Aprendizaje Automático

Práctica 1

Curso 2019-2020

Javier Gálvez Obispo
javiergalvez@correo.ugr.es
Grupo 2 Martes 17:30 – 19:30

1. Ejercicio sobre la búsqueda iterativa de óptimos

1. Implementar el algoritmo de gradiente descendente.

Consultar código para este apartado.

2. Considerar la función $E(u, v) = (ue^{v} - 2ve^{-u})^2$. Usar gradiente descendente para encontrar un mínimo de esta función, comenzando desde el punto $(u, v) = (1, 1)$ y usando una tasa de aprendizaje $\eta = 0{,}1$.

a) Calcular analíticamente y mostrar la expresión del gradiente de la función $E(u, v)$

Las derivadas parciales de $E(u, v)$ son las siguientes:

$$\frac{\partial}{\partial u}(ue^{v}-2ve^{-u})^2=2e^{-2u}(ue^{u+v}-2y)(e^{u+v}+2y)$$

$$\frac{\partial}{\partial v}(ue^{v}-2ve^{-u})^2=2e^{-2u}(ue^{u+v}-2)(ue^{u+v}-2y)$$

b) ¿Cuántas iteraciones tarda el algoritmo en obtener por primera vez un valor de $E(u, v)$ inferior a 10^{-14}?

Se tarda 10 iteraciones en alcanzar un valor $E(u, v)$ inferior a 10^{-14}

c) ¿En qué coordenadas (u, v) se alcanzó por primera vez un valor igual o menor a 10^{-14} en el apartado anterior?

Las coordenadas (u, v) donde se alcanza un valor igual o menor a 10^{-14} son:

(u, v) = (0.044736290397782055 , 0.023958714099141746)

3. Considerar ahora la función $f(x, y)=(x-2)^2+2(y+2)^2+\sin(2\pi x)+\sin(2\pi y)$

a) Usar gradiente descendente para minimizar esta función. Usar como punto inicial ($x_0 = 1$, $y_0 = -1$), tasa de aprendizaje $\eta = 0{,}01$ y un máximo de 50 iteraciones. Generar un gráfico de cómo desciende el valor de la función con las iteraciones. Repetir el experimento pero usando $\eta = 0{,}1$, comentar las diferencias y su dependencia de η.

Como podemos observar en la gráfica utilizando $\eta = 0{,}01$ el algoritmo de gradiente descendente converge en pocas iteraciones a un mínimo, mientras que para $\eta = 0{,}1$ no llega a converger, lo saltos son demasiado bruscos para encontrar una solución rapidamente.

