Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computación.

Programa de Bachillerato en Ingeniería en Computación

IC6400 - Investigación de Operaciones

Trabajo Práctico 02: Optimización

Profesor: Ph. D. Saúl Calderón Ramírez

Estudiantes

Brandon Retana Chacón	2021121141
Ervin Rodríguez Villanueva	2021095970
Josué Castro Ramírez	2020065036

Segundo semestre 2023

Índice

1. Introducción

El presente trabajo trata sobre la implementación en pytorch de los algoritmos *Newton Rhapson, Descenso del Gradiente Adaptativo,* y *Simulated Annealing.* La idea es implementar estos algortimos con distintas funciones matemáticas y estudiar su comportamiento y características; primeramente optimizando los hiperparámetros de cada función con la biblioteca de *Optuna* para evaluar y comparar los resultados.

2. Optimización de funciones

A continuación se explorarán gráficamente una serie de funciones matemáticas con utilizando curvas de nivel, estas curvas nos permitirán examinar en detalle la topografía de las funciones, y a partir de ellas identificar posibles puntos silla, evaluar la convexidad de las superficies y localizar los puntos mínimos.

2.1. Funciones a Optimizar

Sean f_0, f_1 y f_2 las funciones a optimizar en este trabajo práctico, las cuáles se definen a continuación:

- $f_0 : f_0(x, y) = x^2 + y^2$
- $f_1 : f_1(x, y) = (1{,}5 - x + xy)^2 + (2{,}25 - x + xy^2)^2 + (2{,}625 - x + xy^3)^2$
- $f_2 : f_2(x, y) = 0{,}26(x^2 + y^2) - 0{,}48xy$

  con $x, y \in [-10, 10]$.

2.2. Análisis de la función f_0

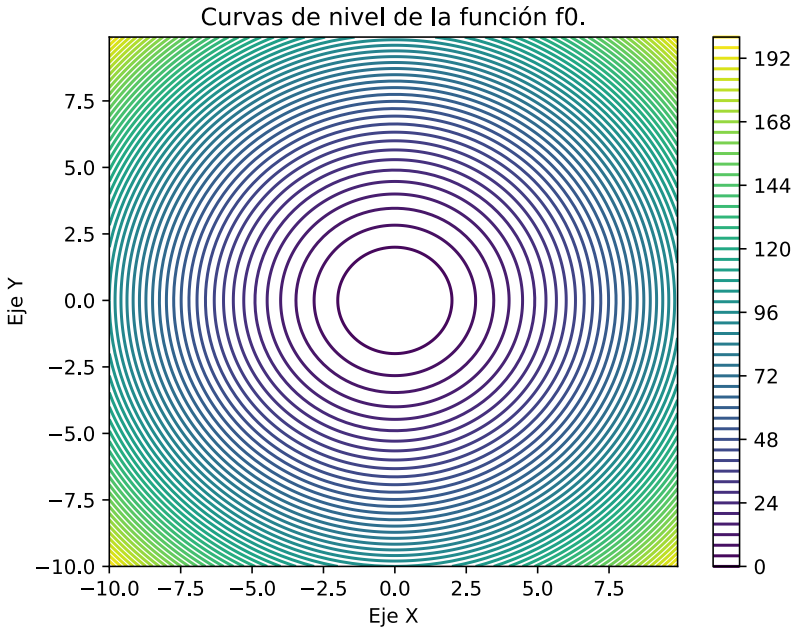


Figura 1: Curvas de nivel de la función f_0

Cuando observamos las curvas representadas en la Figura 1 es evidente que esta **es una función cóncava, es decir no convexa,** debido a que sus curvas de nivel van tendiendo de los puntos más grandes a los puntos más pequeños dejándonos una sola zona en el centro, la cual podemos asumir como el **punto mínimo global de la función** $p = [0, 0]$. Además, la gama de color representada a la derecha de la figura nos indica que cuando los puntos se acercan a 0 predomina la gama de color morado, lo que también es un indicativo de que esta función es convexa. De esta forma podemos concluir que **no se observan valles o puntos silla.**

2.3. Análisis de la función f_1

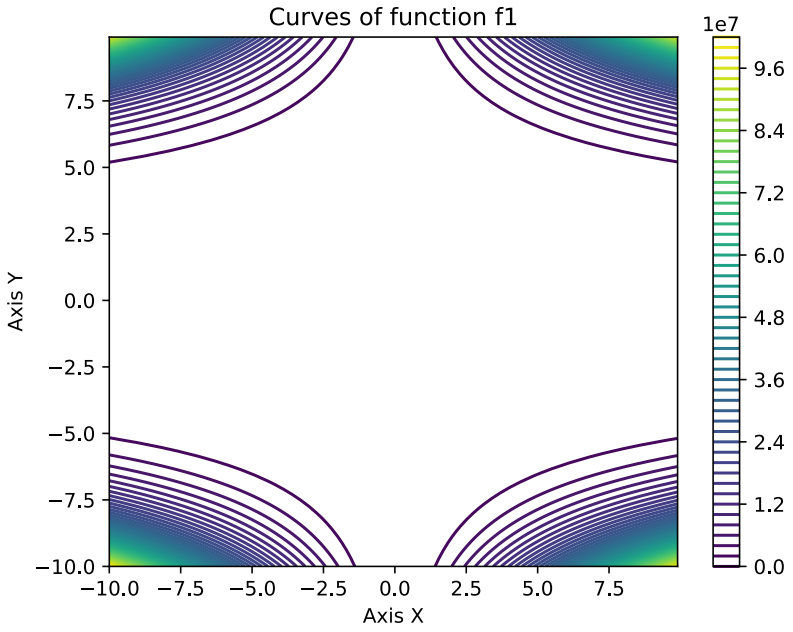


Figura 2: Curvas de nivel de la función f_1

Cuando observamos las curvas representadas en la Figura 2 no es tan evidente que esta **es una función cóncava, es decir no convexa,** pero si tomamos $p = \left[3, \frac{1}{2}\right]$ y lo evaluamos de la en la función, tenemos que $f_1(3, \frac{1}{2}) = 0$. Por lo que podemos asumir a este punto p como el **mínimo global de la función.** Además, si observásemos la gráfica de toda la función podríamos notar que tiene forma de una hoja de papel estirada hacia arriba desde sus cuatro puntas, pero, manteniendo una planicie prolongada bajo cierto rango, siendo estos puntos **puntos silla.**

2.4. Análisis de la función f_2

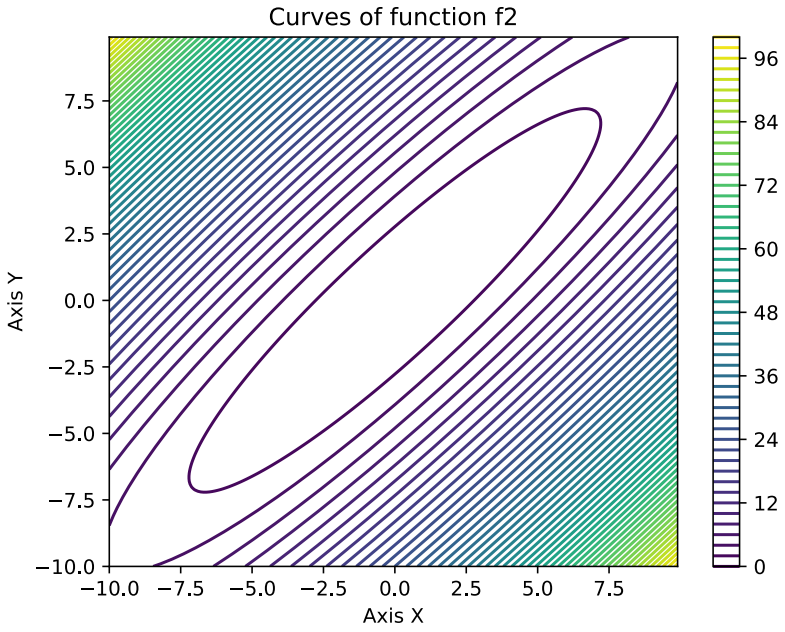


Figura 3: Curvas de nivel de la función f_2

Cuando analizamos las gráficas mostradas en la Figura 3, es fácil notar que se trata de una **función cónica, es decir, no convexa.** Esto se debe a que las curvas de nivel tienden de los valores más altos a los más bajos, dejando una única región central que podemos interpretar como el mínimo global de la función en el punto $p = [0, 0]$. Además, la paleta de colores representada en el lado derecho de la figura indica que a medida que los valores se acercan a 0, prevalece el tono morado, lo que también sugiere que la función es convexa en esa región. En resumen, no se observan valles ni puntos silla en esta representación.

2.5. Puntos iniciales

Para probar los distintos algoritmos se realizaron diez ejecuciones distintas, para cada ejecución se definió un punto inicial para las tres funciones. Se crearon puntos aleatorio con la función ***torch.rand*** y el uso de la función ***torch.manual_seed*** para asegurarnos de que cada función a probar tenga el mismo punto inicial. A continuación una tabla con los diez puntos iniciales.

Número de Iteración	**Punto Inicial [x, y]**	**Semilla utilizada**
0	[4.9626, 7.6822]	0
1	[7.5763, 2.7931]	1
2	[6.1470, 3.8101]	2
3	[0.0426, 1.0557]	3
4	[5.5964, 5.5909]	4
5	[8.3025, 1.2611]	5
6	[5.7220, 5.5388]	6
7	[5.3492, 1.9880]	7
8	[5.9793, 8.4530]	8
9	[6.5578, 3.0202]	9

Cuadro 1: Puntos iniciales para cada ejecución de los algoritmos.

3. Algoritmo Newton Rhapson

3.1. Gráficas de calibración de Optuna

Para calibrar los híper-parámetros del algoritmo Newton Rhapson se ha implementado un estudio utilizando correspondiente en el framework Optuna. En este estudio, se evalúan diferentes valores de α que se encuentran en el intervalo de $0{,}01 < \alpha < 3$ para las funciones f_0, f_1 y f_2. Se busca determinar diferentes valores óptimos de α que permitirán al algoritmo converger de una manera más rápida en su respectiva función a evaluar.

