Práctica 1 - Memoria

Sergio Quijano Rey

April 3, 2021

Contents

1 Ejercicio 1 - Búsqueda iterativa de óptimos

1.1 Apartado 1

En este apartado se nos pide implementar el algoritmo de Grandiente Descendente. Esta función toma como parámetros de entrada:

- starting_point : punto inicial del que parte la búsqueda
- loss_function : función de error que buscamos minimizar. En nuestro caso concreto, es una función real de dos variables
- gradient: función vectorial 2-dimensional que representa el gradiente de loss_function
- learning_rate : la tasa de aprendizaje. Es el parámetro crítico, pues en esta sección nos centramos en estudiar el comportamiento del gradiente descendente en base a este parámetro (y también del starting_point)
- max_iterations: número máximo de iteraciones. Así tenemos una condición sufiente para saber que el algoritmo va a parar
- target_error : error debajo del cual paramos de iterar
- verbose: si es True, la función devuelve el error de cada solución de las iteraciones

Esta función devuelve el vector de soluciones que hemos ido construyendo. Si queremos obtener la solución final, lo único que tenemos que hacer es acceder a la última posición. Así podemos generar las gráficas en las que plasmamos tanto la función de error como los puntos que vamos construyendo. También, si tenemos activado el parámetro Verbose, devolvemos el error en cada iteración, lo que es necesario para hacer las gráficas de evolución de error que se nos piden.

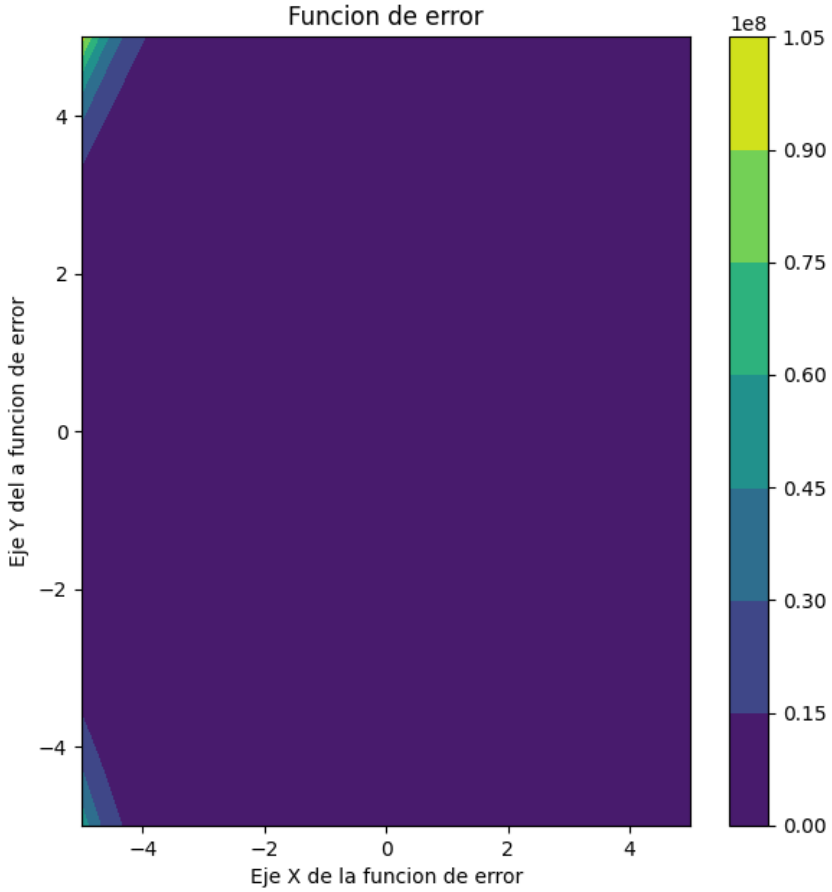
1.2 Apartado 2

Consideramos la función de error dada por:

$$E(u, v) := (u^3 * e^{v-2} - 2v^2 e^{-u})^2$$

Usar gradiente descendente para encontrar un mínimo de esta función, comenzando desde el punto $(u, v) = (1, 1)$ y usando una tasa de aprendizaje $\eta = 0, 1$.

Podemos mostrar la función de error en una vista de pájaro (la vista tridimensional solo será necesaria en casos en los que la vista de pájaro no sea suficiente). Preferimos esta vista bidimensional porque es más rápida de computar y en muchos casos más fácil de interpretar.



Los colores más azulados y oscuros indican los valores más bajos del error, mientras que valores verdosos y claros indican valores altos del error (la leyenda de la gráfica ya indica esto). Así que nuestros puntos deberán ir acercándose a zonas oscuras de la gráfica, si el comportamiento de los algoritmos es bueno.

1.2.1 Subapartado a

Calculamos analíticamente la expresión de las derivadas parciales:

$$\frac{\partial E}{\partial u} = 2 * (u^3 * e^{v-2} - 2v^2e^{-u}) * (3u^2e^{v-2} + 2v^2e^{-u})$$

$$\frac{\partial E}{\partial v} = 2 * (u^3 * e^{v-2} - 2v^2e^{-u}) * (u^3 * e^{v-2} - 4ve^{-u})$$

$$\nabla E = (\frac{\partial E}{\partial u}, \frac{\partial E}{\partial v})$$

Esta es la expresión del gradiente que usamos en el código. Expresión que mostramos por pantalla, como se indica en el guión de la práctica.

