Análisis de optimización y cuadrados mínimos mediante gradiente descendiente

Autores

Santiago Groba[1]
Tomás Lottersberger[2]

grobaalonsos@udesa.edu.ar[1], tlottersberger@udesa.edu.ar[2]

Universidad de San Andrés

30 de noviembre de 2024

Resumen

En este trabajo se analizaron métodos de optimización y resolución de problemas de cuadrados mínimos aplicados a la función de Rosenbrock y al dataset California Housing. Se compararon los métodos de gradiente descendiente y Newton en términos de rapidez y precisión, destacando la sensibilidad del gradiente descendiente a la tasa de aprendizaje y las condiciones iniciales, frente a la rapidez del método de Newton gracias a su convergencia cuadrática. Asimismo, se resolvió el problema de cuadrados mínimos mediante pseudoinversa y gradiente descendiente, mostrando que, aunque la pseudoinversa proporciona una solución exacta, el gradiente descendiente es una alternativa práctica para problemas de mayor escala.

1. Introducción

La regresión lineal y la optimización son técnicas fundamentales en estadística y aprendizaje automático, utilizadas para modelar relaciones entre variables y minimizar funciones objetivo. Este trabajo aborda dos problemas principales: la optimización de la función de Rosenbrock, una referencia clásica en problemas no convexos, y la predicción del valor medio de las viviendas en el estado de California utilizando características demográficas y económicas del dataset California Housing.

En el primer caso, se analizaron los métodos de optimización de gradiente descendiente y Newton, evaluando sus comportamientos en términos de convergencia, sensibilidad a las condiciones iniciales y tasa de aprendizaje. En el segundo caso, se compararon dos enfoques para resolver el problema de regresión lineal: la solución analítica mediante pseudoinversa y el método iterativo de gradiente descendiente. Ambos métodos buscan minimizar el error cuadrático medio entre las predicciones del modelo y los valores reales.

2. Marco Teórico

2.1. Descomposición en valores singulares

2.1.1. Concepto

La Descomposición en Valores Singulares (SVD, por sus siglas en inglés) es una herramienta matemática que descompone cualquier matriz $A \in \mathbb{R}^{mxn}$ en el producto de tres matrices:

$$A = U\Sigma V^T \tag{1}$$

donde U es una matriz ortogonal de tamaño m×m, cuyos vectores columna son los vectores singulares de la izquierda de A, Σ es una matriz diagonal de tamaño m×n que contiene los valores singulares de A ordenados de mayor a menor y V^T es una matriz ortogonal de tamaño n×n, cuyos vectores columna son los vectores singulares de la derecha de A.

Esta descomposición tiene múltiples aplicaciones en áreas como la reducción de dimensiones, la compresión de datos y el procesamiento de señales, debido a su capacidad para descomponer los datos en componentes significativos y menos significativos.

2.1.2. Interpretación

Los valores singulares en Σ representan la "importancia" de cada componente en la estructura de datos original. Los valores singulares más

grandes están asociados con componentes que contienen más información, mientras que los valores singulares menores pueden considerarse como "ruido" o detalles menos relevantes. Al reducir la cantidad de valores singulares en la reconstrucción, se logra una versión comprimida de la matriz original.

2.2. Cuadrados mínimos

El método de cuadrados mínimos es una técnica utilizada para ajustar un modelo lineal a un conjunto de datos, minimizando la suma de los cuadrados de las diferencias entre los valores observados y los valores predichos por el modelo. Si se posee una matriz X de tamaño m $\times$ n donde m$\geq$n, y un vector y en $\mathbb{R}^m$, el objetivo es hallar la solución al problema $X\beta = y$. Dado el caso que $y \notin <x_1, x_2, ..., x_n>$ se debe buscar un vector $\beta \in \mathbb{R}^m$ que minimice el residuo (o error).

$$||X\beta - y||_2 \tag{2}$$

Aproximando linealmente cuando $x \in \mathbb{R}^{n=1}$ dado un conjunto de datos (x_i, y_i), el modelo lineal se puede expresar como:

$$y = \beta_0 + \beta_1 x \tag{3}$$

Donde β_0 y β_1 son los coeficientes del modelo. El objetivo es encontrar los valores de β_0 y β_1 que minimizan la suma de los cuadrados de las diferencias entre los valores de y_i y la aproximación resultante del modelo lineal:

$$E_2(\beta_1, \beta_2) = \sum_{i=1}^{n} [y_i - (\beta_1 x_i + \beta_0)]^2 \tag{4}$$

De manera tal que se minimice la suma de los cuadrados de las diferencias entre los valores de y_i y la aproximación resultante del modelo lineal. Esta clase de error le otorga un mayor peso a los puntos que están muy alejados de la aproximación aunque no le permite dominar la aproximación por completo. La solución analítica a este problema se puede obtener derivando E_2 con respecto a $\beta_0 y \beta_1$ e igualando las derivadas a cero. La solución se puede escribir en forma matricial como:

$$\hat{\beta} = (X^T X)^{-1} X^T y \tag{5}$$

Donde X es la matriz de datos formada por los valores de x_i y un vector de unos, $\hat{\beta}$ es el vector de coeficientes estimados y y es el vector de valores observados.