Cuadro 2: Mejores valores de α para las funciones

Función	**Intervalo de α**
f_0	$\alpha = 0{,}9870813675771739$
f_1	$\alpha = 1{,}4006331054549697$
f_2	$\alpha = 1{,}017869214248408$

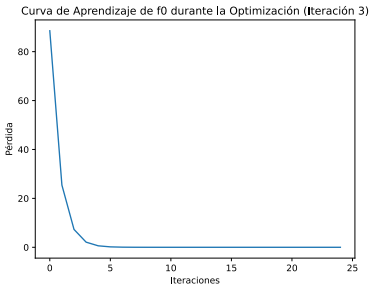
La tabla anterior muestra los mejores hiperparametros seleccionados por optuna luego de realizar un análisis de 100 iteraciones donde se probaron distintos valores para cada una de las funciones, a continuación se muestran las gráficas de aprendizaje para cada una de las funciones aplicando este algoritmo y su respectiva combinación de parámetros.

Cuadro 3: Valores de α para cada iteración en f_0

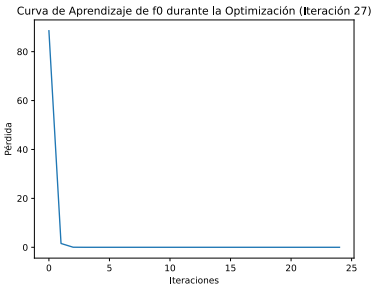
Subfigura	**Valor de α obtenido**
a	0,9609498132547092
b	0,678573801276111
c	2,8100013329038465

Cuadro 4: Resultados obtenidos en cada iteración de f_0

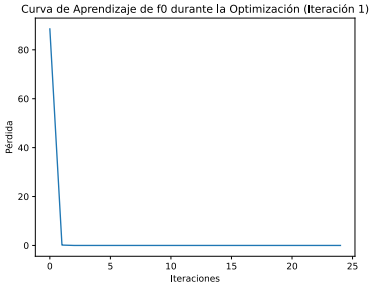
Subfigura	**Valor de α obtenido**
a	0,0
b	$2{,}0009538902465107e-23$
c	677825229094912,0



(a) Iteración 3



(b) Iteración 27



(c) Iteración 1

Figura 4: Curvas de aprendizaje en la función f_0 con Newton Rhapson

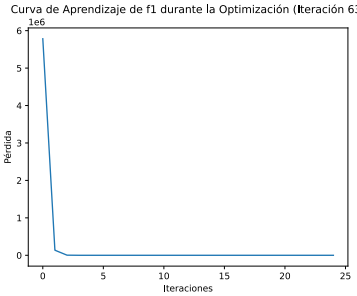
Luego de evaluar los resultados obtenidos y el comportamiento de la función de aprendizaje de los distintos valores para α se logra observar que este algoritmo converge de manera muy rápida y hasta donde se logra observar para esta primera función los α pequeños contribuyen a obtener valores pequeños en los resultados de Optuna.

Cuadro 5: Valores de α para cada iteración en f_1

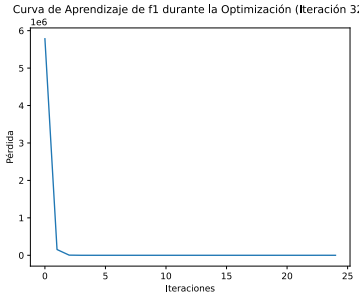
Subfigura	**Valor de α obtenido**
a	2,359428450538155
b	2,215599854888737
c	2,059947470306036

Cuadro 6: Resultados obtenidos en cada iteración de f_1

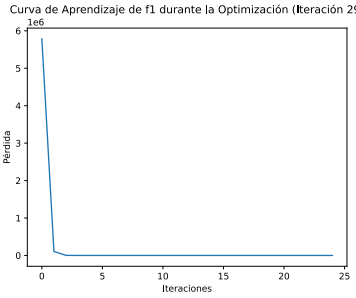
Subfigura	**Valor de α obtenido**
a	2,418576955795288
b	15,741938591003418
c	507945472,0



(a) Iteración 63



(b) Iteración 32



(c) Iteración 29

Figura 5: Curvas de aprendizaje en la función f_1 con Newton Rhapson

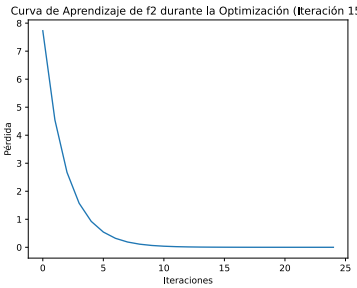
Al igual que la evaluación en $f0$ anterior, estas gráficas demuestran la rápida convergencia del algoritmo Newton Rhapson sin importar el valor de α que se seleccione, las 3 gráficas ejemplifican una rápida convergencia a un valor óptimo

Cuadro 7: Valores de α para cada iteración en f_2

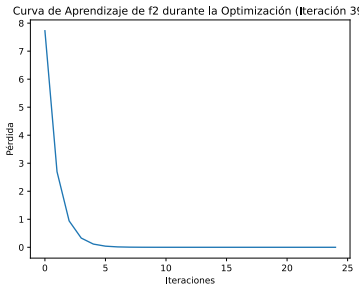
Subfigura	**Valor de α obtenido**
a	1,0414228601636484
b	2,000062008785881
c	2,7938342743454014

Cuadro 8: Resultados obtenidos en cada iteración de f_2x

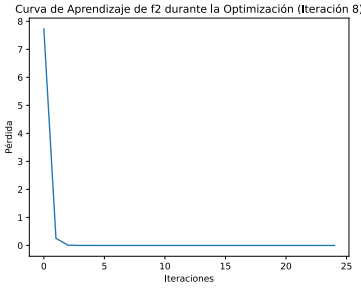
Subfigura	**Valor de α obtenido**
a	0,0
b	7,752358436584473
c	37774841348096,0



(a) Iteración 15



(b) Iteración 39



(c) Iteración 8

Figura 6: Curvas de aprendizaje en la función f_2 con Newton Rhapson

Finalmente, estos resultados demuestran que al usar el algoritmo Newton Rhapson con valores de α pequeños para la función f_2 se obtienen mejores resultado al evaluar el valor de perdida que tiene el algoritmo con las diversas funciones, por otra parte si usamos valores altos obtenemos resultados que crecen demasiado, sin embargo, al igual que con las otras 3 funciones vemos la rápida convergencia de este algoritmo.

3.2. Resultados obtenidos

Al terminar todas las ejecuciones se pudo recopilar la siguiente información: De los valores óptimos encontrados *(mínimos)* obtenemos la cantidad de pasos que requirió para converger y el punto donde convergió, así también se toma el número de la iteración donde sucedió. La siguiente tabla muestra los datos obtenidos:

f_i	**Número Iteración**	**Pasos en converger**	**Punto de Convergencia**	**Valor Óptimo**
f_0	3	12	[9.212443861e-25, 2.281108058e-23]	0.000000
f_1	1	9	[0.0014489889144, 0.9999685883522]	14.203123
f_2	3	0	[0.0426352024078, 1.0556936264038]	0.268635

Cuadro 9: Resultados obtenidos de 10 iteraciones del algoritmo.

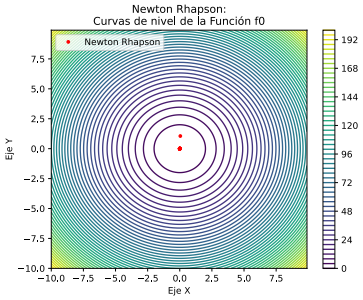
Posteriormente se analiza de la muestra de las diez ejecuciones el promedio de los valores óptimos, así como la cantidad promedio de pasos que dura cada algoritmo en converger. La siguiente tabla muestra los datos obtenidos:

f_i	**Óptimo Promedio**	**Pasos en Promedio**
f_0	2.363118e+00	12.9
f_1	1.145430e+05	10.1
f_2	1.216903e+23	0.0

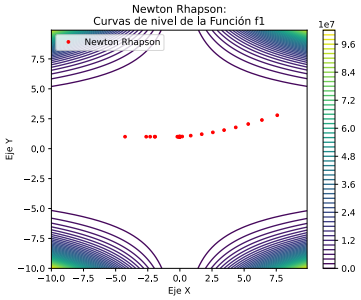
Cuadro 10: Valor óptimo promedio y cantidad promedio de pasos.

3.3. Puntos Visitados

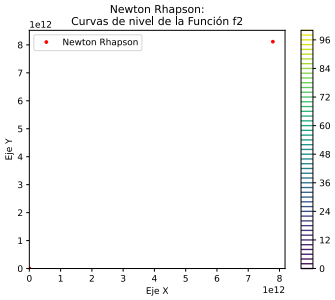
A continuación se muestran los puntos visitados en las mejores iteraciones del algoritmo Newton Rhapson para las funciones f_0, f_1, y f_2.



(a) Iteración 24



(b) Iteración 29



(c) Iteración 52

Figura 7: Curvas de nivel con los puntos visitados en f_0, f_1, f_2.

4. Algoritmo Descenso Del Gradiente

4.1. Gráficas de calibración de Optuna

Para llevar a cabo la calibración de los hiperparámetros del algoritmo de descenso de gradiente en su forma más simple y del descenso de gradiente adaptativo, hemos implementado un estudio utilizando el framework Optuna. En este estudio, evaluamos una variedad de valores de α para el caso del descenso de gradiente y ρ para el descenso de gradiente adaptativo, en los intervalos respectivos. Para las funciones f_0 y f_2, exploramos valores en el rango de $0 < \alpha < 5$. Sin embargo, en el caso de la función f_1, el intervalo de prueba para α se limita a $1 \times 10^{-10} < \alpha < 1 \times 10^{-5}$.

El proceso de calibración de hiperparámetros busca determinar el valor óptimo de α que permitirá al algoritmo de descenso de gradiente converger eficazmente en cada una de las funciones f_0, f_1 y f_2, teniendo en cuenta los rangos de valores específicos para α en cada caso.

Este enfoque permitirá encontrar una configuración óptima de α para cada función, optimizando así el rendimiento del algoritmo de descenso de gradiente en sus respectivos contextos.