1.2.2 Subapartados b y c

Se manda explícitamente que usemos flotantes de 64 bits, para lo cual usamos la orden np.float64 para devolver todas nuestras funciones que representan los errores y las derivadas parciales.

Como se muestra en el código, solo necesitamos 10 iteraciones para quedarnos por debajo de un error (o valor de $E(u, v)$, que es lo que estamos considerando como función de error a minimizar) de valor 10^{-14}. En el código indicamos que la primera iteración por debajo del error es la iteración 9, pero hay que tener en cuenta que empezamos a contar desde el cero.

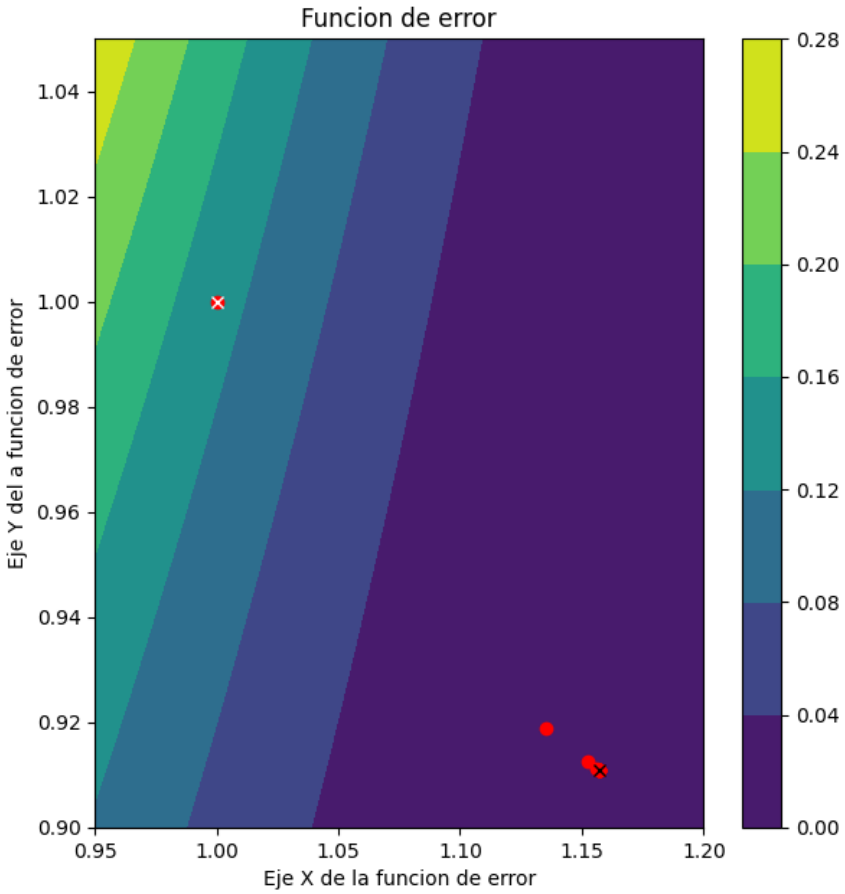
Los resultados que mostramos por pantalla nos indican que alcanzamos la solución:

```
Numero de iteraciones: 10
Pesos encontrados: [1.15728885 0.91083837]
```

Estos son los resultados vinculados a la onceava iteración (recordar que empezamos a contar desde el cero). Los resultados asociados a la décima iteración, en los que ya estamos por debajo del error buscado, son:

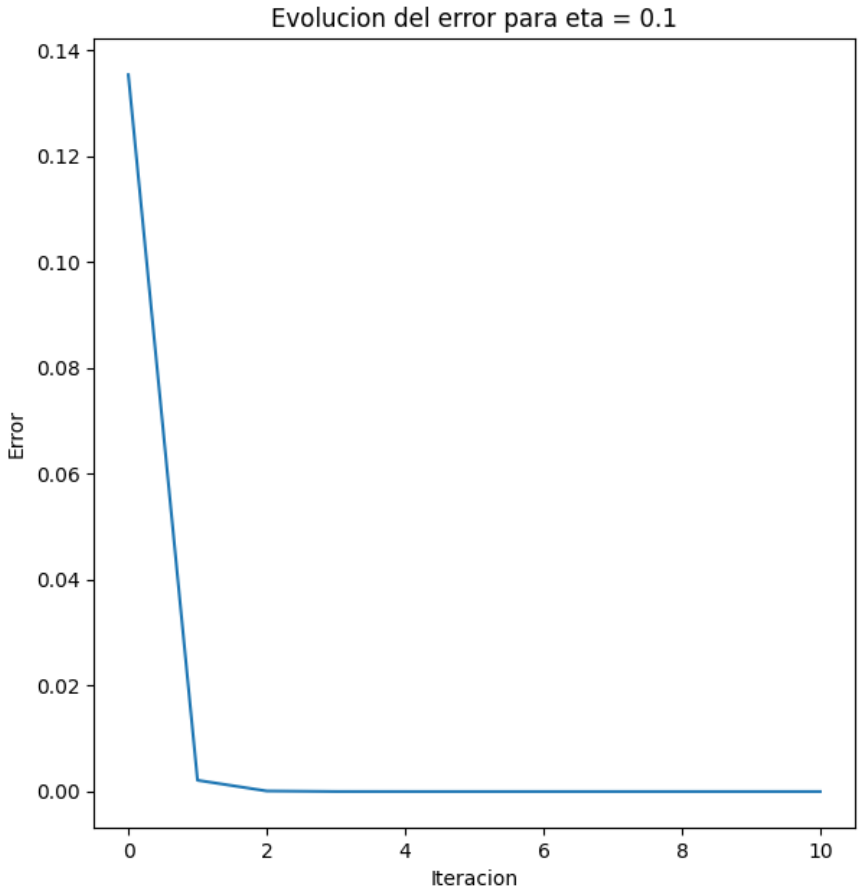
```
Primera iteracion por debajo de 10e-14: 9 (contando desde cero)
Las primeras coordenadas por debajo del error: [1.15728875 0.9108384 ]
```