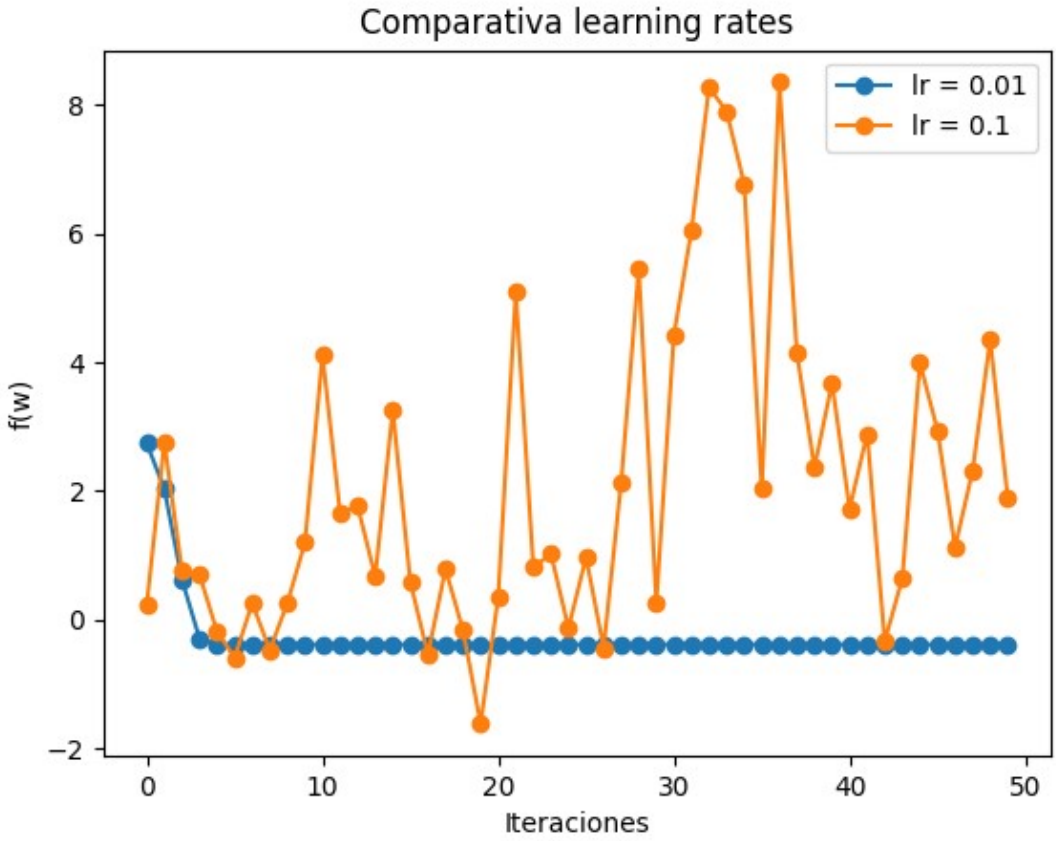

Figure 1: Comparativa del gradiente descendente usando dos valores de η distintos

b) Obtener el valor mínimo y los valores de las variables *(x, y)* en donde se alcanzan cuando el punto de inicio se fija en: (2,1, -2,1), (3, -3), (1,5, 1,5), (1, -1). Generar una tabla con los valores obtenidos.

	(2,1, -2,1)	(3, 3)	(1,5, 1,5)	(1, -1)
(x, y)	(2,2438, -2,2379)	(2.7309, -2.7132)	(1.7791 , 1.0309)	(1.2690 , -1.2867)
f(x, y)	-1.8200785415471	-0.3812494974380	18.0420723493120	-0.3812494974381

4. ¿Cuál sería su conclusión sobre la verdadera dificultad de encontrar el mínimo global de una función arbitraria?

El encontrar el mínimo global de una función depende tanto del *learning rate* que fijemos como del punto inicial del que partimos. Dependiendo de lo que valga el *learning rate* es posible que nunca encontremos el mínimo, ya sea porque no converga al ser demasiado grande, o bien, porque sea demasiado pequeño y no avance lo suficientemente rápido. Es por esto que existen variaciones

del gradiente descendente en el que el *learning rate* no es fijo, es decir, se va actualizando durante las iteraciones del algoritmo.

Respecto al problema del punto de partida, es posible que nos quedemos estancados en mínimos locales y no podamos encontrar el mínimo global de la función. Esto se soluciona fácilmente partiendo de puntos arbitrarios.

2. Ejercicio sobre regresión lineal

Este ejercicio ajusta modelos de regresión a vectores de características extraidos de imágenes de dígitos manuscritos. En particular se extraen dos característcas concretas: el valor medio del nivel de gris y simetría del número respecto de su eje vertical. Solo se seleccionarán para este ejercicio las imágenes de los números 1 y 5.

1. Estimar un modelo de regresión lineal a partir de los datos proporcionados de dichos números (Intensidad promedio, Simetria) usando tanto el algoritmo de la pseudo-inversa como Gradiente descendente estocástico (SGD). Las etiquetas serán {−1,1}, una para cada vector de cada uno de los números. Pintar las soluciones obtenidas junto con los datos usados en el ajuste. Valorar la bondad del resultado usando E_{in} y E_{out}.