4.1.1. Vanilla

Para el proceso de optimización del descenso del gradiente en su forma más simple, se determinó que los mejores óptimos obtenidos mediante Optuna para las funciones son los siguientes:

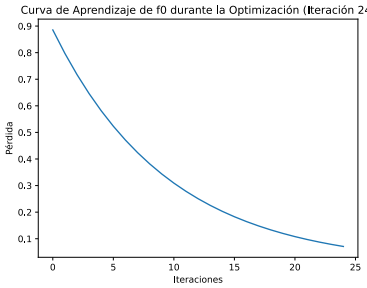
Cuadro 11: Valores de α para las funciones

Función	**Valor de α**
f_0	$\alpha = 0{,}5230717528153774$
f_1	$\alpha = 1 \times 9{,}999662286540719^{-06}$
f_2	$\alpha = 1{,}8448262935515587$

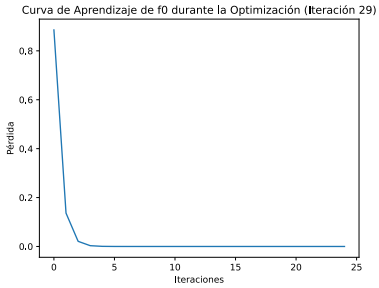
Para obtener los valores presentados en la Tabla 14, Optuna evaluó diversos valores de α dentro de los intervalos correspondientes. A través de la visualización de las curvas de aprendizaje de estos valores, podemos comprender cómo las funciones se optimizan a medida que se ajustan los hiperparámetros. A continuación se muestran algunos ejemplos de hiperparámetros probados por optuna:

Cuadro 12: Valores de α para las iteraciones

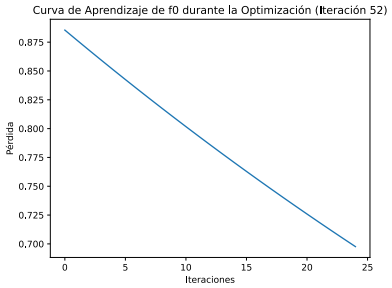
Subfigura	**Valor de α**
a	$\alpha = 0{,}4979299368215893$
b	$\alpha = 1{,}1911687750193$
c	$\alpha = 0{,}7868757807170943$



(a) Iteración 24



(b) Iteración 29



(c) Iteración 52

Figura 8: Curvas de aprendizaje en la función f_0 con el descenso del gradiente

Al analizar estas curvas de aprendizaje, podemos observar algunas características interesantes. En particular, destacamos la velocidad de convergencia de cada una de ellas. En la figura correspondiente a la iteración 24 (consulte la figura 8a), notamos un comportamiento suave y continuo a medida que la curva desciende. Sin embargo, en el gráfico de la iteración 52 (véase la figura 8c), observamos un comportamiento más bien lineal, caracterizado por una tendencia recta en la curva.

Es importante señalar que en estas iteraciones, la que más se acerca al óptimo es la iteración 29 (consulte la figura 8b). Esto se debe a un cambio abrupto en la curva, lo que resulta en un salto significativo hacia una mejora del rendimiento del algoritmo, pero aunque esta sea la que más se acerca, no es la mejor, a continuación se presentara la gráfica de la mejor iteración con el α mostrado en la Tabla 14.

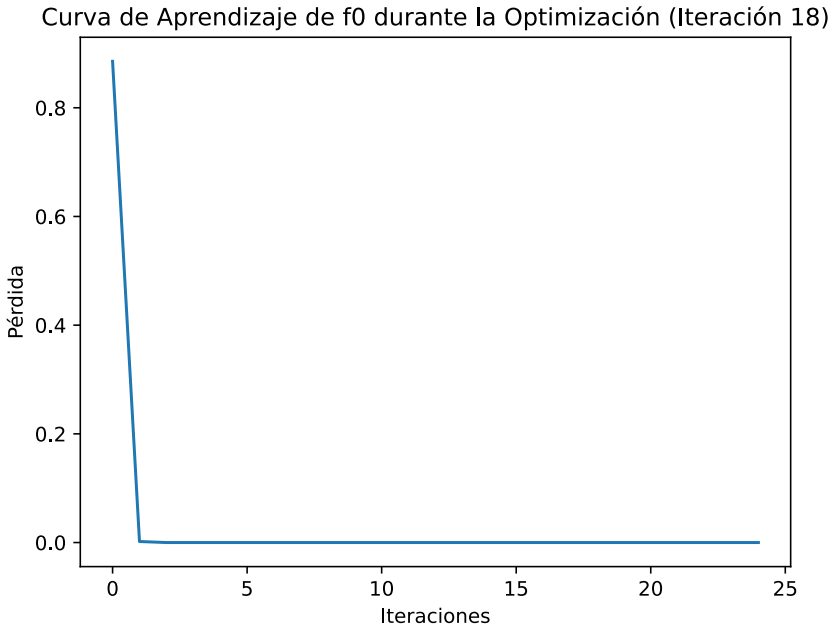


Figura 9: Curva de aprendizaje f_0 (Iteración 18)

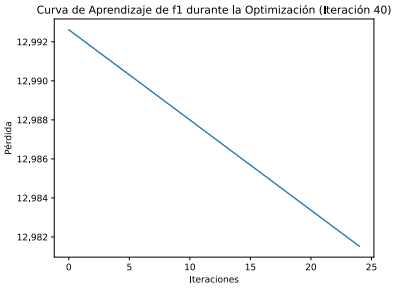
Como se puede apreciar en la figura 9, esta se destaca como la más sobresaliente de todas. La razón principal es el cambio drástico en su curva durante las primeras iteraciones, logrando acercarse al óptimo de manera casi instantánea. Estos indicios sugieren que los hiperparámetros asociados con esta iteración son los más adecuados encontrados por Optuna.

Este fenómeno es particularmente significativo, ya que no solo refleja la eficiencia de los hiperparámetros en la convergencia hacia una solución óptima, sino que también sugiere un posible camino hacia la aceleración del proceso de optimización.

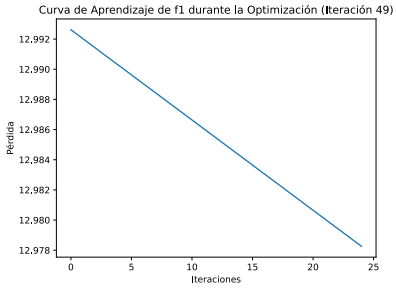
Para el proceso de nuestra segunda función f_1 tenemos las siguientes curvas de nivel:

Cuadro 13: Valores de α para las iteraciones

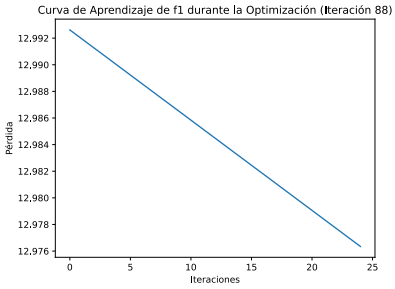
Subfigura	**Valor de α**
a	$\alpha = 1 \times 9{,}390504659542572^{-06}$
b	$\alpha = 1 \times 8{,}204671591215201^{-06}$
c	$\alpha = 1 \times 9{,}660747066249346^{-06}$



(a) Iteración 40



(b) Iteración 49



(c) Iteración 88

Figura 10: Curvas de aprendizaje en la función f_1 con el descenso del gradiente

Como se logra observar en las curvas de nivel mostradas en la figura 10, todas las curvas mantienen un comportamiento similar a lo largo de las iteraciones de optimización. Se caracterizan por una tendencia descendente que se asemeja a una línea recta a medida que avanzan las iteraciones.

Este comportamiento constante en las curvas de aprendizaje de las iteraciones sugiere que, en este contexto específico, los hiperparámetros probados pueden no estar optimizados de manera eficiente. Aunque las curvas muestran una disminución constante, el ritmo de mejora es limitado y lento en comparación con otros conjuntos de hiperparámetros. Este tipo de comportamiento

lineal puede indicar que se necesita una exploración aún más exhaustiva de las configuraciones de hiperparámetros.

El comportamiento visto en la figura 10 se replica en su mejor iteración, la cual se muestra a continuación:

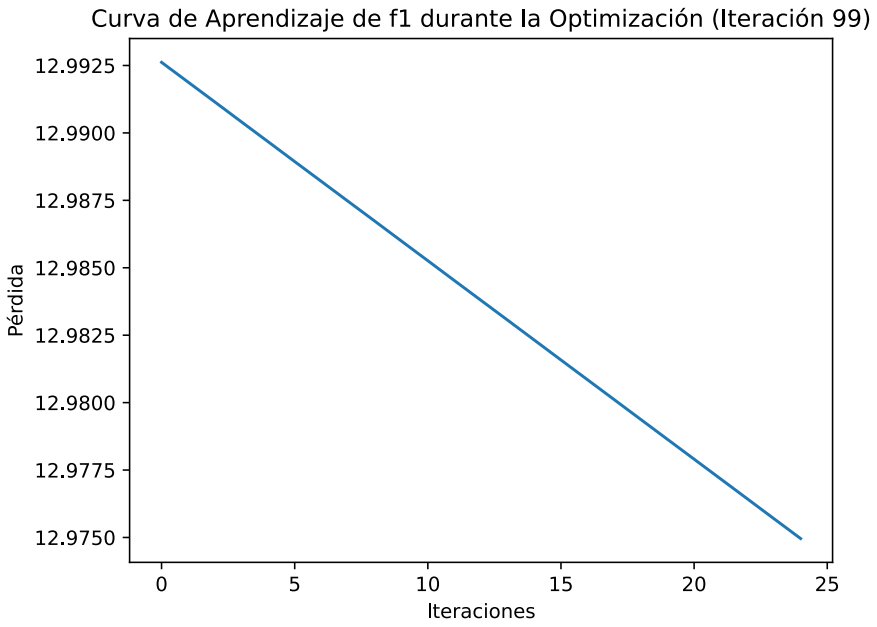


Figura 11: Curva de aprendizaje f_1 (Iteración 99)

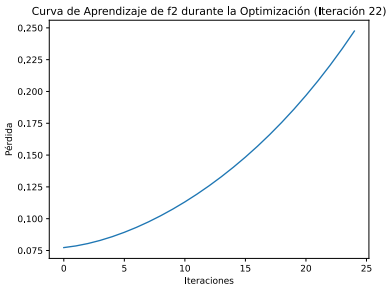
Al examinar detenidamente esta gráfica, podemos identificar un patrón de comportamiento lineal que se replica, al igual que en las gráficas anteriores de la figura 10. Sin embargo, se destaca una distinción notable: en el eje y, que representa la pérdida del modelo, notamos que alcanza un valor ligeramente menor en comparación con las otras curvas de aprendizaje.