Todo esto se muestra en las salidas por pantalla de nuestro programa. Además, podemos mostrar cómo avanzan nuestras soluciones sobre la superficie que representa la función de error:



El punto con una cruz blanca es el punto inicial. El punto con una cruz negra es el punto final de la búsqueda. Viendo la gráfica de las soluciones generadas es claro que podríamos seguir avanzando con la búsqueda, pero decidimos parar porque estamos por debajo de la cota de error pedida.

La gráfica del error para los parámetros que se nos han indicado en 1.2 es la siguiente:



1.3 Apartado 3

Se considera ahora la función:

$$f(x,y) := (x+2)^2 + 2(y-2)^2 + 2sin(2\pi x)sin(2\pi y)$$

Como vamos a usar la técnica del gradiente descendente, necesitamos calcular analíticamente la expresión del gradiente:

$$\frac{\partial f(x,y)}{\partial x} = 2(x+2) + 4\pi sin(2\pi y)cos(2\pi x)$$

$$\frac{\partial f(x,y)}{\partial y} = 4(y-2) + 4\pi sin(2\pi x)cos(2\pi y)$$

$$\nabla f(x,y) = (\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y})$$

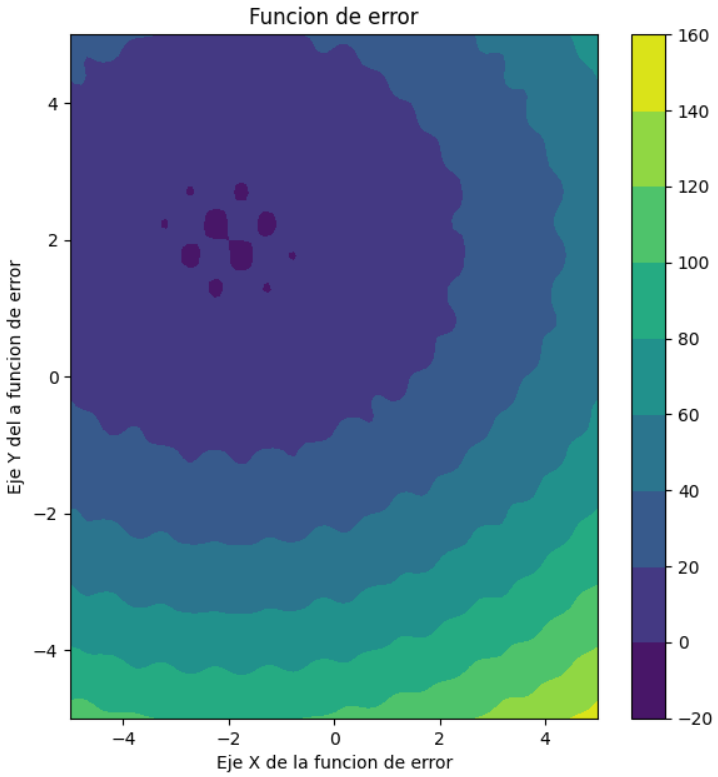
1.3.1 Apartado a

Se nos pide usar gradiente descendiente para minimizar la función de error f. Además, se nos pide que usemos como parámetros del gradiente descendente:

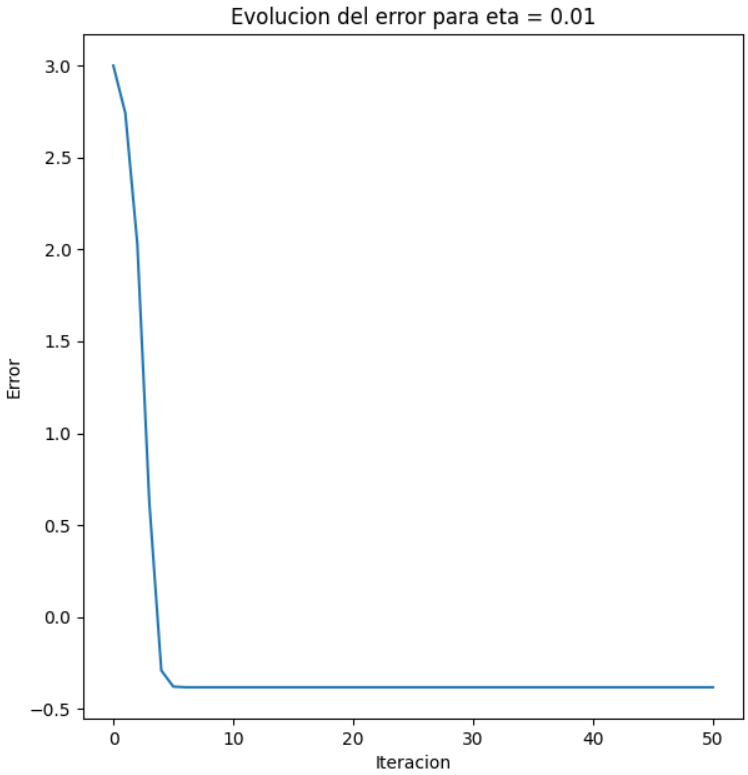
- $\eta = 0.01$
- $x_0 = -1, y_0 = 1$
- Máximo 50 iteraciones

Y que repitamos el experimento con un valor de $\eta = 0.1$.