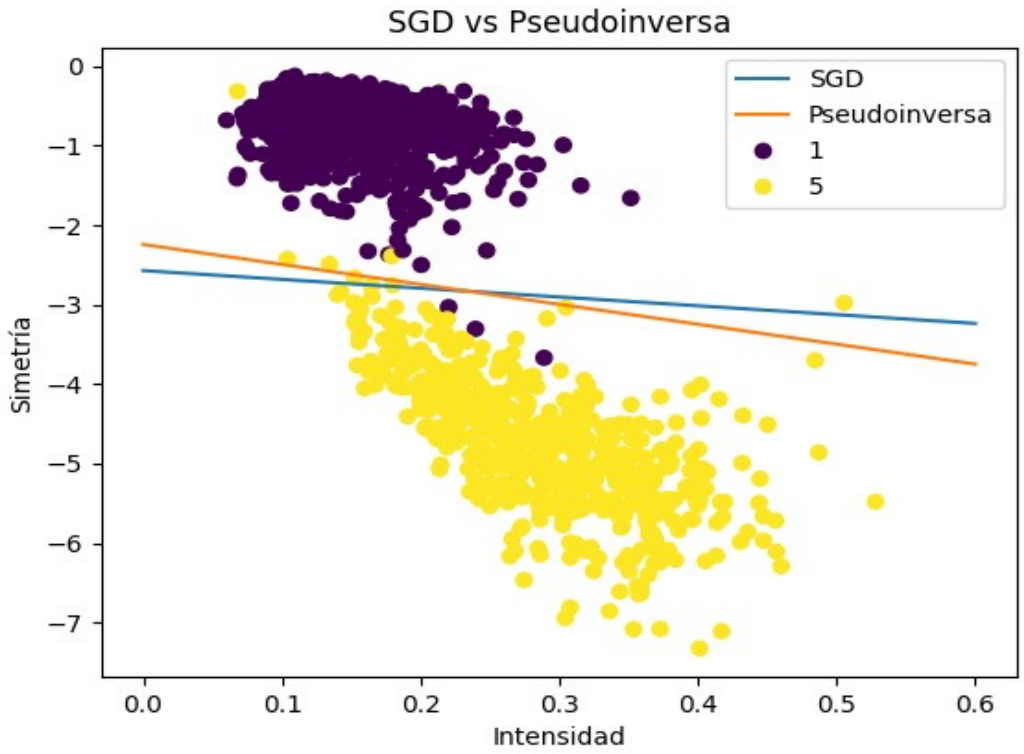

Figure 2: Comparativa de los algoritmos SGD y Pseudoinversa para los mismos datos

	SGD	Pseudoinversa
E_{in}	0.08041077153991012	0.07918658628900395
E_{out}	0.13413600770624948	0.13095383720052584

Para el algoritmo de gradiente descendente estocástico se ha utilizado un *learning rate* η = 0,01, los minibatches están formados por 128 elementos y se ha puesto un límite de 10000 iteraciones.

Como podemos ver en la tabla, los resultados del algoritmo de la pseudoinversa son mejores aunque ambos errores son parecidos. La ventaja del algoritmo de la pseudoinversa es que éste solo depende de los datos de entrada y sus etiquetas, mientras que el SGD depende de otros tres parámetros adicionales que afectan al error estimado tal y como hemos visto en el primer ejercicio. A pesar de ser una variación del algoritmo utilizado en el primer ejercicio, el concepto sobre el que se basa el SGD es el mismo, por lo que el *learning rate* sigue siendo igual de importante para alcanzar una buena solución.

2. En este apartado exploramos como se transforman los errores E_{in} y E_{out} cuando aumentamos la complejidad del modelo lineal usado. Ahora hacemos uso de la función simula_unif(N,2,size) que nos devuelve N coordenadas 2D de puntos uniformemente muestreados dentro del cuadrado definido por [−size,size]×[−size,size].

a) Generar una muestra de entrenamiento de N= 1000 puntos en el cuadrado X = [−1,1]×[−1,1]. Pintar el mapa de puntos 2D.