Este descenso de la pérdida es modesto y sugiere que es probable que se requieran más iteraciones del algoritmo de descenso de gradiente para alcanzar un óptimo en este caso específico. El avance hacia la convergencia es notoriamente más lento en esta función en particular.

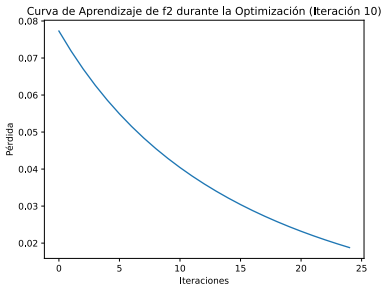
Para el proceso de nuestra segunda función f_2 tenemos las siguientes curvas de nivel:

Cuadro 14: Valores de α para las iteraciones

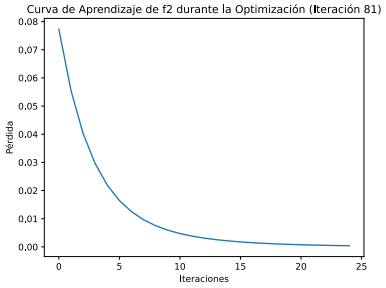
Subfigura	**Valor de α**
a	$\alpha = 1{,}9367574545044346$
b	$\alpha = 1{,}474658435940447$
c	$\alpha = 1{,}821983238229652$



(a) Iteración 22



(b) Iteración 10



(c) Iteración 81

Figura 12: Curvas de aprendizaje en la función f_2 con el descenso del gradiente

Al analizar detenidamente las gráficas presentadas en la figura 12, observamos una diversidad de comportamientos en función del valor de α utilizado. En particular, podemos notar el comportamiento de la gráfica en la figura 12a, donde, en lugar de disminuir para converger hacia el óptimo, la pérdida aumenta. Este comportamiento indica que el valor de α elegido en esta iteración particular es inadecuado y no conduce a la mejora del rendimiento del algoritmo.

Por otro lado, las gráficas en las figuras 12b y 12a muestran comportamientos más habituales, con una tendencia a la disminución de la pérdida a medida que avanzan las iteraciones. Sin embargo, incluso en estos casos, existen oportunidades de mejora para lograr una convergencia más eficiente y rápida hacia

el óptimo.

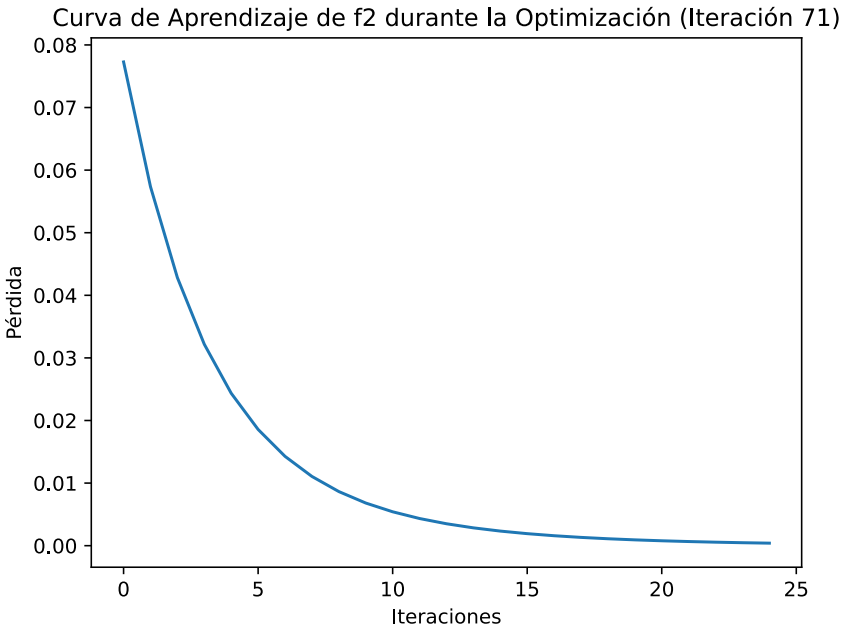


Figura 13: Curva de aprendizaje f_2 (Iteración 71)

Pronunciado como el que observamos en la figura 9 de la primera función. A pesar de la curvatura, esta curva de aprendizaje nos permite converger gradualmente hacia un valor óptimo.

La diferencia en la forma de estas curvas de aprendizaje entre las funciones resalta la influencia de los hiperparámetros y la naturaleza de las funciones mismas. Mientras que en la primera función observamos una convergencia más abrupta, en la segunda función, la optimización requiere un enfoque más gradual. Esto sugiere que el rendimiento y la eficiencia de la optimización varían según la función y la configuración de hiperparámetros.

4.1.2. Adagrad

Al realizar la optimización con optuna de para el hiperparámetro ρ para el descenso del gradiente adaptativo obtuvimos los siguientes valores óptimos para cada una de las funciones:

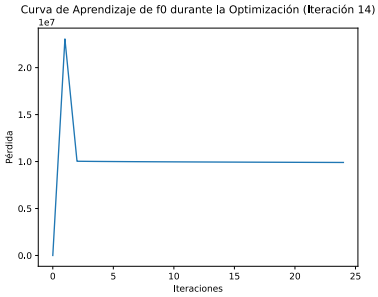
Cuadro 15: Valores de ρ para las funciones

Función	**Valor de ρ**
f_0	$\rho = 0{,}004982589525009401$
f_1	$\rho = 1 \times 4{,}345609289818026e^{-7}$
f_2	$\rho = 0{,}009793979523676201$

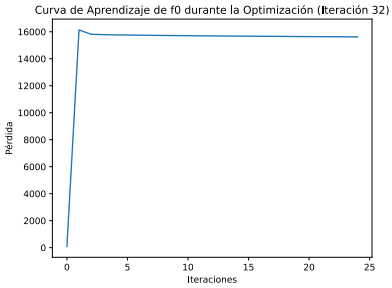
Para obtener los datos que figuran en la Tabla 16, Optuna examinó una variedad de valores para el parámetro ρ dentro de los intervalos específicos. Mediante la observación de las curvas de aprendizaje generadas para estos valores, podemos adquirir una comprensión más profunda de cómo las funciones se optimizan a medida que se ajustan los hiperparámetros. A continuación, se presentan ejemplos de algunos de los hiperparámetros que fueron evaluados por Optuna:

Cuadro 16: Valores de ρ para las iteraciones

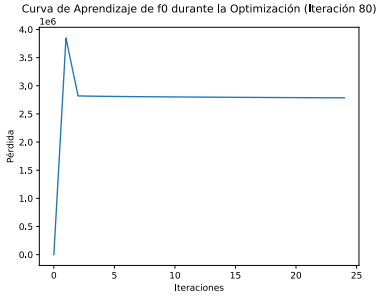
Subfigura	**Valor de ρ**
a	$\rho = 2{,}556340257793353$
b	$\rho = 0{,}07249812946454744$
c	$\rho = 1{,}0472297964272534$



(a) Iteración 14



(b) Iteración 32



(c) Iteración 80

Figura 14: Curvas de aprendizaje en la función f_0 con el descenso del gradiente adaptativo

Las gráficas representadas en la figura 14 ofrecen una perspicaz representación de cómo se comporta el algoritmo del descenso de gradiente. Al examinar estas curvas, podemos identificar un patrón interesante: inicialmente, la pérdida aumenta de manera exponencial, lo que indica una búsqueda exploratoria en el espacio de los parámetros. Sin embargo, esta fase de crecimiento se detiene de manera abrupta y el algoritmo comienza a retroceder, ajustándose en busca de una solución óptima.

Este fenómeno puede interpretarse como una adaptación del algoritmo para evitar caer en mínimos locales y permitir una exploración más eficaz del espacio de búsqueda. La transición desde un crecimiento exponencial a un descenso sugiere una estrategia que equilibra la exploración inicial con la explotación de regiones que muestran un potencial de mejora.

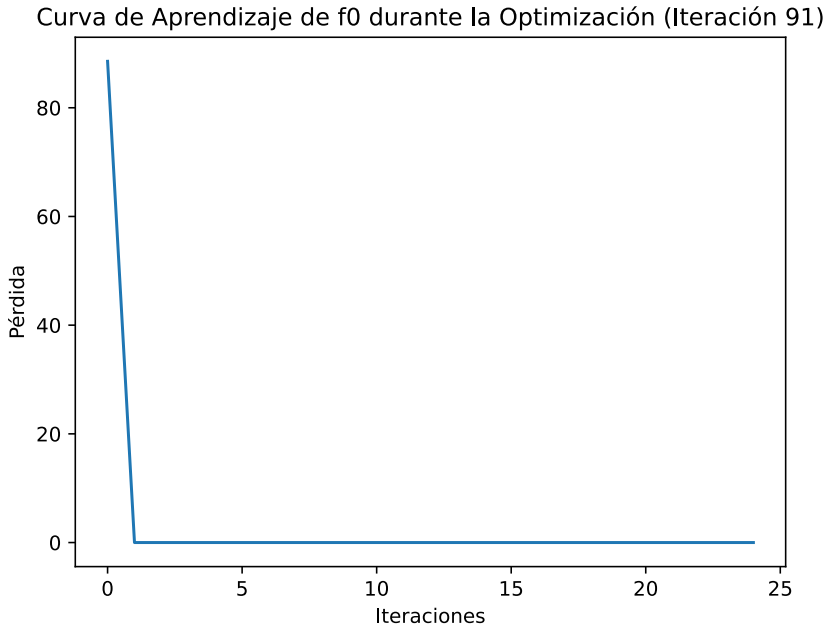


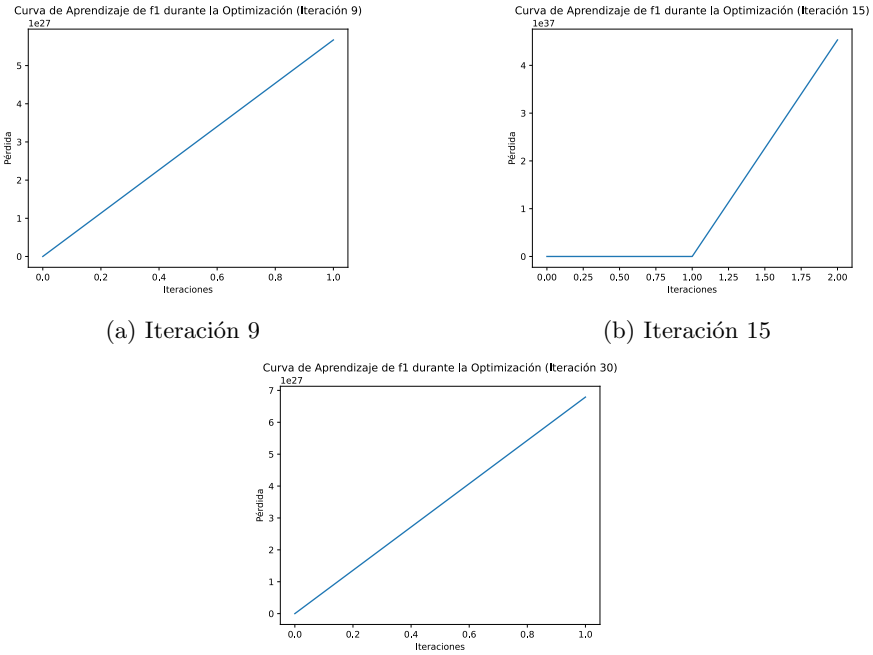
Figura 15: Curva de aprendizaje f_0 (Iteración 91)

