.2.1. Evolución del ECM con gradiente descendente

En la Figura (15) se muestra la evolución del error cuadrático medio (ECM) en el conjunto de entrenamiento para distintas tasas de aprendizaje. Resultados similares se observan para el conjunto de prueba, como se presenta en la Figura (16). Se destaca que (en ambos casos, conjunto de prueba y de entrenamiento) la tasa de aprendizaje con menor ECM es ($\eta = 1/\sigma^2$), la cual decae rápidamente y se estabiliza en 0. Con $\eta = 0.1/\sigma^2$ el error también cae a 0 antes de las mil iteraciones, pero con las otras tasas de aprendizaje, a pesar de que se reduce la pendiente (empieza a estabilizarse de a poco) no llegan a bajar hasta 0.

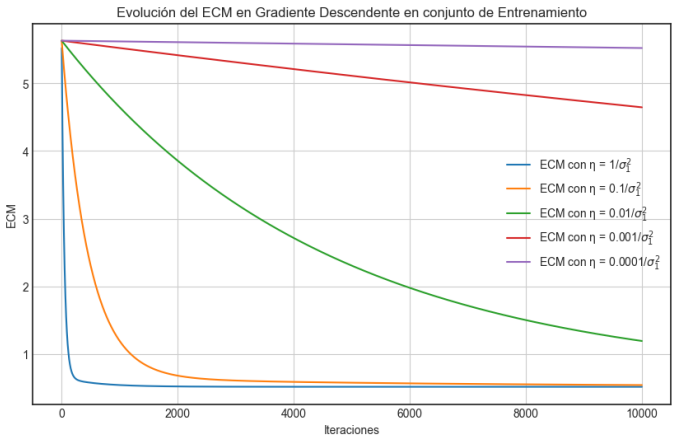

Figura 15: Evolución del ECM en el conjunto de entrenamiento para distintas tasas de aprendizaje.

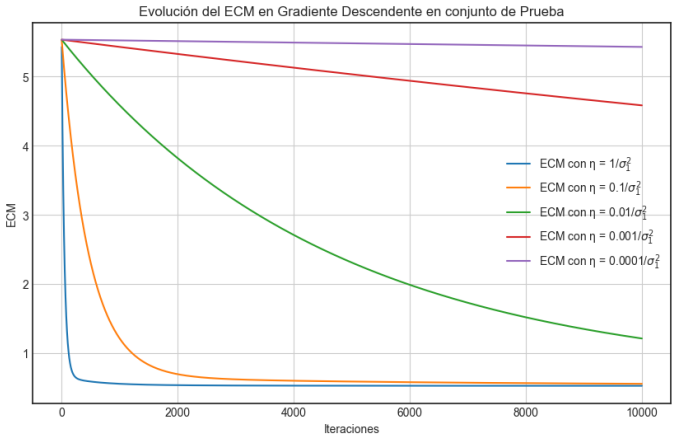

Figura 16: Evolución del ECM en el conjunto de prueba para distintas tasas de aprendizaje.

Luego se realizó el procedimiento analíticamente (mediante la pseudoinversa).

	Gradiente Descendente	**Pseudoinversa**
Error en entrenamiento	5.52127	0.51793
Error en prueba	5.42714	0.55589

Tabla 3: Errores en Entrenamiento y Prueba para Pseudoinversa y Gradiente Descendente.

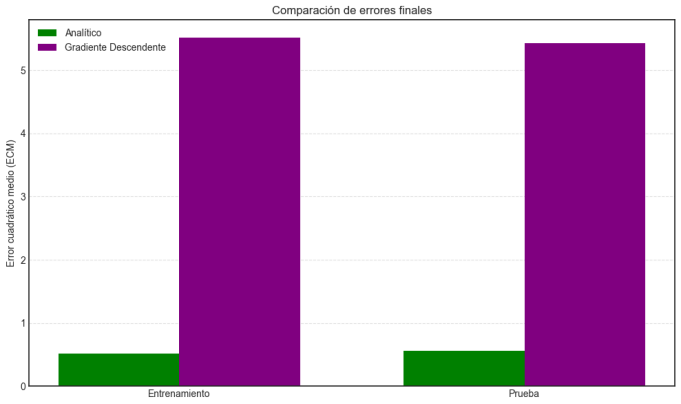

Figura 17: Comparación de la media de los errores finales para soluciones analíticas y mediante gradiente descendente.