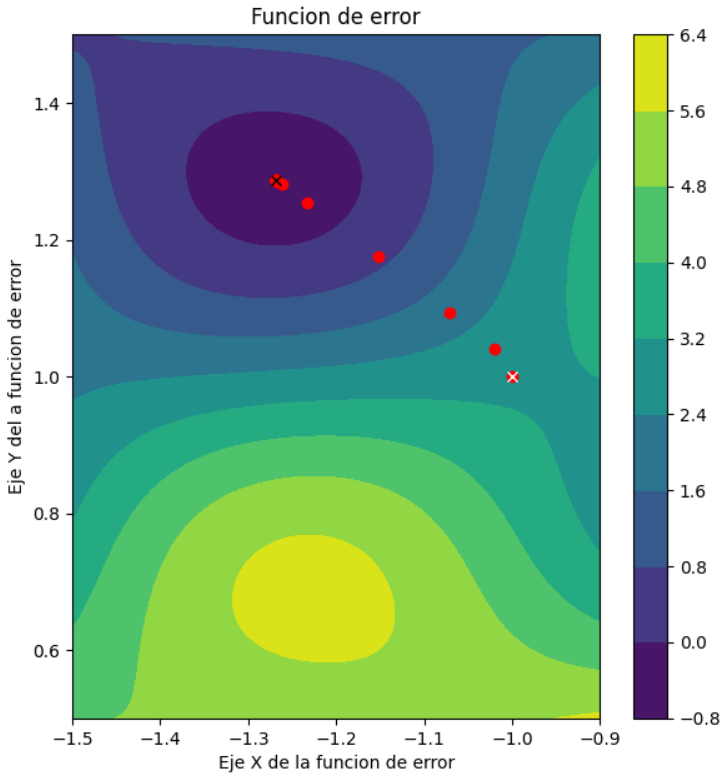
Lo primero que hacemos es mostrar una gráfica en dos dimensiones de la función de error que queremos minimizar:



Realizando gradiente descendente con los primeros parametros dados obtenemos la siguiente gráfica, en la que se muestra cómo evoluciona el valor del error según avanzan las iteraciones del algoritmo.

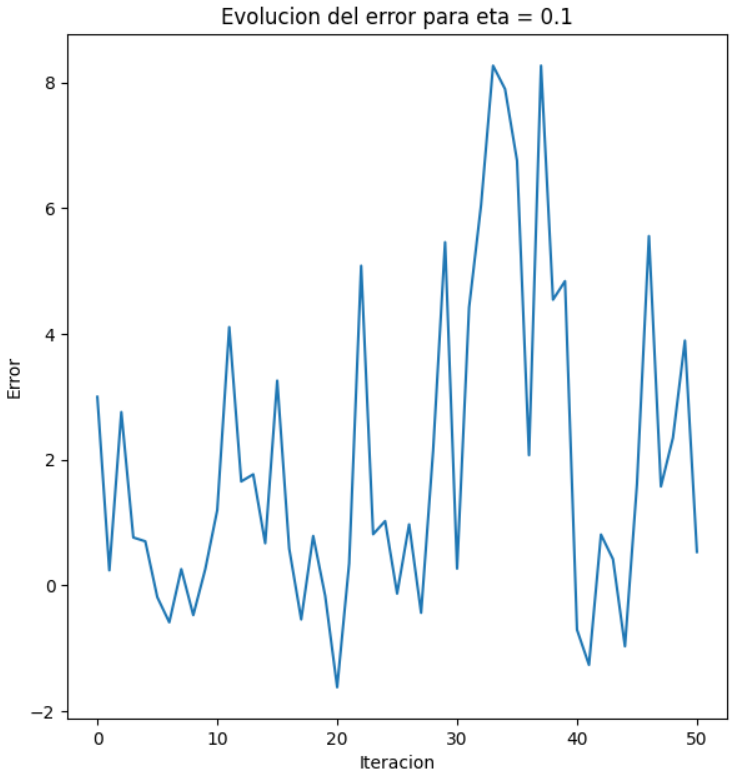


Mostramos, aunque no se nos pida en el guión, la traza del algritmo (soluciones plasmadas sobre la gráfica bidimensiona del error):