2.3. Error cuadrático medio

El error cuadrático medio (MSE, por sus siglas en inglés) es una métrica que mide el desempeño de un modelo al comparar los valores predichos ($\hat{y}$) con los valores reales (y). Es ampliamente utilizado en problemas de regresión debido a su simplicidad y capacidad para penalizar errores grandes.

Para un conjunto de datos con n muestras, el MSE se define como:

$$MSE(\omega) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{6}$$

donde y_i es el valor real de la i-ésima muestra, $\hat{y}_i = X_i \omega$ es el valor predicho por el modelo para la i-ésima muestra, X_i es el vector de características de la i-ésima muestra y ω es el vector de parámetros del modelo.

El MSE puede definirse de manera matricial para todo el conjunto de datos como:

$$MSE(\omega) = \frac{1}{n} ||y - X\omega||_2^2 \tag{7}$$

donde $y \in \mathbb{R}^n$ es el vector de valores reales, $X \in \mathbb{R}^{n \times d}$ es la matriz de características, $\omega \in \mathbb{R}^d$ es el vector de parámetros desconocidos y $|| \cdot ||_2^2$ es la norma euclidiana al cuadrado.

2.4. Matriz pseudoinversa

La pseudoinversa es una generalización de la inversa de una matriz que se aplica cuando una matriz no es cuadrada o no es invertible. Se denota como $X^\dagger$ y tiene aplicaciones fundamentales en álgebra lineal y optimización, como en la solución de problemas de cuadrados mínimos.

La pseudoinversa se calcula comúnmente utilizando SVD (véase en 2.1), donde se define como:

$$X^\dagger = V\Sigma^\dagger U^T \tag{8}$$

donde $\Sigma^\dagger$ es la inversa generalizada de Σ.

Esto permite manejar matrices singulares o mal condicionadas, donde algunos valores singulares son cercanos a cero o nulos.

2.5. Gradiente descendiente

El gradiente descendiente es un método iterativo de optimización ampliamente utilizado para encontrar el mínimo de una función objetivo. En problemas de aprendizaje automático, se aplica comúnmente para minimizar funciones de pérdida, como el MSE(véase en 2.3) en la regresión lineal.

Dado un conjunto de parámetros ω, el algoritmo actualiza sus valores según la regla:

$$\omega_{t+1} = \omega_t - \eta\nabla f(\omega_t) \tag{9}$$

donde ω_t son los valores actuales de los parámetros en la iteración t, η es la tasa de aprendizaje, un factor que controla el tamaño del paso y $\nabla f(\omega_t)$ es el gradiente de la función objetivo $f(\omega)$ evaluado en ω_t.

2.5.1. Orden de convergencia

El Gradiente Descendente tiene una convergencia lineal bajo condiciones generales. Esto significa que el error en cada iteración se reduce proporcionalmente con una constante $0 < \gamma < 1$ tal que:

$$||\omega_{t+1} - \omega^*|| \leq \gamma||\omega_t - \omega^*|| \tag{10}$$

2.6. Método de Newton

El Método de Newton es una técnica de optimización que se utiliza para encontrar el mínimo o máximo de una función, especialmente cuando esta es diferenciable. Es un método iterativo que, a partir de un punto inicial, busca encontrar la raíz (o ceros) de la derivada de la función objetivo.

Supongamos que se tiene una función objetivo $f(\omega)$ y se desea encontrar su mínimo. El Método de Newton parte de una aproximación cuadrática de la función alrededor de un punto ω_t, en cada iteración t, y actualiza los valores de los parámetros ω siguiendo la fórmula:

$$\omega_{t+1} = \omega_t - H^{-1}\nabla f(x_t) \tag{11}$$

El valor actual de los parámetros es denotado por ω_t. El gradiente, $\nabla f(\omega_t)$, corresponde al vector de derivadas primeras de la función objetivo evaluadas en ω_t. La matriz H, conocida como la matriz Hessiana, es la matriz de derivadas segundas de la función $f(\omega)$, y representa la curvatura de la función en el punto ω_t.

2.6.1. Orden de convergencia

El Método de Newton tiene una convergencia cuadrática bajo condiciones ideales. Esto significa que el error en cada iteración se reduce aproximadamente al cuadrado del error de la iteración anterior. Formalmente, se dice que un algoritmo tiene convergencia cuadrática si existe una constante C tal que, en cada iteración t:

$$||\omega_{t+1} - \omega^*|| \leq C||\omega_t - \omega^*||^2 \tag{12}$$

donde ω^* es el mínimo global y ω_T es la estimación en la iteración t.