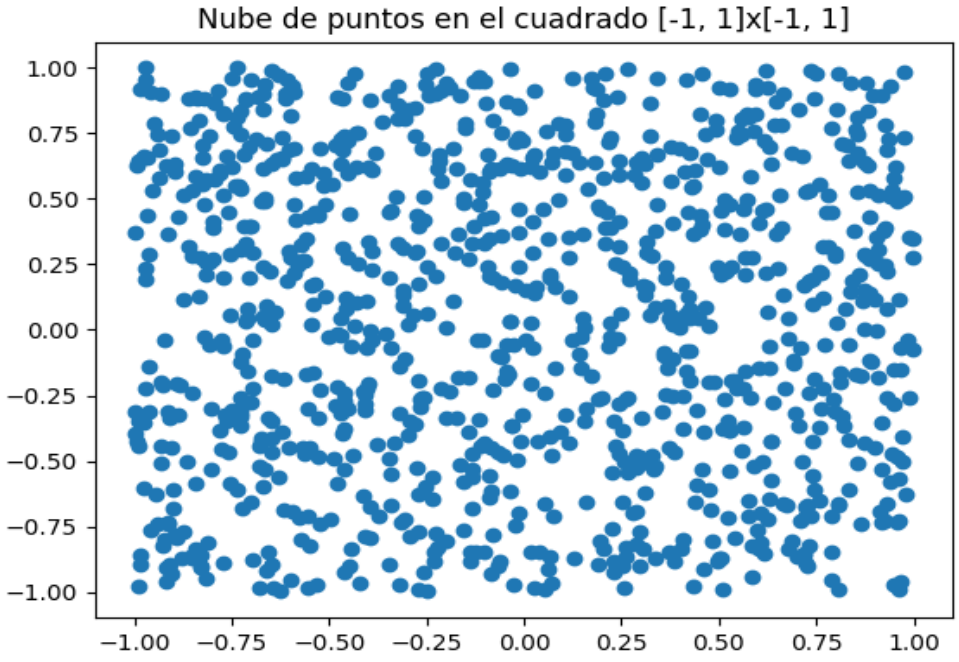

Figure 3: Nube de puntos uniformemente muestreados en el cuadrado [-1,1]x[-1,1]

b) Consideremos la función $f(x1,x2)=sign((x_1-0,2)^2+x_2^2-0,6)$ que usaremos para asignar una etiqueta a cada punto de la muestra anterior. Introducimos ruido sobre las etiquetas cambiando aleatoriamente el signo de un 10 % de lasmismas. Pintar el mapa de etiquetas obtenido.

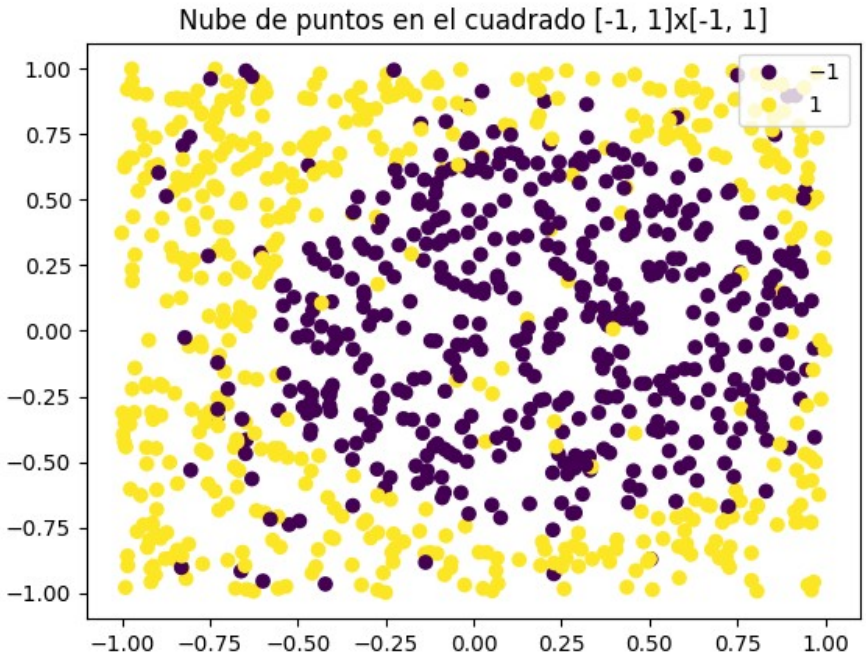

Figure 4: Nube de puntos uniformemente muestreados en el cuadrado [-1,1]x[-1,1] coloreados según sus etiquetas

c) Usando como vector de características (1, x_1, x_2) ajustar un modelo de regresion lineal al conjunto de datos generado y estimar los pesos *w*. Estimar el error de ajuste E_{in} usando Gradiente Descendente Estocástico (SGD).