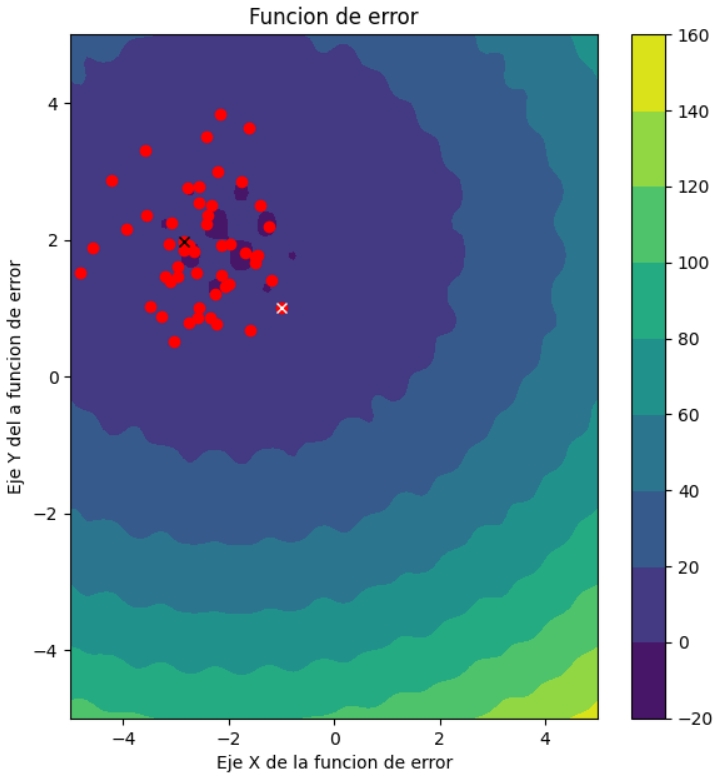


En esta gráfica se puede apreciar con facilidad que el algoritmo tiene el comportamiento deseado, pues cae con facilidad en un óptimo local.

Ahora realizamos el mismo proceso pero cambiando a $\eta = 0.1$. Obtenemos la siguiente gráfica de evolución del errolución del error:



De esta gráfica ya podemos ver que el comportamiento del algoritmo no es el deseado. Lo que esperaríamos es que el error disminuyese monótamente en cada iteración. Sin embargo, lo que nos encontramos es con que el error oscila con grandes saltos. De lo estudiado en teoría, sabemos que el motivo sea seguramente un valor de η demasiado grande. COn esto, cuando estamos cerca de una solución local, damos un salto muy grande en cierta dirección, saltando el óptimo local y yendo a una zona de alto error. Del mismo modo, de un salto desde una zona de alto error podemos acabar en una zona de muy bajo error. Es claro que este comportamiento errático no es deseable. Mostramos la traza de las soluciones en cada iteración para reforzar esta hipótesis sobre el mal comportamiento:



Con este gráfico, vemos que en la última iteración (del gráfico de evolución de error se deduce ya que se han consumido el total de iteraciones) hemos acabado, por buena suerte, en una buena zona de la función del error.

Con este ejemplo, podemos ver que es importante explorar con distintos valores de η, pues estos influyen mucho en el buen o mal comportamento del gradiente descendiente. Es más, ya podemos intuir que un ajuste dinámico del valor de η puede ser muy interesante, pues como hemos visto en teoría, nos interesa avanzar con *"pasos de gigante"* al principio con un η grande, y disminuir el valor de η al acercarnos a un óptimo local para tener más precisión en las iteraciones.

En estos dos ejemplos no muestro los valores numéricos de las soluciones obtenidas ni del error alcanzado. Considero que es mucho más interesante e informativo visualizar las gráficas, tanto del error como de la traza, pues estamos estudiando el comportamiento del algoritmo, no una solución concreta que no nos interesa al no ser un problema que realmente queramos

resolver.

1.3.2 Apartado b

Ahora repetiremos el mismo experimento pero tomando distintos valores para las soluciones iniciales. Y en este caso se pide que generemos una tabla con los valores obtenidos. Como no se especifica nada, tomamos $\eta \in 0.01, 0.1$ y un máximo de 1000 iteraciones. Dejamos la cota de error a None para que no se hagan comprobaciones de cotas. Además de la tabla, mostraremos las gráficas de la evolución del error y la traza de soluciones en los casos en los que sea interesante comentarlas (no en todas pues resultaría repetitivo).

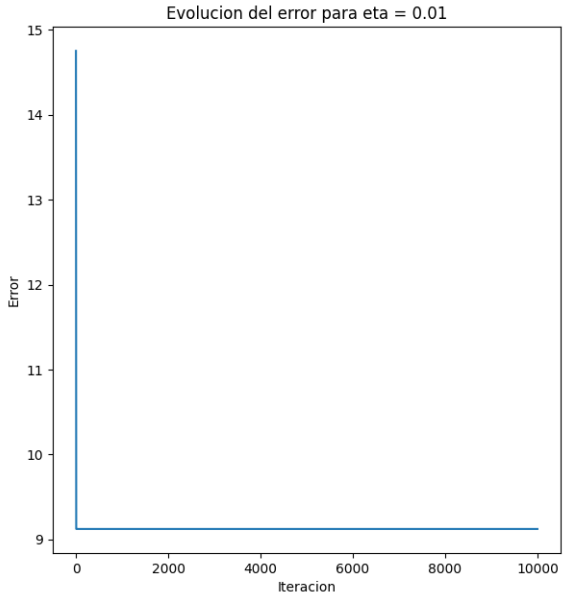
La tabla con los datos generados es:

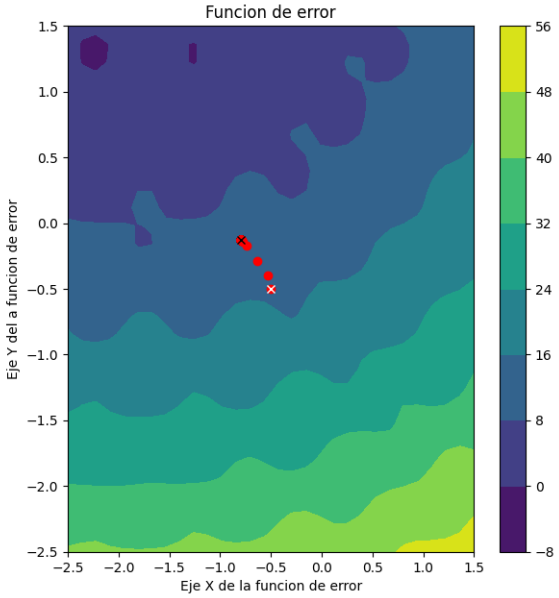
Punto Inicial	Learning Rate η	Solución alcanzada	Error Alcanzado
(-0.5, -0.5)	0.01	(-0.7934, -0.1259)	9.1251
(-0.5, -0.5)	0.1	(-1.1000, 3.4598)	4.7785
(1, 1)	0.01	(0.6774, 1.2904)	6.4375
(1, 1)	0.1	(-2.3820, 2.0917)	-0.5734
(2, 1)	0.01	(1.6466, 1.2954)	12.7624
(2, 1)	0.1	(-1.8963, 2.1082)	0.7969
(-2, 1)	0.01	(-2.2436, 1.2864)	-0.8685
(-2, 1)	0.1	(-0.8079, 2.0814)	2.3492
(-3, 3)	0.01	(-2.7309, 2.7132)	-0.3812
(-3, 3)	0.1	(-2.1558, 3.0127)	1.9427
(-2, 2)	0.01	(-2, 2)	$-4.7992 * 10^{-31}$
(-2, 2)	0.1	(-1.3420, 0.6155)	5.3779

Esta tabla ha sido obtenida a partir de los datos mostrados por pantalla. El programa no genera está tabla formateada, pero muestra por pantalla todos los valores que esta recoge.

1.3.3 Comentarios sobre los resultados obtenidos

Para los primeros valores de la tabla tenemos las siguientes dos gráficas:

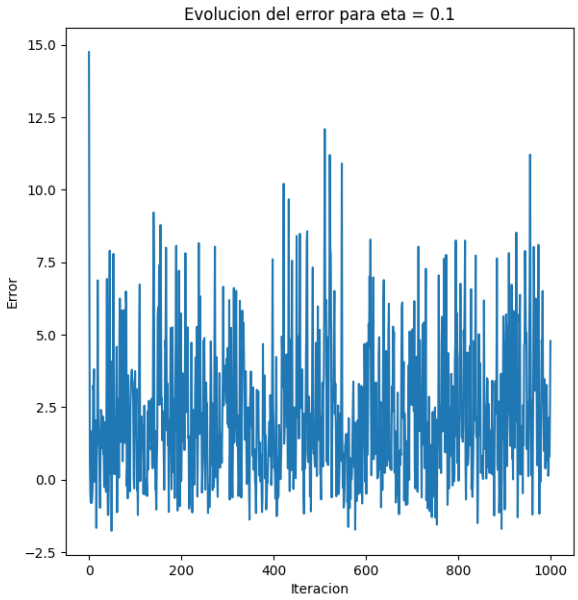




Vemos que damos un primer salto grande en el descenso del error, pero

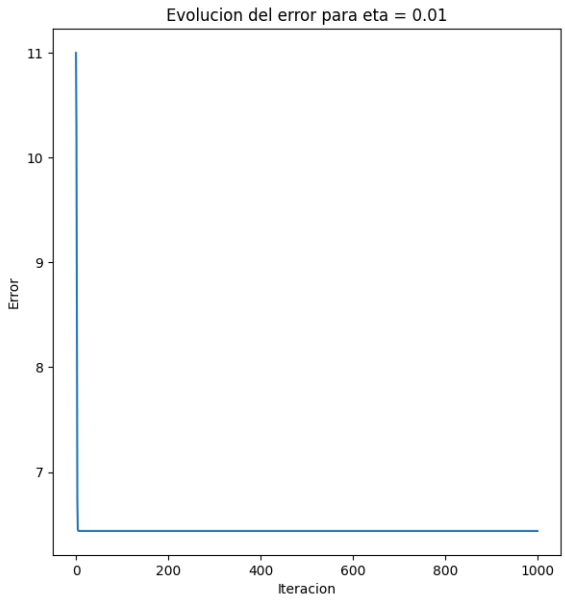
luego nuestro valor de η es demasiado pequeño para converger a un óptimo local. La gráfica en tres dimensiones que generamos no aporta ninguna información de interés.