3. Experimentación numérica, consideraciones y resultados

En esta sección se describirán los experimentos numéricos realizados, incluyendo los procedimientos correspondientes y los resultados alcanzados.

3.1. Gradiente descendiente

El gradiente descendiente es un algoritmo iterativo de optimización que busca minimizar una función de error ajustando los parámetros del modelo en la dirección opuesta al gradiente de la función. En este proceso, el learning rate (η) es un parámetro crucial que determina el tamaño del paso dado en cada iteración. A continuación se detallan algunos de los factores que afectan el desempeño del gradiente descendiente.

3.1.1. Varianza del learning rate

La varianza de la tasa de aprendizaje (η) tiene un impacto significativo en la convergencia del gradiente descendiente. Un valor demasiado alto puede provocar que el algoritmo salte sobre el mínimo de la función de error, resultando en una divergencia del proceso de optimización. En cambio, un valor demasiado bajo puede hacer que la convergencia sea extremadamente lenta, lo que aumenta el tiempo de cómputo sin alcanzar un resultado óptimo en el tiempo razonable.

En nuestros experimentos, se evaluaron varios valores de η para observar su influencia en el tiempo de convergencia y la precisión del modelo ajustado. A medida que la varianza de η aumenta, se observa que el algoritmo tiende a oscilar alrededor del mínimo global sin llegar a estabilizarse completamente. Por otro lado, con un η pequeño, aunque la convergencia es más estable, el proceso tarda mucho más en llegar al mínimo deseado.

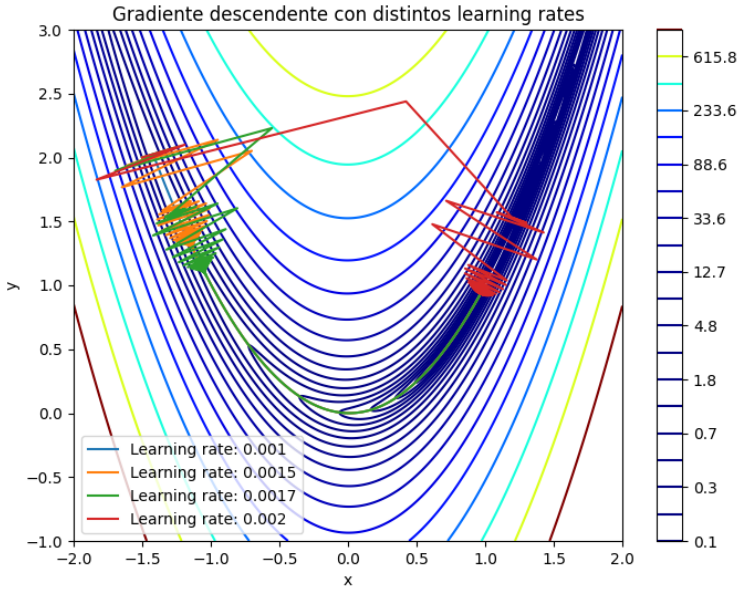

Figura 1

Este gráfico 1 muestra las trayectorias del algoritmo de gradiente descendente para minimizar la función de Rosenbrockn. Las trayectorias se trazan para diferentes tasas de aprendizaje (η): 0.001, 0.0015, 0.0017 y 0.002. La elección de la tasa de aprendizaje tiene un impacto significativo en la convergencia del algoritmo de gradiente descendente. Una tasa de aprendizaje pequeña (por ejemplo, 0.001) conduce a una convergencia más estable pero más lenta, mientras que una tasa más alta (por ejemplo, 0.002) resulta en una convergencia más rápida, aunque con más oscilaciones alrededor del mínimo.

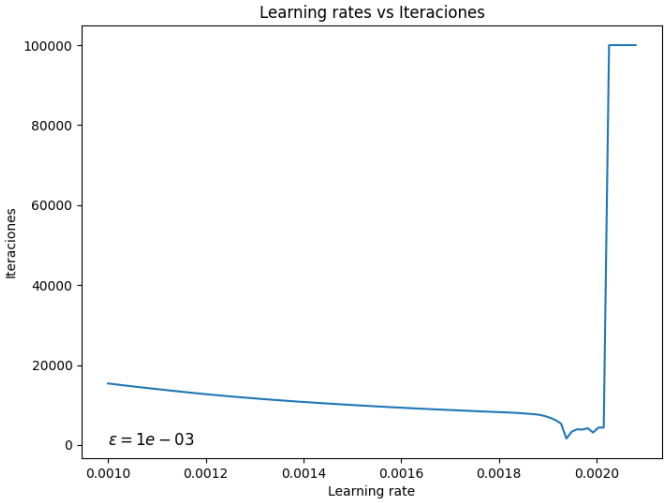

Figura 2

En la figura 2 podemos ver como al aumentar el learning rate se deben hacer menos iteraciones para converger al resultado, pero hay un detalle muy importante que luego de pasar un learning rate optimo se vuelve mas inestable hasta que termina sin converger. Se puede ver que pasando el step 0.002 deja de converger cortando en el limite impuesto de 100.000 iteraciones.