Al ajustar el conjunto de datos con un modelo de regresión lineal la estimación del error que obtenemos es E_{in} = 0.9146834438996155. Este resultado es exageradamente malo ya que nuestros datos no se aproximan a un modelo lineal.

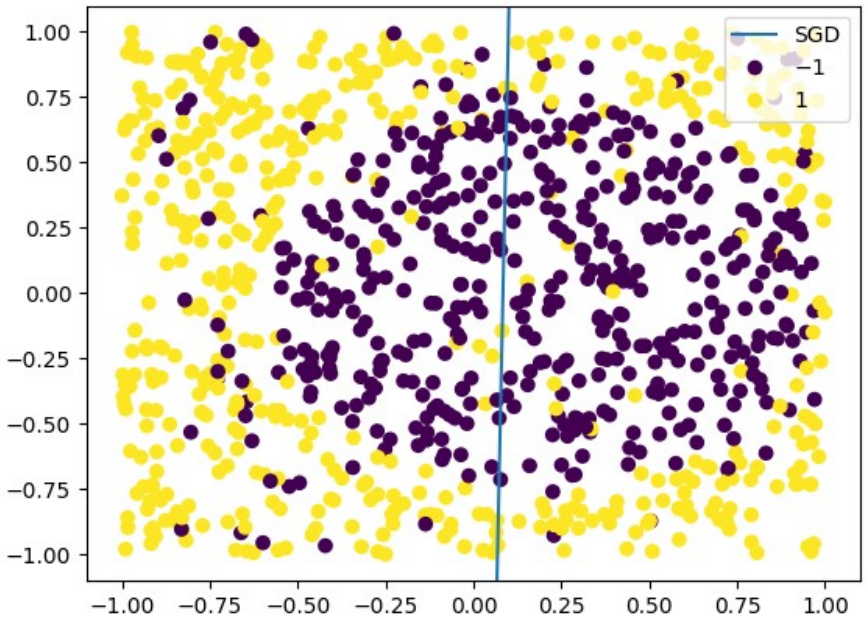

Figure 5: Nubes de puntos ajustado a un modelo lineal

En la gráfica se puede observar claramente que la división que hace el plano de nuestro conjunto de datos no es realmente útil.

d) Ejecutar todo el experimento definido por (a)-(c) 1000 veces (generamos 1000 muestras diferentes) y

- Calcular el valor medio de los errores E_{in} de las muestras.
- Generar 1000 puntos nuevos por cada iteración y calcular con ellos el valor de E_{out} en dicha iteración. Calcular el valor medio de E_{out} en todas las iteraciones.

  Media de los E_{in} = 0.9239637033308549

  Media de los E_{out} = 0.9290054647636586

  Aunque repitamos el experimento 1000 veces nuestros datos no van a poder aproximarse a un modelo lineal, los resultados siguen siendo exageradamente malos.

e) Valore que tan bueno considera que es el ajuste con este modelo lineal a la vistade los valores medios obtenidos de E_{in} y E_{out}.

Ya se ha comentado en los apartados anteriores lo malos que son los resultados al no poder ajustar el conjunto de datos que tenemos a un modelo lineal.

- Repetir el mismo experimento anterior pero usando características no lineales. Ahora usaremos el siguiente vector de características: $\Phi_2(x)=(1,x_1,x_2,x_1x_2,x_1^2,x_2^2)$. Ajustar el nuevo modelo de regresión lineal y calcular el nuevo vector de pesos $\hat{w}$. Calcular los errores promedio de E_{in} y E_{out}.