El análisis de los resultados obtenidos permite observar diferencias significativas entre los métodos de pseudoinversa y gradiente descendente en términos de error cuadrático medio (ECM) en los conjuntos de entrenamiento y prueba. En el caso del conjunto de entrenamiento, el método de Pseudoinversa obtiene un error de 0.51793, considerablemente menor que el obtenido mediante gradiente descendente, que alcanza un valor de 5.52127. Este comportamiento era esperado, dado que la Pseudoinversa proporciona una solución exacta que minimiza el ECM en un solo paso analítico. Por otro lado, el Gradiente Descendente no logra una convergencia completa, lo cual podría deberse a una elección no óptima de la tasa de aprendizaje o a un número insuficiente de iteraciones.

En el conjunto de prueba, se observa un comportamiento similar. La pseudoinversa alcanza un error de 0.55589, mientras que el gradiente gescendente muestra un error significativamente mayor de 5.42714. Esto indica que el gradiente descendente no solo presenta limitaciones en la convergencia durante el entrenamiento, sino que además podría estar mostrando dificultades para generalizar adecuadamente como se ve en la

Figura (18).

Esto muestra como la pseudoinversa es superior en términos de precisión en este caso particular, logrando menores errores en ambos conjuntos. Sin embargo, el gradiente descendente puede ser una herramienta valiosa cuando se enfrentan restricciones computacionales o problemas de escalabilidad (pero depende de la elección se sus hiperparámetros, como la tasa de aprendizaje y el número de iteraciones).

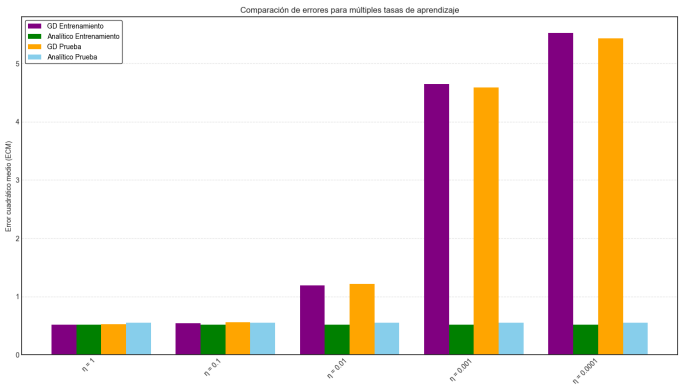

Figura 18: Comparación de la media de los errores finales para soluciones analíticas y mediante gradiente descendente para distintas tasas de aprendizaje.

4.3. Regularización L_2

La Figura (19) muestra cómo la regularización L_2 mejora la generalización al penalizar coeficientes grandes y reducir el sobreajuste. La elección del parámetro de regularización λ es clave para equilibrar la precisión y estabilidad del modelo.

Los errores cuadráticos medios (ECM) para diferentes valores de λ revelan su impacto en el rendimiento. Con $\lambda = 1e-06 \cdot \sigma_1$, el modelo tiene un buen ajuste, con un ECM de entrenamiento de 0.518 y un ECM de prueba de 0.556, lo que indica que la regularización es adecuada. Sin embargo, al aumentar λ, los errores en ambos conjuntos aumentan, sugiriendo que la regularización excesiva penaliza demasiado los coeficientes.

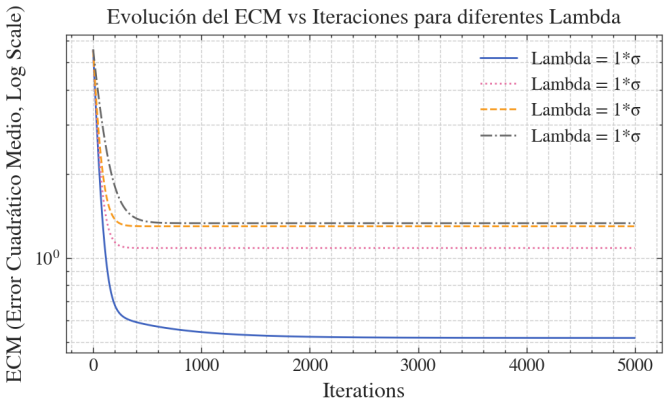

Figura 19: Evolución del ECM regularizado frente a iteraciones para distintos valores de λ.