En esta figura 3 se puede ver lo mismo que en la figura 2 pero desde el punto de vista de error cuadrático medio. los learning rates mas chicos tardan mas en converger mientras que los mas grades lo hacen mas rápido, en este caso se agrego el learning rate que consiguió mínimas iteraciones y vemos que es mas pequeño que 0.002, esto es debido a la inestabilidad que se presenta.

3.1.2. Puntos iniciales

El comportamiento del gradiente descendiente también está estrechamente relacionado con los puntos iniciales de la optimización. Si los parámetros iniciales son seleccionados aleatoriamente en una región no óptima del espacio de parámetros, el algoritmo puede quedar atrapado en un **mínimo local** en lugar de alcanzar el mínimo global. Para mitigar este efecto, se probaron diferentes estrategias para la inicialización de los puntos, como la inicialización aleatoria, la inicialización con valores cercanos al óptimo basado en conocimiento previo, y la inicialización basada en una heurística derivada de la estructura del problema.

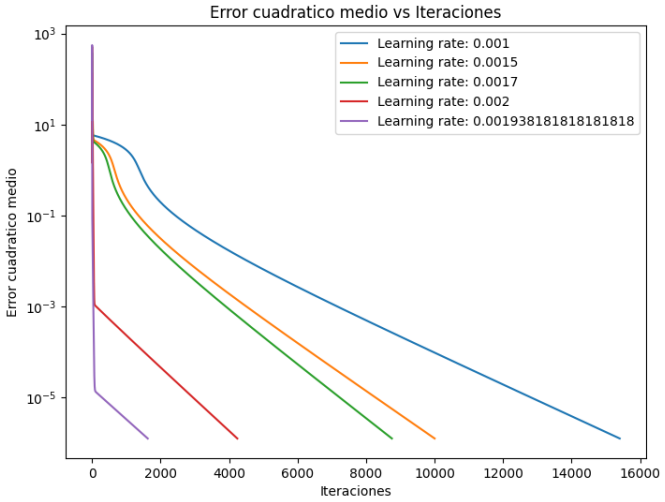

Figura 3

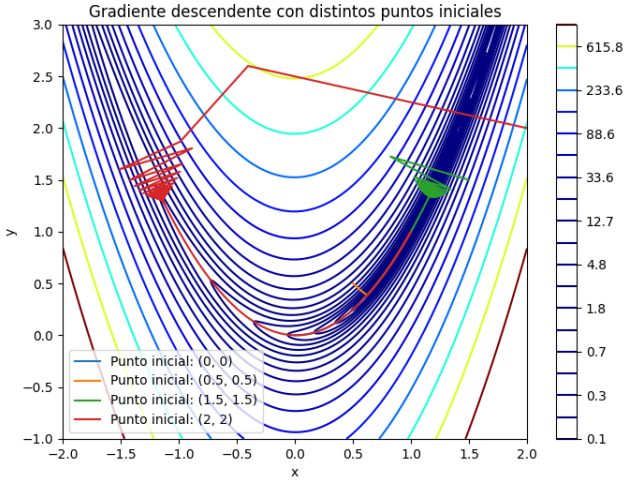

Figura 4

Los resultados de la figura 4 muestran que, en general, una mejor elección de los puntos iniciales acelera la convergencia y evita que el algoritmo se estanque en mínimos locales, mejorando la efectividad del proceso de optimización. El punto inicial (1.5, 1.5) parece tener la tasa de convergencia más rápida entre las condiciones iniciales probadas. Otros puntos iniciales, en particular (2, 2), parecen requerir más iteraciones para alcanzar el mínimo.

3.1.3. Método de Newton

El método de Newton es un algoritmo de optimización que, a diferencia del gradiente descendiente clásico, utiliza tanto el gradiente como la segunda derivada (la matriz Hessiana) de la función objetivo para realizar una actualización más precisa de los parámetros. Este método puede ser especialmente útil cuando se busca una convergencia más rápida cerca de un mínimo local, ya que la información adicional sobre la curvatura de la función permite una mayor precisión en el ajuste de los parámetros.