La figura 15 nos brinda una visión del óptimo más destacado. Al observar detenidamente esta gráfica, se distingue un ángulo muy cerrado en su curva de aprendizaje. Este ángulo estrecho indica que el algoritmo ha alcanzado un nivel de convergencia muy eficiente y se encuentra cerca de una solución óptima. Esto es altamente indicativo de un rendimiento sobresaliente.

En comparación con los resultados presentados en la Tabla 16, esta gráfica en particular se destaca como una representación gráfica de un valor de hiperparámetro altamente efectivo. El ángulo cerrado en la curva de aprendizaje sugiere que el algoritmo ha encontrado una solución con una pérdida extremadamente baja y que la convergencia hacia el óptimo es excepcionalmente rápida.

Cuadro 17: Valores de ρ para las iteraciones

Subfigura	**Valor de ρ**
a	$\rho = 1 \times 7{,}85656931140569^{-6}$
b	$\rho = 1 \times 6{,}851981801127774^{-07}$
c	$\rho = 1 \times 8{,}034794240056727^{-06}$



(a) Iteración 9

(b) Iteración 15

(c) Iteración 30

Figura 16: Curvas de aprendizaje en la función f_0 con el descenso del gradiente adaptativo

La figura 16 nos presenta un comportamiento peculiar que se correlaciona con los valores proporcionados en la Tabla 17. Al examinar detenidamente esta gráfica de aprendizaje, notamos un patrón inusual en la curva de pérdida. Este patrón inusual sugiere que los valores de los hiperparámetros, como se detallan en la Tabla 17, están generando un comportamiento atípico en el proceso de optimización, además este comportamiento se repite a lo largo de todas las pruebas hechas con optuna, excepto en algunos casos puntuales.

En lugar de converger hacia un mínimo óptimo, la curva de pérdida tiende a crecer de manera continua a medida que avanzan las iteraciones. Este comportamiento inusual, en el que los valores tienden hacia el infinito, indica una dificultad en la convergencia del algoritmo. Es probable que los valores de hiperparámetros elegidos no sean adecuados para esta función específica, lo que

resulta en un rendimiento subóptimo.

Después de llevar a cabo las 100 iteraciones con Optuna para esta función, el hiperparámetro óptimo produce la siguiente curva de aprendizaje:

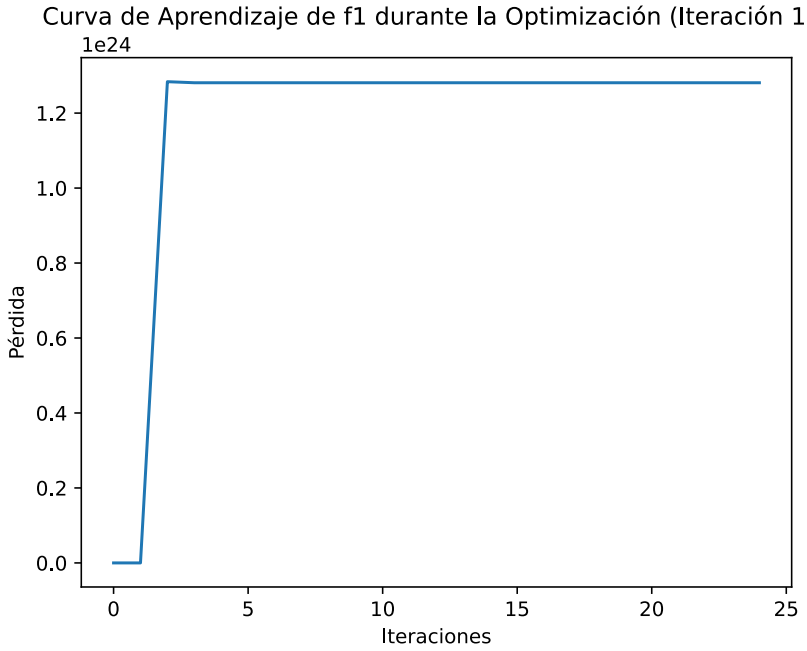


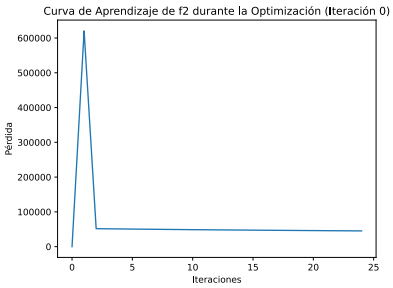
Figura 17: Curva de aprendizaje f_1 (Iteración 12)

La curva de aprendizaje que se puede observar en la imagen 17 ofrece un comportamiento interesante. Aunque no alcanza un estado de perfección, presenta una característica notable: se mantiene estable en un punto específico a medida que avanzan las iteraciones. Esto significa que, a pesar de que la pérdida no disminuye de manera continua, alcanza un equilibrio y se estabiliza en un valor constante.

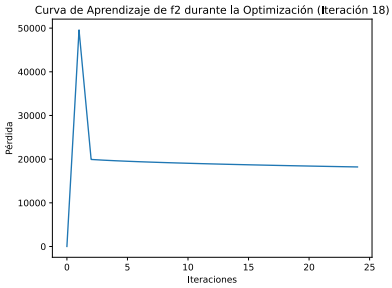
Este comportamiento puede ser interpretado como un signo de que el algoritmo ha alcanzado un valle en su proceso de optimización. Aunque no llega a un óptimo absoluto, la estabilidad en un punto fijo sugiere que ha encontrado una solución aceptable y que no experimenta fluctuaciones significativas en la pérdida.

Cuadro 18: Valores de ρ para las iteraciones

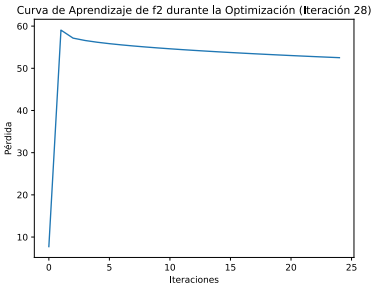
Subfigura	**Valor de ρ**
a	$\rho = 3{,}1717199733860775$
b	$\rho = 0{,}9037899321680318$
c	$\rho = 0{,}009793979523676201$



(a) Iteración 0



(b) Iteración 18



(c) Iteración 28

Figura 18: Curvas de aprendizaje en la función f_2 con el descenso del gradiente adaptativo

La figura 18 nos brinda una perspicaz visión del comportamiento de las curvas de aprendizaje en la función f_2. En esta función en particular, se observa un patrón recurrente en las curvas de aprendizaje, que revela una estrategia distintiva del algoritmo.

En las primeras etapas de optimización, el algoritmo prioriza la exploración del espacio de parámetros, lo que se manifiesta en un crecimiento inicial de la pérdida. Esta fase de exploración es esencial para descubrir posibles regiones óptimas en el espacio de parámetros y evitar quedarse atrapado en mínimos locales.

Posteriormente, las curvas muestran un cambio de dirección, indicando una transición hacia la explotación de las regiones prometedoras. En esta fase, el algoritmo comienza a ajustar los parámetros de manera más precisa, lo que

conduce a una disminución en la pérdida y, en última instancia, a una convergencia hacia un óptimo.

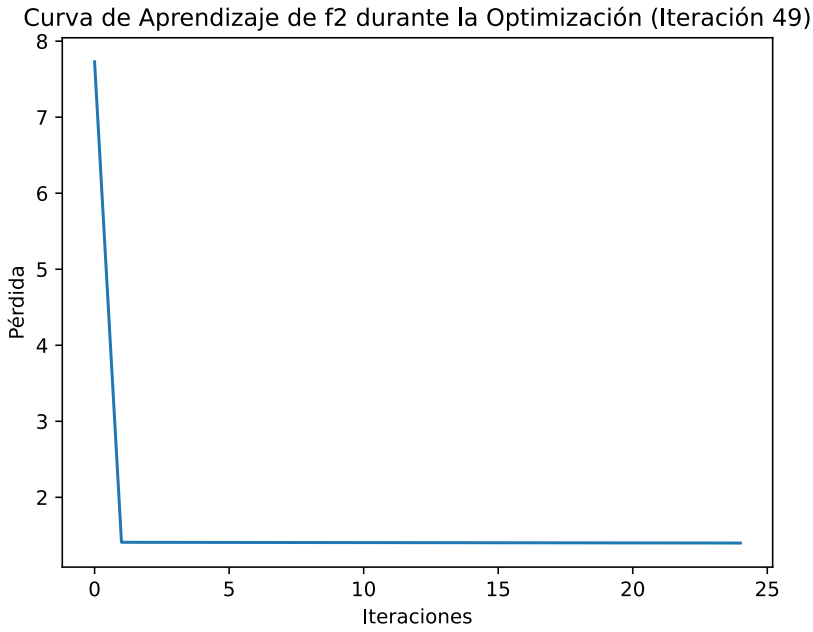


Figura 19: Curva de aprendizaje f_2 (Iteración 49)

La figura 19 nos revela que el algoritmo logro la identificación de un punto óptimo altamente efectivo. Este logro se obtiene con los valores presentes en la Tabla 16, y podemos apreciarlo claramente en la representación gráfica.