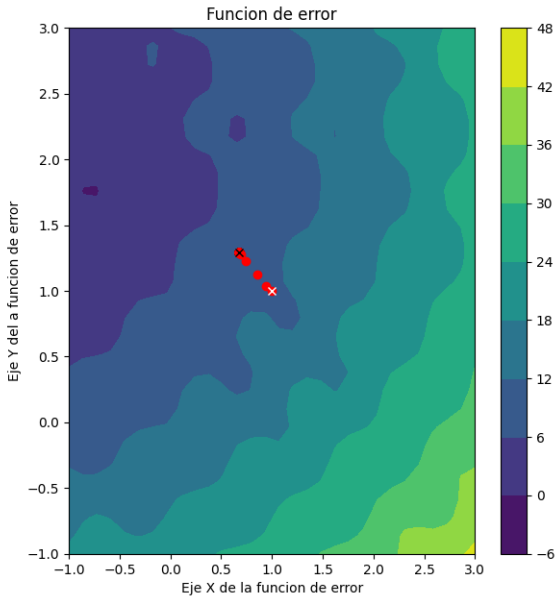
Para la segunda fila de la tabla, tenemos la siguiente gráfica de evolución del error:



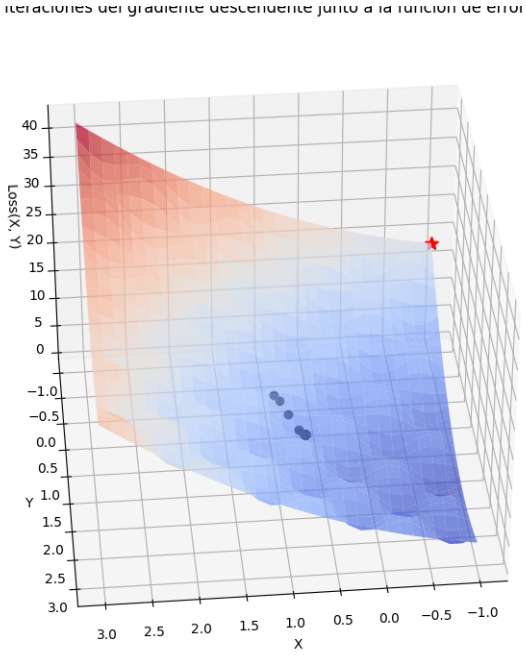
El valor de η es demasiado grande, por lo tanto, el error fluctúa tanto. No es un valor apropiado para η, por lo que hemos comentado en apartados anteriores. Y a pesar de este comportamiento errático, por casualidad, acabamos con un error menor que con el valor apropiado para η que aparece en la primera fila de la tabla.

Para la tercera fila de la tabla, parece que nos vuelve a pasar lo mismo que para la primera fila, a vista de las siguientes dos gráficas:





Pero si nos fijamos en la siguiente gráfica:

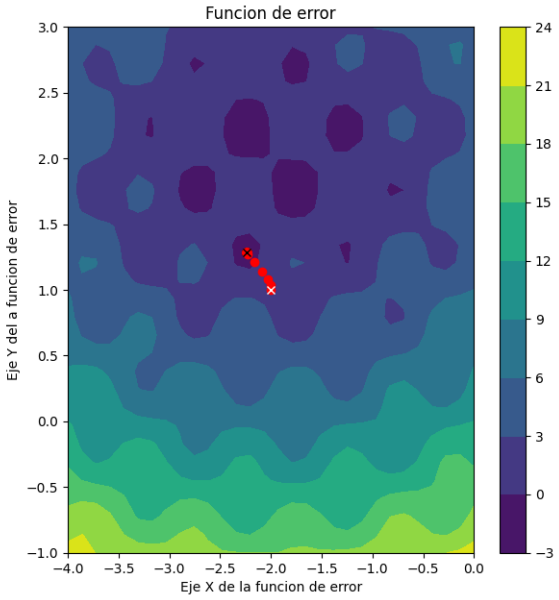


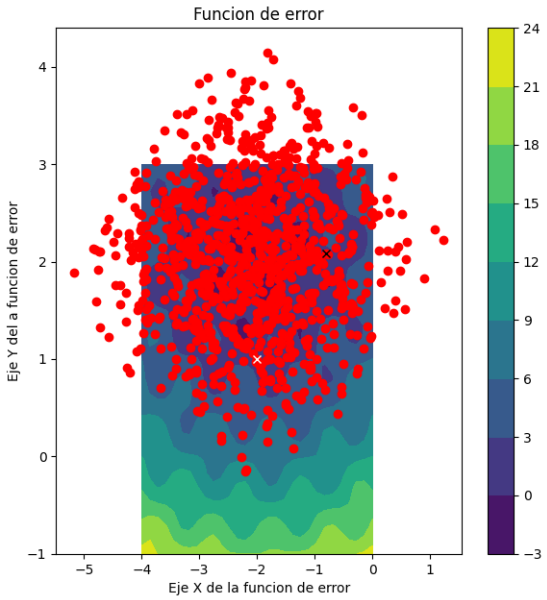
Lo que podemos ver es que nos quedamos en una especie de *bache* sobre el terreno. La conclusión es la misma, el valor de η tan pequeño nos deja atascado, pero ahora es más claro el motivo. No es que simplemente los 1000 *pasos* que damos sean muy cortos, es que llegamos a un óptimo local que con el pequeño valor de η no conseguimos superar.

Para la cuarta fila, con un $\eta = 0.1$, tenemos un valor demasiado grande, lo que produce un comportamiento errático. Y de nuevo, por pura casualidad, alcanzamos un mejor valor de del error que usando un valor de η adecuado. No incluyo gráficas pues son muy parecidas a las ya mostradas cuando ocurre esto, para evitar ser repetitivo.

Para la quinta fila, tenemos el mismo comportamiento comentado en el caso en el que las soluciones se quedaban atascadas en un *bache*. Mientras que para la sexta fila volvemos a tener un comportamiento errático, que de nuevo, vuelve a dar mejores resultados por casualidad.