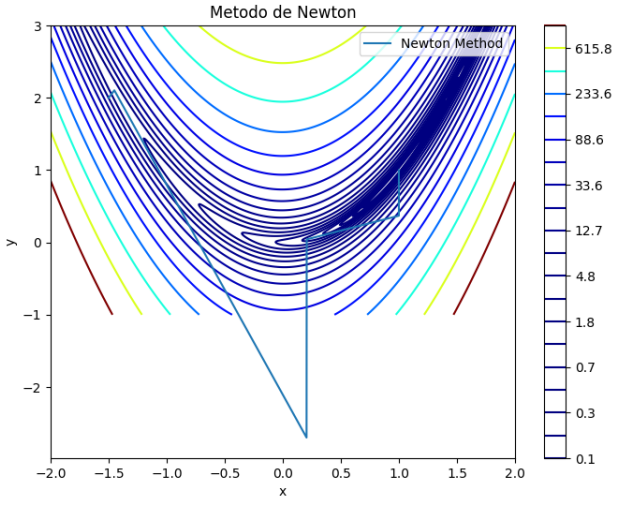

Figura 5

El gráfico 5 muestra la trayectoria del método de Newton para minimizar la función de Rosenbrock. El método de Newton es un algoritmo de optimización alternativo que utiliza tanto el gradiente como la matriz Hessiana de la función objetivo. El uso de la matriz Hessiana en el método de Newton permite al algoritmo considerar la curvatura de la función objetivo, lo que conduce a actualizaciones más eficientes de las variables de optimización.

Sin embargo, el método de Newton también tiene desventajas. La principal de ellas es la necesidad de calcular y almacenar la matriz Hessiana, lo cual puede ser costoso computacionalmente, especialmente cuando se trabaja con modelos de alta dimensionalidad. En los experimentos realizados, se comparó el rendimiento del gradiente descendiente y el método de Newton en términos de rapidez y precisión en la convergencia.

El gradiente descendiente, con su orden de convergencia lineal, mostró una reducción constante del error proporcional a cada iteración, pero su velocidad depende fuertemente de factores como la tasa de aprendizaje y las condiciones iniciales. Por otro lado, el método de Newton, con su convergencia cuadrática, destacó por una rápida reducción del error en las iteraciones finales, especialmente cerca del mínimo global, donde su eficiencia es notablemente superior.

Los resultados indican que, en problemas donde la matriz Hessiana es difícil de calcular o de gran tamaño, el método de Newton puede resultar más lento o impráctico. Sin embargo, en problemas de menor escala o cuando la función objetivo tiene una buena aproximación cuadrática, el método de Newton ofrece ventajas claras en términos de eficiencia de convergencia, gracias a su comportamiento cuadrático.”

3.2. Cuadrados mínimos mediante descenso por gradiente

Esta experimentación se basó en minimizar una función objetivo, la cual fue la de MSE (véase en 2.3), de ecuación 6. Para solucionar este problema se establecieron dos enfoques: uno analítico mediante pseudoinversa (véase en 2.4) y otro iterativo mediante el método de gradiente descendiente (véase en 2.5).

3.2.1. Utilización de dataset

Para esta experimentación se utilizó el dataset *California Housing*, disponible en la biblioteca sklearn[5]. Este conjunto de datos proporciona información sobre características demográficas y económicas de bloques habitacionales en el estado de California, con el objetivo de predecir el valor medio de las viviendas en dichas regiones.

El dataset contiene un total de 20,640 muestras, cada una de las cuales representa un bloque habitacional. Las variables predictoras están normalizadas para facilitar el análisis y mejorar la estabilidad numérica de los métodos implementados. Cada muestra consta de 8 (ocho) variables (es decir, forman vectores de $\mathbb{R}^8$) que incluyen, por ejemplo, el promedio de habitaciones, dormitorios y ocupantes por vivienda y la variable objetivo (y) que es el valor medio de las viviendas por bloque.

El dataset se dividió en un conjunto de entrenamiento (que equivale al 80 % de las muestras) y un conjunto de prueba (que equivale al 20 % de las muestras) de manera aleatoria.

3.2.2. Resolución analítica por pseudoinversa

El cálculo analítico de la pseudoinversa se basa en encontrar la solución exacta al problema de mínimos cuadrados ordinarios. Se desarrolló en su sección correspondiente (6.1) del apéndice.

Se obtuvo que:

$$\omega = (X^T X)^{-1} X^T y \tag{13}$$

Un problema posible de la ecuación 13 es que $X^T X$ no sea invertible ya que, por ejemplo, X puede estar mal condicionada o no tener rango completo, por lo que se optó desarrollar la pseudoinversa mediante SVD (véase ecuación 8).

Con este método se obtuvo:

$$\omega = X^{\dagger} y \tag{14}$$

Cada coeficiente de ω representa cómo cambia el valor medio de las viviendas cuando aumenta en una unidad la característica correspondiente, manteniendo las demás constantes. Por ejemplo, si un coeficiente asociado a la característica "Ingresos promedio" es alto, esta característica tiene un fuerte impacto en el valor de las viviendas.