En la figura, notamos un ángulo casi recto en la curva de aprendizaje, lo que indica que el algoritmo ha logrado una convergencia eficaz hacia un mínimo óptimo. Esta convergencia es especialmente significativa, ya que sugiere que el algoritmo ha alcanzado una solución altamente eficiente.

Este ángulo recto es un indicativo de que la pérdida disminuye de manera constante y significativa a medida que avanzan las iteraciones, lo que implica una convergencia rápida hacia un óptimo. El algoritmo ha logrado ajustar los hiperparámetros de manera precisa, lo que se traduce en un rendimiento excepcional.

4.2. Adagrad y Decenso Del Gradiente en puntos sillas

¿Por qué el algoritmo Adagrad es más efectivo en evitar atascarse en puntos silla que el algoritmo del descenso del gradiente?

El algoritmo Adagrad es más efectivo en evitar atascarse en puntos silla en comparación con el algoritmo del descenso del gradiente debido a una característica clave de Adagrad. La adaptación de la tasa de aprendizaje a cada parámetro individual durante el proceso de optimización. Esto proporciona una ventaja significativa en situaciones donde existen puntos silla en la función objetivo.

Adagrad ajusta la tasa de aprendizaje para cada parámetro en función de su historia de gradientes pasados. Esto significa que los parámetros que tienen gradientes grandes recibirán tasas de aprendizaje más pequeñas, lo que permite un refinamiento más cuidadoso en su vecindad.

El algoritmo del descenso del gradiente con una tasa de aprendizaje constante puede tener dificultades en puntos silla porque continuará avanzando incluso cuando el gradiente es pequeño o cercano a cero. Esto puede hacer que el algoritmo quede atrapado en el punto silla sin hacer un progreso significativo. En contraste, Adagrad es más sensible a los gradientes pequeños y disminuirá la tasa de aprendizaje en consecuencia, lo que facilita la salida de puntos silla.

Adagrad se adapta a la topografía de la función objetivo al ajustar las tasas de aprendizaje de manera individual y automática. Esto permite al algoritmo "sentir"la estructura de la función y responder en consecuencia. Cuando se encuentra en un punto silla, Adagrad ajusta las tasas de aprendizaje para permitir una exploración más minuciosa y una posible salida del punto silla.

Como ejemplo ilustrativo de la adaptabilidad del Adagrad podemos observar la Figura 20 en sus sub figuras, como el algoritmo en su versión más básica sale fuera de la región graficada de la función, no siendo así cuando observamos la Figura 21 en la cual podemos observar como se mantiene dentro del gráfico y tiende a moverse de una forma más atenuada debido a la adaptabilidad automática del descenso del gradiente adaptativo.

4.3. Resultados obtenidos

Al terminar todas las ejecuciones se pudo recopilar la siguiente información: De los valores óptimos encontrados *(mínimos)* obtenemos la cantidad de pasos que requirió para converger y el punto donde convergió, así también se toma el número de la iteración donde sucedió. La siguiente tabla muestra los datos obtenidos:

4.3.1. Vanilla

f_i	**Número Iteración**	**Pasos en converger**	**Punto de Convergencia**	**Valor Óptimo**
f_0	0	18	[3.309031683e-24, 7.790474755e-24]	0.0000e+00
f_1	0	0	[3.6787831783294, 8.6610193252563]	5.7836e+06
f_2	0	0	[3.6787831783294, 8.6610193252563]	7.7283e+00

Cuadro 19: Resultados obtenidos de 10 iteraciones del algoritmo.

Posteriormente se analiza de la muestra de las diez ejecuciones el promedio de los valores óptimos, así como la cantidad promedio de pasos que dura cada algoritmo en converger. La siguiente tabla muestra los datos obtenidos:

f_i	**Óptimo Promedio**	**Pasos en Promedio**
f_0	3.549425e+00	18.0
f_1	inf	0.0
f_2	1.172966e+12	0.0

Cuadro 20: Valor óptimo promedio y cantidad promedio de pasos.

4.3.2. Adagrad

f_i	**Número Iteración**	**Pasos en converger**	**Punto de Convergencia**	**Valor Óptimo**
f_0	3	0	[0.04263520240783, 1.0556936264038]	1.116307
f_1	9	1	[6.22164201736450, 0.8096880912780]	0.197772
f_2	3	0	[0.04263520240783, 1.0556936264038]	0.268635

Cuadro 21: Resultados obtenidos de 10 iteraciones del algoritmo.

Posteriormente se analiza de la muestra de las diez ejecuciones el promedio de los valores óptimos, así como la cantidad promedio de pasos que dura cada algoritmo en converger. La siguiente tabla muestra los datos obtenidos:

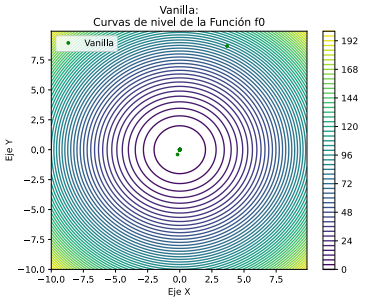
f_i	**Óptimo Promedio**	**Pasos en Promedio**
f_0	2.648251e+04	0.0
f_1	3.325567e+34	7.5
f_2	1.338127e+31	0.0

Cuadro 22: Valor óptimo promedio y cantidad promedio de pasos.

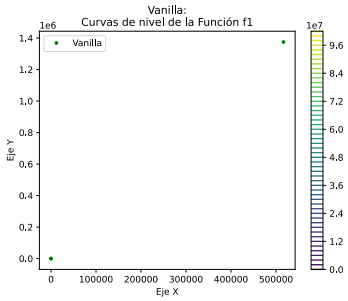
4.4. Puntos Visitados

4.4.1. Vanilla

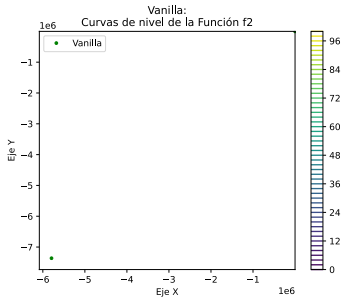
A continuación se muestran los puntos visitados en las mejores iteraciones del algoritmo del Descenso del Gradiente para las funciones f_0, f_1, y f_2.



(a) Iteración 24



(b) Iteración 29

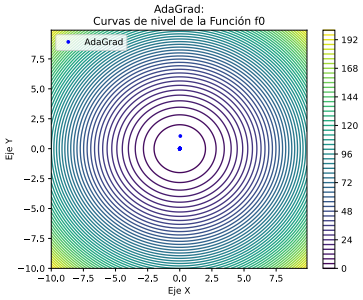


(c) Iteración 52

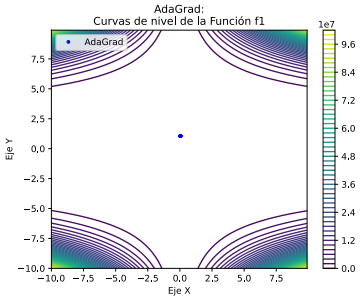
Figura 20: Curvas de nivel con los puntos visitados en f_0, f_1, f_2.

4.4.2. Adagrad

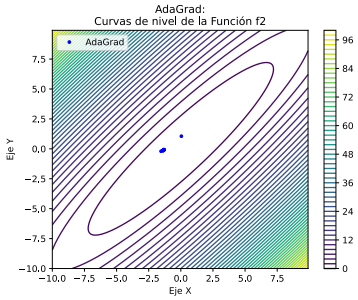
A continuación se muestran los puntos visitados en las mejores iteraciones del algoritmo del Descenso del Gradiente Adaptativo para las funciones f_0, f_1, y f_2.



(a) Iteración 24



(b) Iteración 29



(c) Iteración 52

Figura 21: Curvas de nivel con los puntos visitados en f_0, f_1, f_2.

4.5. Análisis comparativo Adagrad y Descenso Del Gradiente

El descenso de gradiente, en sus diversas variantes, es el corazón de muchos algoritmos de optimización en el aprendizaje automático y la optimización numérica en general. Dos enfoques ampliamente utilizados son el descenso de gradiente vanilla o básico y el descenso de gradiente adaptativo. A continuación, comparamos estos dos enfoques desde diferentes perspectivas:

Tasa de aprendizaje:

Vanilla

En la versión más básica del descenso del gradiente podemos observar que cuenta con una tasa de aprendizaje fija, la cual se mantiene a lo largo de todo el proceso, esta taza de aprendizaje se configura muchas veces de manera empírica, lo que puede causar diversos efectos en el algoritmo.

Descenso de Gradiente Adaptativo

El descenso de gradiente adaptativo ajusta de manera automática la tasa de aprendizaje durante la optimización. Cada uno de los parámetros tiene su propia tasa de aprendizaje, que se adapta según el historial de gradientes que han pasado por ese parámetro, lo que de alguna forma hace que el algoritmo pueda "sentir" o "recordar" la forma que tiene el terreno a su alrededor, lo que permite una adaptación en tiempo real de los parámetros.

Comportamiento en Puntos Silla:

Vanilla

Puede tener dificultades en puntos silla, ya que sigue avanzando incluso cuando el gradiente es pequeño o cercano a cero. Esto puede llevar a quedar atascado en puntos silla sin hacer un progreso significativo.

Descenso de Gradiente Adaptativo

Tiende a lidiar mejor con puntos silla. Ajusta las tasas de aprendizaje para los parámetros individuales, lo que permite una salida más fácil de puntos silla y la convergencia hacia mínimos locales o globales.

Convergencia y Velocidad:

Vanilla

Puede converger eficazmente si se selecciona una tasa de aprendizaje adecuada y si la función objetivo es convexa o suave. Sin embargo, puede ser sensible a la elección de la tasa de aprendizaje y puede ser lento en converger en problemas con superficies de pérdida complejas. Un ejemplo de esto se puede ver en las Figuras 9 y 13 en la primera figura (9) se puede observar como logra llegar de manera perfecta al punto óptimo, en solo dos pasos, y utilizando 100 iteraciones del framework optuna para calibrar el hiperparámetro α, pero sí vemos la segunda figura (13) nos damos cuenta de que a pesar de realizar la calibración pertinente para la función f_2 le cuesta un poco más llegar a ese óptimo, y esto se ve por la forma que existe en la curva de aprendizaje.