En la séptima fila, vemos como las soluciones convergen correctamente hacia un óptimo local para un valor de $\eta = 0.01$. Mientras que para $\eta = 0.1$, tenemos un comportamiento errático. Esto se muestra con claridad en las siguientes dos gráfica de la traza de las soluciones (en la leyenda se muestran los valores de *eta*)



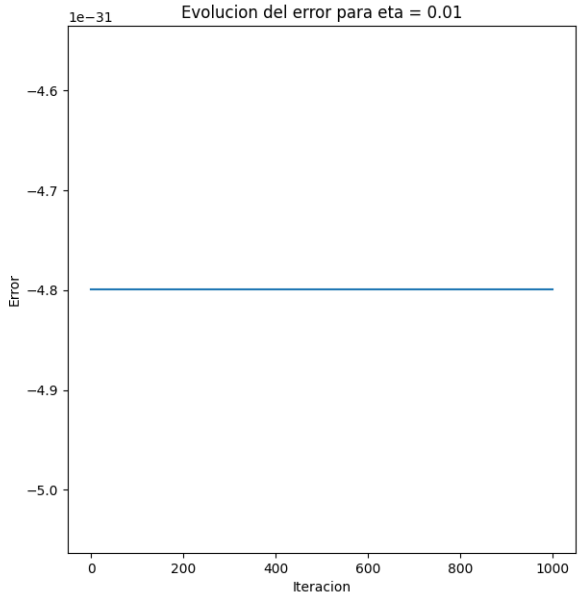


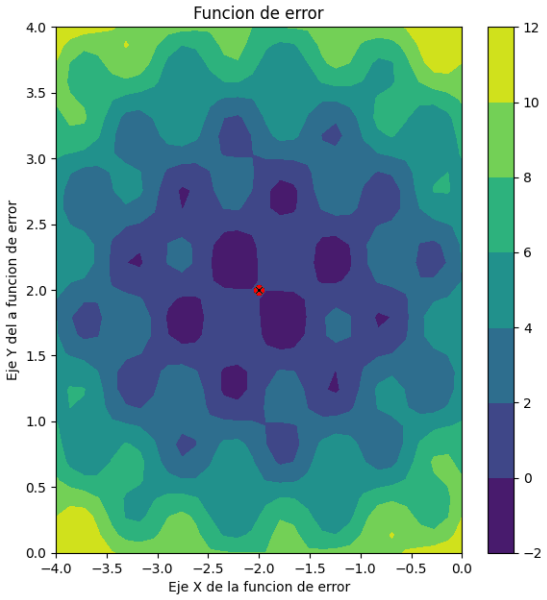
Además, en este caso, un valor apropiado de η consigue mejores resulta-

dos que con un valor que produce comportamiento errático.

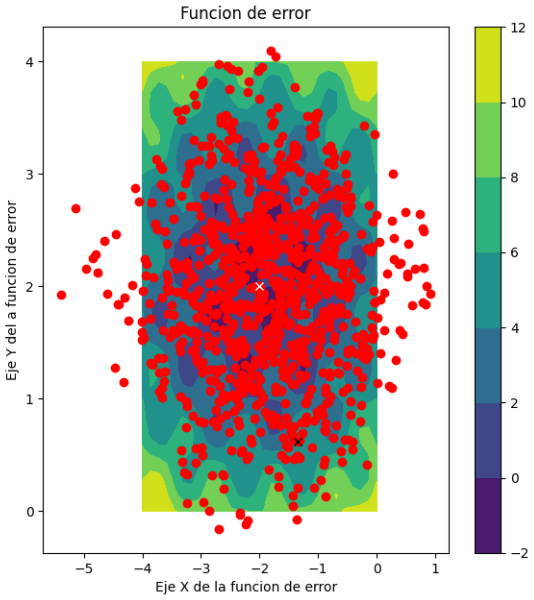
En la novena fila tenemos un muy buen comportamiento qeu da convergencia hacia un óptimo local (en la gráfica de la traza de la solución que muestra el programa esto se ve claramente. No añado esta gráfica pues ya se ha mostrado una muy parecida, para evitar ser repetitivo). Mientras que en la décima fila (η demasiado grande) tenemos el claro caso de comportamiento errático, que en este caso produce peor solución que un adecuado valor de η.

En la undécima fila tenemos un comportamiento muy curioso. La solución inicial y final coinciden. Esto se ve claramente en las siguientes dos gráficas:





Vemos como estamos en el filo de dos *agujeros* a los que nos interesaría caer. Estamos en un equilibrio muy inestable en lo que podría llamarse un punto de silla. En este caso en concreto, queremos que se rompa dicho equilibrio pues nos haría caer en uno de los dos óptimos locales. Notar también, que consumimos las mil iteraciones sin modificar la solución. Esto nos asegura que estamos en un punto de silla, pues el que no se mueva implica que el gradiente es cero. Gradiente nulo en un punto que no es óptimo indica con seguridad que es un punto de silla. Sin embargo, para un valor más grande de η nos salimos del equilibrio, pero produciendo un comportamiento errático (como se muestra en la siguiente gráfica) que nos lleva a una peor solución:


Funcion de error
Eje Y del a funcion de error
Eje X de la funcion de error
4
3
2
1
0
−5
−4
−3
−2
−1
0
1
12
10
8
6
4
2
0
−2