Los resultados obtenidos por la solución de la matriz pseudoinversa fueron los siguientes:

Índice	Componente	Valor
0	MedInc	2,0720
1	HouseAge	0,8544
2	AveRooms	0,1226
3	AveBedrms	-0,2944
4	Populations	0,3393
5	AveOccup	-0,0023
6	Latitude	-0,0408
7	Longitude	-0,8969
8	Término independiente	-0,8698

3.2.3. Consideraciones para el método de gradiente descendiente

Para ejecutar el método de gradiente descendiente debimos tener ciertas consideraciones en cuenta para los parámetros de éste, a continuación se detallarán estos aspectos:

La tasa de aprendizaje (η) es el determinante de cuán grandes son los pasos que toma el

descenso del gradiente en la dirección del mínimo local. Debido a esto, la elección de la tasa de aprendizaje es importante y tiene un impacto significativo en la estabilidad y velocidad de convergencia del algoritmo. Un η muy pequeño garantiza una convergencia estable hacia el mínimo pero a una velocidad lenta, mientras que un η muy grande aceleraría el algoritmo pero aumenta el riesgo de oscilaciones cerca del mínimo o incluso divergencia.

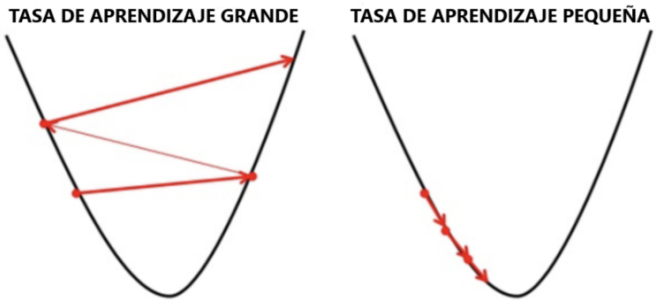

Figura 6: Comparación entre la elección de un η pequeño y otro grande.

Por el motivo visible en la figura 6 es que se tomó la decisión de calcular un η_{optimo}, el cual asegure converger hacia el mínimo en la máxima velocidad posible. Dicho η_{optimo} se calculó de la siguiente manera:

$$\eta_{optimo} = \frac{1}{\sigma_1^2} \tag{15}$$

donde σ_1 es el mayor valor singular de la matriz de X, obtenido usando SVD.

La resolución a la que se ha llegado a través de la ecuación 15 es

$$\eta_{optimo} = \frac{1}{(182,86211)^2} = 3 \cdot 10^{-5}$$

el cálculo se realizó mediante la librería de Python "NumPy"[5] y puede verse en su sección correspondiente (véase en 6.2) del apéndice.

En la siguiente figura se puede observar una comparación realizada entre el η_{optimo} calculado y dos η arbitrarios, uno más grande que η_{optimo} y otro más pequeño para determinar la convergencia de cada uno.

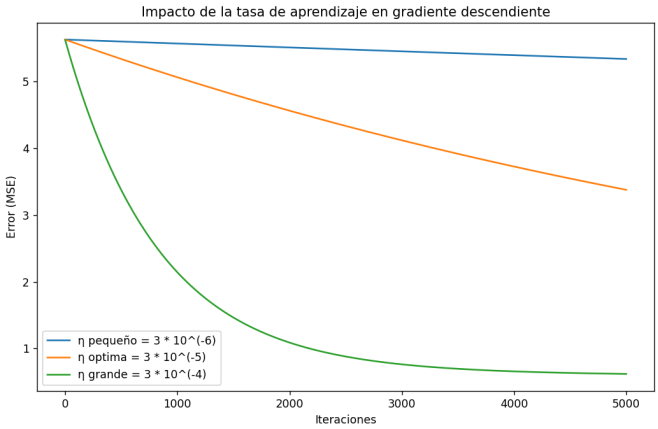

Figura 7: Comparación de comportamiento del método de gradiente descendiente con diferentes tasas de aprendizaje alrededor de η_{optimo}.

En la figura 7 se puede observar que, al evaluar un η 10 (diez) veces más grande que el η_{optimo} la convergencia del método es mucho más rápida, donde parece converger a 0 (cero) antes de las 5000 (cinco mil) iteraciones, mientras que con un η 10 (diez) veces más chico que el η_{optimo} apenas se puede denotar una variación en el método, haciendo que éste tarde mucho más en converger hacia el mínimo.

Otra consideración importante que se debió tener en cuenta fue la elección de iteraciones a realizar por el método de gradiente descendiente, para esto se iteró hasta encontrar un error medio arbitrario el cual, para esta experimentación, se estableció en 10^{-4}.

El resultado de dicho procedimiento arrojó que el error medio elegido ocurre exactamente a las 10000 (diez mil) iteraciones.