  Media de los E_{in} = 0.580567962971636

  Media de los E_{out} = 0.5868922772396934

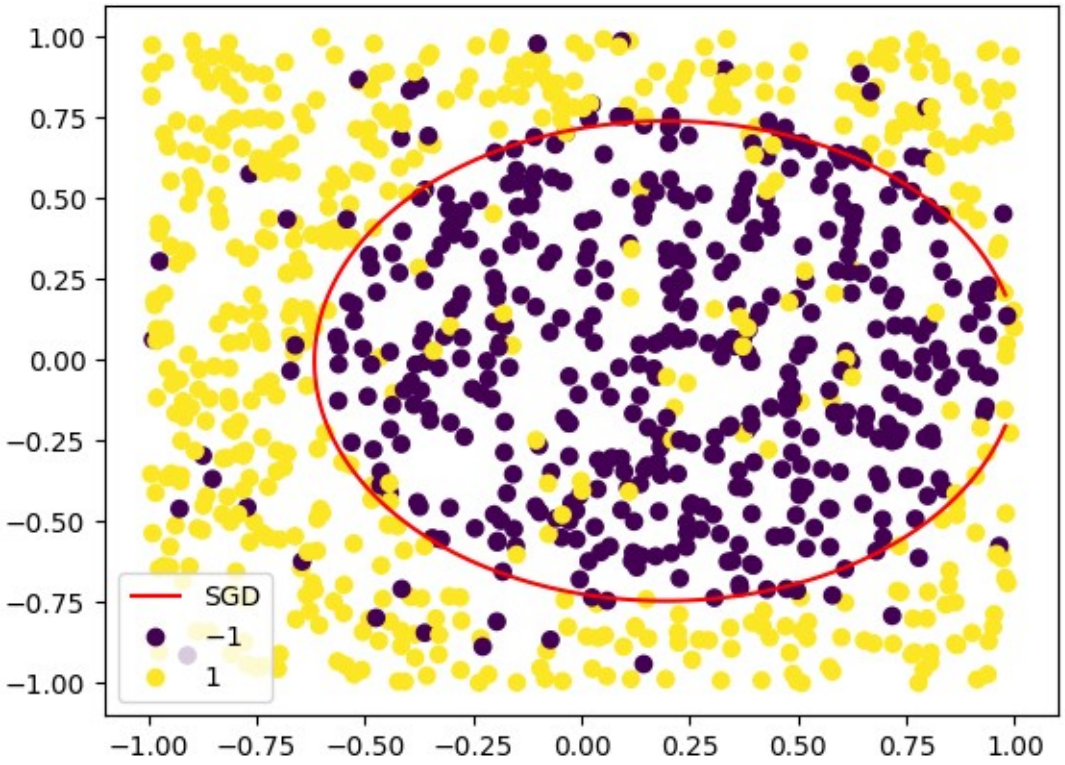

Figure 6: Representación del paraboloide en 2 dimensiones

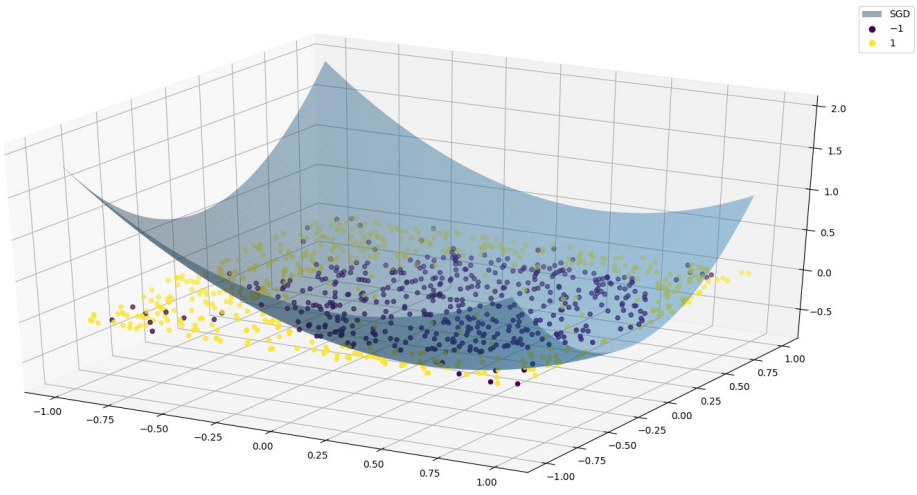

Figure 7: Representación del paraboloide en 3 dimensiones

- A la vista de los resultados de los errores promedios E_{in} y E_{out} obtenidos en los dos experimentos ¿Que modelo considera que es el más adecuado? Justifique la decisión.