Descenso de Gradiente Adaptativo

Tiende a converger más rápido en problemas con superficies de pérdida irregulares o no convexas. La adaptación automática de la tasa de aprendizaje permite una convergencia eficiente. Esto lo podemos ver reflejado en las Figuras 19 y

15 donde en comparación con la versión básica el algoritmo Adargad en ambas ocasiones, con 100 iteraciones del framework optuna para calibrar su hiperparámetro, ρ logra llegar al punto óptimo de forma más rápida.

Recomendada para funciones:

Vanilla

Adecuado para problemas simples y suaves

Descenso de Gradiente Adaptativo

Efectivo en problemas complejos y con superficies de pérdida irregulares

5. Algoritmo Simulated Annealing

5.1. Gráficas de calibración de Optuna

5.1.1. Enfoque

Para el algoritmo de Simulated Annealing se realizó un proceso de calibración utilizando el framework optuna, en este caso se ha buscado obtener los mejores valores para los siguientes hiperparámetros:

- α: Representa el cambio en la temperatura en el algoritmo y controla la disminución gradual de la temperatura durante la búsqueda.
- T_t: Representa la temperatura inicial en el algoritmo, por lo que determina la probabilidad de aceptar soluciones subóptimas en las primeras etapas de la búsqueda.
- h: Representa el valor que determina el tamaño del espacio de búsqueda y se calcula aplicando la siguiente fórmula:

$$\left[\vec{X}_t - h, \vec{X}_t + h\right]$$

5.1.2. Mejores hiperparámetros

Una vez finalizado el proceso de calibración de hiperparámetros se han obtenido los valores que se presentan en la siguiente tabla:

Cuadro 23: Valores de α para las funciones

Función	**Intervalo de α**
f_0	$\alpha = 0{,}8599468781832491$
f_1	$\alpha = 0{,}8712508926417811$
f_2	$\alpha = 0{,}8500076641449615$

Cuadro 24: Valores de T_t inicial para las funciones

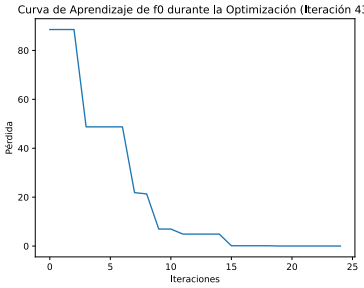
Función	**Intervalo de α**
f_0	$T_t = 2{,}2735197480829803$
f_1	$T_t = 1{,}1046845878590525$
f_2	$T_t = 6{,}283942569628704$

Cuadro 25: Valores de h para las funciones

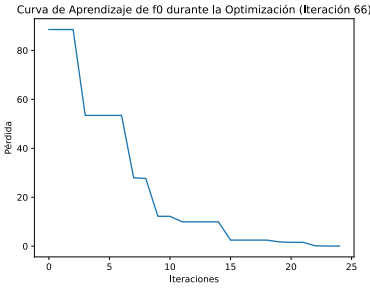
Función	**Intervalo de α**
f_0	$h = 2{,}544484099970395$
f_1	$h = 3{,}632011640203967$
f_2	$h = 3{,}1829746414563966$

Los valores presentados en las tablas anteriores han sido generados por Optuna, basándose en los resultados obtenidos al evaluar el comportamiento del algoritmo simulated annealing con diferentes combinaciones de parametros. A continuación, se presenta el comportamiento de las funciones de aprendizaje con 3 iteraciones respaldas con su grafica de aprendizaje. Para comprender de mejor las iteraciones es importante considerar el siguiendo el patrón:

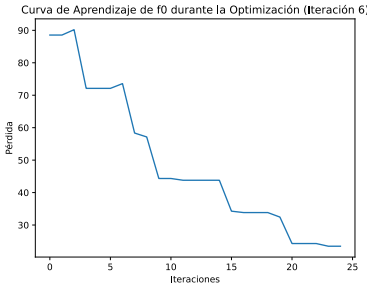
- a: Mejor Iteración lograda
- b: Iteración comparativa intermedia
- c: Peor Iteración completada



(a) Iteración 43



(b) Iteración 66



(c) Iteración 6

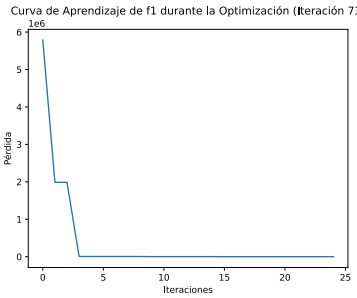
Figura 22: Curvas de aprendizaje en la función f_0 con Simulated Annealing

En la figura anterior se puede observar el comportamiento del algoritmo con la función f_0 en tres iteraciones distintas con la siguiente combinación de parámetros:

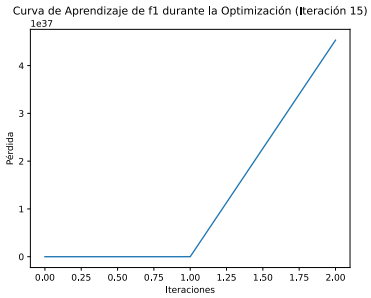
Cuadro 26: Hiperparámetros en gráficas para f_0

Subfigura	α	T_t	h
a	0.919776318717975	1.262224408290402	2.5447376535344564
b	0.9106894926527385	1.9676680426140905	3.390261479713804
c	0.9516696799464767	3.0933712848473984	1.0537125576587942

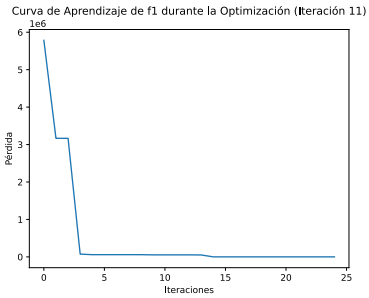
Se puede observar que los valores que más cambian al comparar la mejor iteración con la peor son la temperatura inicial y el criterio del espacio de búsqueda, se nota una convergencia más rápida y un mayor acercamiento en las funciones a y b en contraste con el comportamiento de la iteración c.



(a) Iteración 73



(b) Iteración 15



(c) Iteración 11

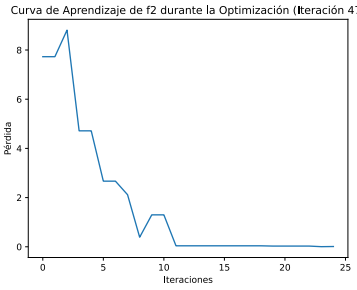
Figura 23: Curvas de aprendizaje en la función f_1 con Simulated Annealing

La figura anterior muestra el comportamiento del algoritmo con la función f_1 en tres iteraciones distintas con la siguiente combinación de parámetros:

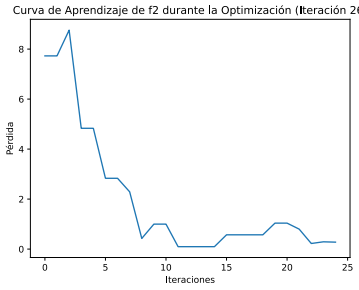
Cuadro 27: Hiperparámetros en gráficas para f_1

Subfigura	α	T_t	h
a	0.9115918489368464	0.9115918489368464	5.123975289039711
b	0.8500389537922496	4.708755060275312	2.8952818221213925
c	0.9547335226710361	1.0323097514199002	2.825825818870995

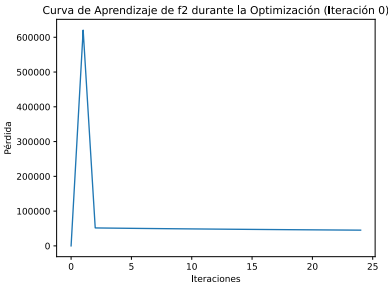
La información que se puede observar de las gráficas anteriores demuestran que el cambio más significativo de la mejor iteración con respecto a las otras dos es la elección de un h mayor para poder tomar un nuevo punto aleatorio en un espacio más amplio.



(a) Iteración 47



(b) Iteración 26



(c) Iteración 0

Figura 24: Curvas de aprendizaje en la función f_2 con Simulated Annealing

Las tres graficas anteriores han sido obtenidas con la siguiente combinación de hiperparámetros.

Cuadro 28: Hiperparámetros en gráficas para f_2

Subfigura	α	T_t	h
a	0.8979283590288949	1.2741889304493768	3.1579854125837747
b	0.90490702875916	1.5989435605575526	3.0171093365838533
c	0.9587569062050809	4.240369398551003	1.7408651680274425

Es notable que la diferencia más marcada en los datos de la tabla anterior es la elección del valor α para determinar la disminución de temperatura, vemos que un valor alto influyó negativamente en la gráfica c dando por resultado la peor iteración del algoritmo aplicado a esta función, los resultados de la tabla anterior pueden ser consultados en la siguiente tabla.

5.2. Resultados obtenidos

Haciendo uso de los hiperparámetros la sección 5.1 y los puntos utilizados con los algortimos anteriores se ejecutará Simulated Annealing 10 veces para cada una de las funciones.

f_i	**Número Iteración**	**Pasos en converger**	**Punto de Convergencia**	**Valor Óptimo**
f_0	5	21	[0.1601684093475, -0.1697366237640]	0.054464
f_1	3	18	[2.4034957885742, 0.2915222644805]	0.122704
f_2	4	22	[0.8171668052673, 1.0316667556762]	0.045684

Cuadro 29: Resultados obtenidos de 10 iteraciones del algoritmo.

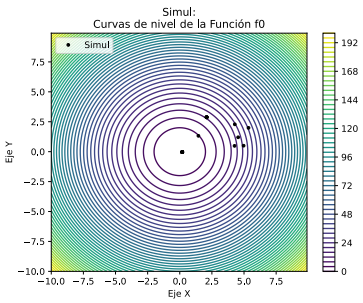
Posteriormente se analiza de la muestra de las diez ejecuciones el promedio de los valores óptimos, así como la cantidad promedio de pasos que dura cada algoritmo en converger. La siguiente tabla muestra los datos obtenidos:

f_i	**Óptimo Promedio**	**Pasos en Promedio**
f_0	15.972546	16.4
f_1	211083.468055	19.3
f_2	1.597155	17.1

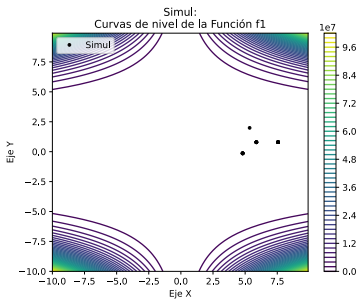
Cuadro 30: Valor óptimo promedio y cantidad promedio de pasos.