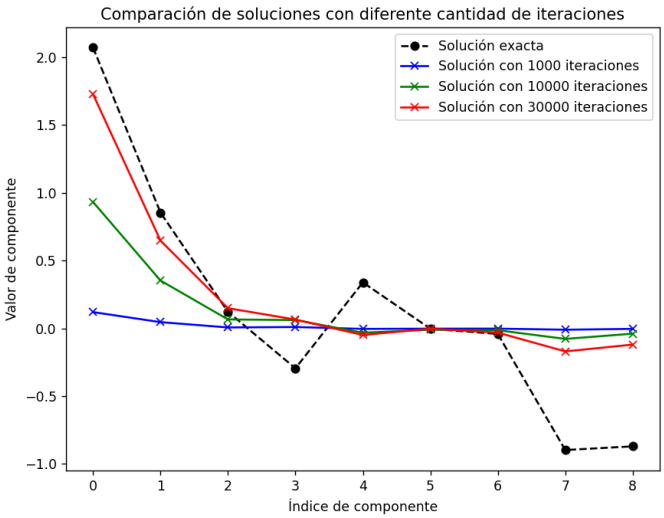

Figura 8: Comparación de los resultados obtenidos a raíz de cambiar la cantidad de iteraciones que el método de gradiente descendiente realiza.

En la figura 8 se observa que al aumentar la cantidad de iteraciones que el método de gradiente descendiente realiza los resultados son más exactos, por éste motivo, por el resto de la experimentación usaremos una cantidad de iteraciones fijada en 30000 (treinta mil) ya que arroja resultados certeros y no es tan computacionalmente caro.

3.2.4. Aplicación de gradiente descendiente

Para minimizar la función MSE mediante el método de gradiente descendiente se sustituyó $\nabla f(\omega_t)$ en la ecuación 9 por la ecuación 17 correspondiente al gradiente de MSE y también la tasa de aprendizaje η por el η_{optimo} calculado (véase en 3.2.3).

Entonces se obtuvo:

$$\omega gd_{t+1} = \omega_t - 3 \cdot 10^{-5}(-\frac{2}{n}X^T(y - X\omega)) \quad (16)$$

De esta manera se obtuvo el valor de cada componente, aproximándolo a través del método iterativo, el cual arrojó los siguientes resultados:

Índice	Componente	Valor
0	MedInc	1,7275
1	HouseAge	0,6474
2	AveRooms	0,1491
3	AveBedrms	0,0655
4	Populations	-0,0488
5	AveOccup	-0,0022
6	Latitude	-0,0309
7	Longitude	-0,1695
8	Término independiente	-0,1194

3.2.5. Comparación de los resultados obtenidos

Como se ha dicho anteriormente en este informe, la pseudoinversa arroja la solución exacta al problema planteado, por otro lado, con el método iterativo de gradiente descendiente se llega a una solución aproximada en base a iteraciones. Por estos motivos se comparó ambas soluciones para evaluar qué tan bien es que el método de gradiente descendiente aproximó el resultado exacto.

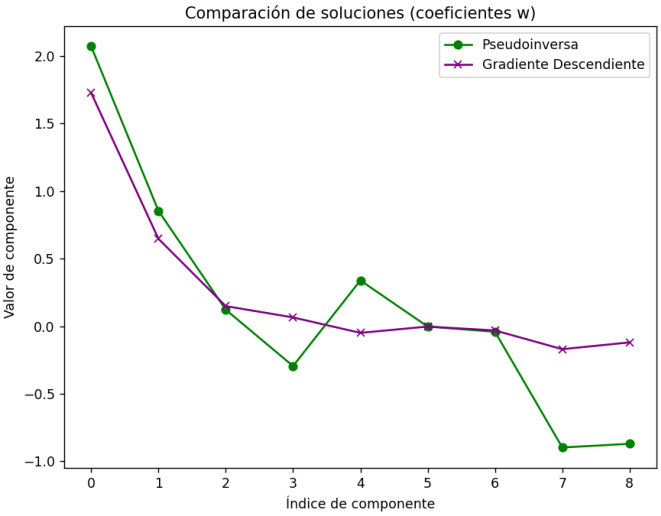

Figura 9: Comparación de los resultados obtenidos mediante la pseudoinversa y los obtenidos mediante el método de gradiente descendiente.