  Viendo los resultados de ambos experimentos claramente el segundo es mejor, aun así, el error sigue siendo muy alto y deja mucho que desear. Para mejorar los resultados podríamos usar otro vector de característica no lineales.

2.1. Bonus

1. Implementar el algoritmo de minimización de Newton y aplicarlo a la funciónf(x,y) dada en el ejercicio 1.3. Desarrolle los mismos experimentos usando los mismos puntos de inicio.

- Generar un gráfico de como desciende el valor de la función con las iteraciones

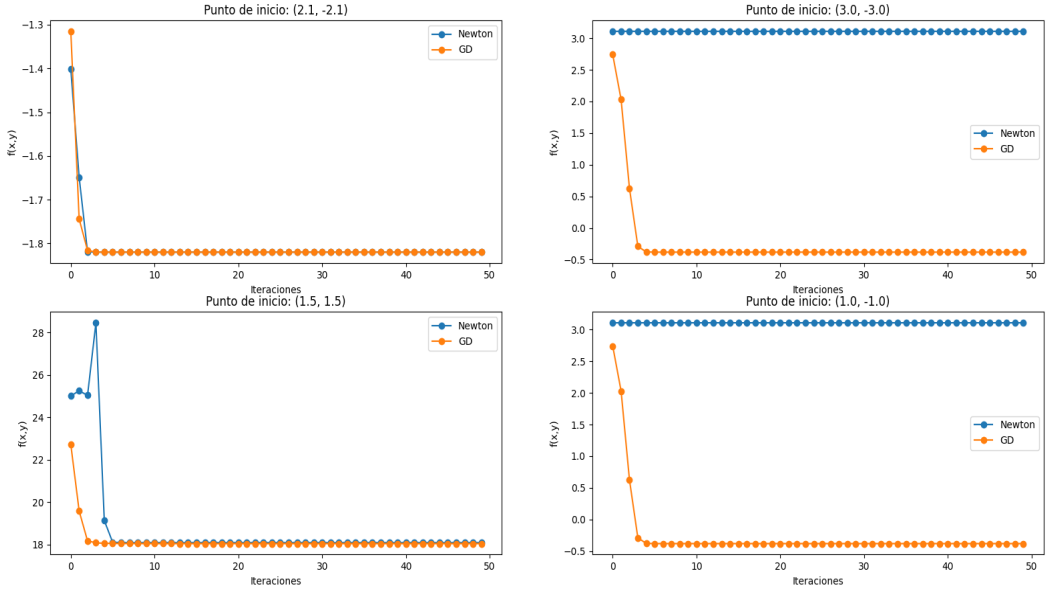

Figure 8: Comparativa del gradiente descendete y el método de Newton partiendo de los cuatro mismos puntos

	(2,1, -2,1)	(3, 3)	(1,5, 1,5)	(1, -1)
(x, y)	(1.7561, -1.7620)	(3.053, -3.0284)	(1.7048, 0.9731)	(0.9460, -0.9715)
f(x, y)	-1.8200785415471	3.10798006103520	18.0887420327078	3.10798006103520

- Extraer conclusiones sobre las conductas de los algoritmos comparando la curva de decrecimiento de la función calculada en el apartado anterior y la correspondiente obtenida con gradiente descendente.

  Podemos observar tanto en las gráficas como en las tablas que el método de Newton obtiene resultados similares a los del gradiente descendete para los puntos (2,1, -2,1) y (1,5, 1,5) mientras que para los puntos (3, -3) y (1, -1) los resultados son muy diferentes. Esto se debe a que el método de Newton busca puntos donde la pendiente sea 0, puntos de silla, y se detiene en estos. En vista de estos resultados no parece un buen algoritmo para utilizar, si lo que queremos es buscar óptimos globales. Además, el método de Newton hace uso de la matriz Hessiana, que puede llegar a ser muy complicada de obtener en casos mas complejos, lo cual lo hace poco atractivo a pesar de tener la ventaja de no depender del *learning rate* que fijemos.