5.3. Puntos Visitados

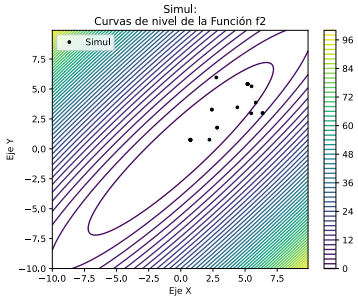
A continuación se muestran los puntos visitados en las mejores iteraciones del algoritmo Simulated annealing para las funciones f_0, f_1, y f_2.



(a) Iteración 24



(b) Iteración 29



(c) Iteración 52

Figura 25: Curvas de nivel con los puntos visitados en f_0, f_1, f_2.

5.4. Simulated annealing y descenso del gradiente

El algoritmo de descenso de gradiente es uno de los métodos más utilizados para identificar valores óptimos. Sin embargo, es susceptible de quedar atrapado en mínimos locales o puntos de silla, especialmente cuando se trata de problemas de optimización no convexos multidimensionales.

Por otro lado, el algoritmo de simulated annealing es capaz de explorar el espacio de búsqueda de manera más efectiva, evitando quedar atrapado en estos puntos conflictivos gracias a su componente probabilístico, que permite seleccionar otro punto en función de una probabilidad que disminuye con el tiempo.

Al combinar el rápido recorrido del descenso de gradiente, con la capacidad probabilística de simulated annealing es posible obtener un algoritmo que converja rápidamente y evite quedar atrapado en puntos no deseados, tal y como

es propuesto en el artículo de Cai (2021).

Donde se describe la implementación de un algoritmo denominado SA-GD (descenso de gradiente mejorado por recocido simulado) que agrega un componente probabilístico al algoritmo de descenso de gradiente, similar al usado en simulated annealing, de esta manera es probable que el algoritmo salga de un punto indeseado y converja con mayor seguridad.

6. Comparación de todos los algoritmos

En el campo de la optimización, la selección del algoritmo adecuado desempeña un papel crucial en el logro de resultados óptimos. En esta comparativa, exploraremos y analizaremos cuatro algoritmos ampliamente utilizados: Simulated Annealing, Descenso de Gradiente Vanilla, Adagrad y el método de Newton-Raphson. Cada uno de estos enfoques tiene sus propias características, ventajas y desventajas, lo que hace que sean idóneos para diferentes tipos de problemas y situaciones. En esta comparativa, examinaremos el rendimiento de estos algoritmos en diversas funciones definidas previamente como f_0, f_1 y f_2, lo que nos permitirá comprender mejor cómo se comparan en términos de eficiencia y eficacia. Esta comparativa arrojará luz sobre las fortalezas y debilidades de cada enfoque, proporcionando información valiosa para la elección adecuada de algoritmos en aplicaciones del mundo real.

Cuando evaluamos la velocidad con la que estos algoritmos alcanzan una solución, es esencial analizar detenidamente las tablas de resultados correspondientes a los algoritmos: Newton-Raphson, Descenso de Gradiente Vanilla, Adagrad y Simulated Annealing, identificadas como 9, 19, 21 y 29, respectivamente.

Observamos que Adagrad se destaca como el algoritmo más veloz para llegar a una solución en la mayoría de los casos. Esto se debe a su capacidad para adaptarse eficazmente a la topografía de la función, como lo menciona Ruder (2016),"su capacidad para adaptar la tasa de aprendizaje a los parámetros del modelo. Como indica Ruder, Adagrad realiza actualizaciones más grandes para los parámetros menos frecuentes y actualizaciones más pequeñas para los parámetros más frecuentes." Lo que hace que el algoritmo logre actualizar los movimientos de manera más inteligente.

En contraste, Simulated Annealing se revela como el algoritmo más lento, ya que generalmente requiere un número significativamente mayor de iteraciones, que oscilan entre 16 y 17, para alcanzar la convergencia. La razón detrás de esta diferencia radica en el enfoque fundamental de Simulated Annealing. Este algoritmo se basa en la exploración probabilística de la función objetivo, lo que implica visitar puntos aleatorios en la función y moverse de manera más errática por su superficie, lo que puede llegar a ser muy beneficioso si se tiene una función compleja, no diferenciable o caótica como lo indica Cai (2021)" Con la capacidad de montar colinas, se supone que el modelo salta de estas áreas locales intratables y converge a un estado óptimo." Aunque esto puede tener un costo y puede llevar más tiempo para encontrar una solución óptima, como lo menciona Busetti (2003) "Existe un claro equilibrio entre la calidad de las soluciones y el tiempo necesario para calcularlas." Esta desventaja es una de las principales para tomar en cuenta a la hora de escoger este algoritmo en modelos muy grandes como en las redes neuronales, este puede ser una mala opción si buscamos un optimizador que sea eficiente a nivel de tiempo computacional.

Las formas de movimiento anteriormente descritas también la podemos observar en las figuras 21 y 25 donde podemos los diferentes movimientos dentro de las figuras, y aunque el Adagrad se mueva de manera lenta, puede llegar ser extremadamente lenta y que el proceso de optimizado sea lento, según (Ruder,

2016), “Adagrad puede experimentar una ralentización en la tasa de aprendizaje durante el entrenamiento.” lo que es una razón de peso a la hora de escoger este algoritmo.

Si nos adentramos en el análisis de los algoritmos intermedios, encontramos que, aunque no sean los más rápidos en alcanzar la convergencia, poseen propiedades interesantes que merecen nuestra atención. Estas propiedades se hacen evidentes al examinar las curvas de aprendizaje presentadas en las figuras representadas en 5, las cuales revelan que la forma en que el algoritmo Newton-Raphson converge es distintas funciones, es más estable en comparación con el algoritmo de Descenso de Gradiente Vanilla, como se ilustra en la figura 10, donde la convergencia tiene una forma lineal.

Esta estabilidad es una de las principales ventajas de estos algoritmos, tal como se menciona en Akram y Ann (2015), donde se destaca que “este método es ampliamente conocido por su rápida velocidad de convergencia y por mejorar la propiedad de convergencia.” Sin embargo, es importante señalar que en ciertos casos, como el que involucra la función f_1, este algoritmo presenta un comportamiento peculiar que no siempre es beneficioso. Este fenómeno se puede apreciar en la figura 7c, donde el algoritmo llega a un punto completamente distante, incluso fuera del intervalo en el que se graficaron las curvas de nivel.

Este comportamiento, que implica que los puntos no caigan dentro de las áreas sombreadas de las curvas de nivel representadas, también se hace evidente en la Figura 20, donde dos de las sub figuras están mayoritariamente en blanco. Esta observación se debe a una limitación del algoritmo de Descenso de Gradiente en su forma más básica, como lo señala IBM (2023) “En problemas convexos, donde la función de pérdida tiene una única solución óptima, el Descenso de Gradiente suele funcionar eficazmente y puede encontrar el mínimo global con relativa facilidad. No obstante, en problemas no convexos, caracterizados por funciones de pérdida que pueden tener múltiples mínimos locales, el Descenso de Gradiente puede enfrentar desafíos significativos al intentar encontrar el mínimo global”. Esto se refleja en la dificultad que encuentra el algoritmo en abordar estas situaciones y su tendencia a quedar atrapado en mínimos locales en lugar de alcanzar la solución óptima global.

De manera similar, el desempeño de este algoritmo se destaca en la optimización de funciones que no sean especialmente complejas, como se ilustra en la Figura 20a. En casos donde la función a optimizar es relativamente sencilla, el algoritmo demuestra su eficacia, como menciona Donges (2021), quien señala que .[es]te enfoque es capaz de generar un gradiente de error estable y una convergencia igualmente estable..[A]demás, su comprensión y aplicación resultan bastante accesibles, lo que facilita su implementación en diversas situaciones.

En conclusión, esta comparativa detallada de cuatro algoritmos de optimización clave, Simulated Annealing, Descenso de Gradiente Vanilla, Adagrad y el método de Newton-Raphson, ha proporcionado una visión integral de sus fortalezas y limitaciones en una variedad de contextos. La elección del algoritmo adecuado debe basarse en la naturaleza del problema y en los objetivos específicos. Adagrad se destaca por su rapidez y adaptabilidad en funciones bien comportadas, mientras que Simulated Annealing brilla en la exploración de so-

luciones en problemas complejos y no lineales, a pesar de su relativa lentitud.

Los algoritmos intermedios, como el método de Newton-Raphson, ofrecen estabilidad en la convergencia en la mayoría de los casos, pero pueden mostrar comportamientos inusuales en situaciones particulares. El Descenso de Gradiente Vanilla, aunque eficiente en funciones simples y convexas, puede enfrentar dificultades en problemas no convexos con múltiples mínimos locales.

En última instancia, esta comparativa subraya la importancia de comprender las características intrínsecas de cada algoritmo y su idoneidad para un contexto particular. La selección adecuada del algoritmo puede marcar la diferencia en la eficiencia y efectividad de la optimización en aplicaciones del mundo real. Este análisis proporciona un recurso valioso para los profesionales que buscan abordar una amplia gama de problemas de optimización con mayor conocimiento y precisión.

Referencias

Akram, S., y Ann, Q. U. (2015). Newton raphson method. *International Journal of Scientific & Engineering Research*, *6*(7), 1748–1752.

Busetti, F. (2003). Simulated annealing overview. *World Wide Web URL www. geocities. com/francorbusetti/saweb. pdf*, *4*.

Cai, Z. (2021). Sa-gd: Improved gradient descent learning strategy with simulated annealing. *arXiv preprint arXiv:2107.07558*.

Donges, N. (2021, 7). *Gradient Descent in Machine Learning: A Basic Introduction.* Descargado de https://builtin.com/data-science/gradient-descent

IBM. (2023, 31 de October). *What is gradient descent?* Descargado de https://www.ibm.com/topics/gradient-descent (Retrieved October 31, 2023)

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.