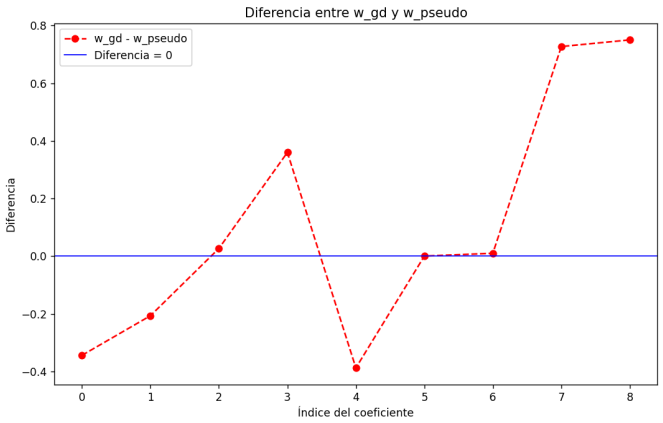

Figura 10: Diferencia punto a puntos de las soluciones de pseudoinversa y gradiente descendiente, donde cada punto es la diferencia de cada índice.

En las figuras 9 y 10 se puede observar que, si bien el método de gradiente descendiente no da la solución exacta, se acerca mucho en su comportamiento a la solución exacta calculada mediante pseudoinversa, lo que sugiere que funciona muy bien para realizar este tipo de aproximaciones.

4. Conclusiones

Se estudiaron los métodos de gradiente descendiente y método de Newton aplicados a la optimización de la función de Rosenbrock, una referencia clásica en problemas de optimización no convexa. A través de la experimentación con diferentes tasas de aprendizaje y condiciones iniciales, se observaron comportamientos que reflejan las propiedades teóricas de cada método.

El gradiente descendiente demostró ser un método flexible y fácil de implementar, pero su desempeño está altamente influenciado por la elección de la tasa de aprendizaje (η) y las condiciones iniciales. Si bien puede converger de manera estable con una tasa de aprendizaje adecuada, el tiempo de convergencia aumenta significativamente en comparación con el método de Newton. Además, su sensibilidad a las condiciones iniciales destacó los desafíos de alcanzar el mínimo global en funciones no convexas.

Por otro lado, el método de Newton, gracias a su tasa de convergencia cuadrática, mostró una ventaja significativa en términos de rapidez y precisión. Sin embargo, su dependencia del cálculo y la inversión de la matriz Hessiana lo hace menos práctico en problemas de alta dimensionalidad o donde los recursos computacionales son limitados.

Además, en este trabajo se abordó la solución del problema de cuadrados mínimos ordinarios utilizando dos métodos: la pseudoinversa y el gradiente descendente. A partir de la experimentación, se obtuvieron resultados tales como que la pseudoinversa proporciona una solución exacta y analítica al problema en una única operación. Por otro lado, el gradiente descendiente es un método iterativo que aproxima la solución óptima al minimizar progresivamente la función objetivo, pero para que esto sea eficiente deben tenerse en cuenta consideraciones cruciales como la elección de una tasa de aprendizaje óptima y una cantidad suficiente de iteraciones.

El uso del dataset California Housing permitió evaluar estos métodos en un contexto real, donde se analizaron relaciones entre características demográficas/económicas y el valor medio de las viviendas. Para este problema, la pseudoinversa ofreció una solución más rápida y precisa debido al tamaño manejable del dataset. En cambio el gradiente descendiente, aunque más lento, demostró ser una alternativa escalable y práctica para datasets de gran tamaño donde calcular la pseudoinversa sería impráctico.

En trabajos futuros podrían implementarse técnicas como la regularización L2 para analizar cómo la introducción de un término de penalización afecta el comportamiento y la estabilidad de los métodos.

5. Bibliografía

- Harris, C.R. et al., 2020. Array programming with NumPy. Nature, 585, pp.357–362.
- Pedregosa, F. et al., 2011. Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), pp.2825–2830.
- Hunter, J.D., 2007. Matplotlib: A 2D graphics environment. Computing in science & engineering, 9(3), pp.90–95.

6. Apéndice

6.1. Pseudoinversa analítica de MSE

Desarrollando el término cuadrático de la ecuación 7:

$$MSE(\omega) = \frac{1}{n}(y - X\omega)^T(y - X\omega)$$

aplicando gradiente:

$$\nabla MSE(\omega) = -\frac{2}{n}X^T(y - X\omega) \qquad (17)$$

se obtiene un sistema lineal que se puede resolver analíticamente si igualamos a cero:

$$\nabla MSE(\omega) = -\frac{2}{n}X^T(y - X\omega) = 0$$

De aquí se deduce:

$$X^T X\omega = X^T y$$

Si $X^T X$ es invertible entonces la solución exacta es:

$$\omega = (X^T X)^{-1} X^T y$$

6.2. Cálculo de η_{optimo}

```
def optimum_learning_rate(X_train):
    U, S, Vt = np.linalg.svd(X_train, full_matrices=False)
    sigma_1 = S[0]
    eta_opt = 1 / (sigma_1**2)
    return eta_opt
```

```
Mayor valor singular sigmauno: 182.86211
Tasa de aprendizaje optima eta: 0.00003
```

Figura 11: Código del cálculo de η_{optimo}.