Para $\lambda = 0.01 \cdot \sigma_1$, los ECM de entrenamiento y prueba suben a 0.926 y 0.916, respectivamente, y este aumento se vuelve más pronunciado a medida que λ se incrementa. Con $\lambda = 1 \cdot \sigma_1$, los errores alcanzan 1.329 y 1.303.

En resumen, es esencial elegir un λ que evite tanto el sobreajuste como la subregularización. En este caso, $\lambda = 1e-06 \cdot \sigma_1$ ofrece el mejor equilibrio proporcionando un buen rendimiento tanto en entrenamiento como en prueba.

5. Conclusiones

Este estudio se compararon diversos métodos numéricos de optimización y regresión, destacando diferencias clave en precisión, eficiencia y capacidad de generalización. El método de Newton-Raphson demostró una rápida convergencia hacia el mínimo, estabilizándose en pocas iteraciones, mientras que el gradiente descendente, aunque más lento, ofrece mayor flexibilidad para escenarios de alta dimensionalidad y problemas con restricciones computacionales pero

depende de la elección de η (tasa de aprendizaje) y de las condiciones iniciales.

En el caso de la regresión lineal, la solución analítica mediante pseudoinversa es precisa y eficiente, pero el gradiente descendente combinado con regularización L_2 (Ridge Regression) se destacó como una alternativa efectiva para mejorar la generalización y reducir el sobreajuste, especialmente en problemas de grandes dimensiones. Sin embargo, la efectividad de estos métodos depende críticamente de la selección adecuada de hiperparámetros como η y λ (regularización), subrayando la importancia de ajustar estos valores para garantizar estabilidad y rendimiento óptimo.

Referencias

[1] Crypto1 (user). *Gradient Descent Algorithm: How Does it Work in Machine Learning?* Accessed: 2024-23-11. 2024. URL: https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/.

[2] Hinton Rumelhart. *Learning representations by back-propagating errors.* Accessed: 2024-23-11. 1986. URL: https://www.nature.com/articles/323533a0.

[3] Wikipedia. *Rosenbrock function.* Accessed: 2024-23-11. -. URL: https://en.wikipedia.org/wiki/Rosenbrock_function.

[4] Rolly T. Dagdagui. *Predicting Students' Academic Performance Using Regression Analysis.* Accessed: 2024-11-29. 2017. URL: https://pubs.sciepub.com/education/10/11/2/education-10-11-2.pdf.

[5] Robert Kwiatkowski. *Gradient Descent Algorithm — a deep dive.* Accessed: 2024-23-11. 2021. URL: https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21.

[6] Alex Wilson. *Gradient Descent Stopping Criteria.* Accessed: 2024-23-11. 2018. URL: https://trymachinelearning.com/gradient-descent-stopping-criteria/.

[7] J. Doulgas Faires Richard L. Burden. *Numerical Analysis, 9th Edition.* Cengage Learning, 2010.

[8] Roger Grosse. *Second-Order Optimization.* Accessed: 2024-23-11. 2021. URL: https://www.cs.toronto.edu/~rgrosse/courses/csc2541_2021/readings/L04_second_order.pdf.

[9] probabilidadyestadistica.net. *Regresión lineal.* Accessed: 2024-11-29. -. URL: https://www.probabilidadyestadistica.net/regresion-lineal/.

[10] Statistics By Jim. *Mean Squared Error (MSE).* Accessed: 2024-11-29. 2023. URL: https://statisticsbyjim.com/regression/mean-squared-error-mse/.

[11] Harteker Gotz. *Análisis de Reducción de Dimensionalidad y Predicción mediante SVD.* Accessed: 2024-11-29. 2024. URL: -.

[12] John Burkardt. *Applications of the Singular Value Decomposition.* Accessed: 2024-11-09. 2019. URL: https://people.sc.fsu.edu/~jburkardt//classes/mlds_2019/svd_applications.pdf.

[13] MIT Linear Algebra Course. *Left and right inverses - pseudoinverse*. Accessed: 2024-11-29. 2011. URL: https://ocw.mit.edu/courses/18-06sc-linear-algebra-fall-2011/0550c89b69c99e97dcbf52074e293308_MIT18_06SCF11_Ses3.8sum.pdf.

[14] Geeks4Geeks. *Regularization in Machine Learning*. Accessed: 2024-11-29. 2024. URL: https://www.geeksforgeeks.org/regularization-in-machine-learning/.