Informe del Proyecto de Optimización: Minimización de $f(x,y)=(x^2-1)^2+(y^2-2)^2$

1970-01-01

Resumen Ejecutivo

Objetivo: Encontrar el mínimo de la función $f(x,y)=(x^2-1)^2+(y^2-2)^2$ comparando dos algoritmos de optimización.

Métodos Implementados

- **Gradient Descent** (Descenso del Gradiente): Método simple de primer orden
- **BFGS** (Broyden-Fletcher-Goldfarb-Shanno): Método cuasi-Newton de segundo orden

Configuración Experimental

Rango de Experimentación: Puntos iniciales en $[-100,100)\times[-100,100)$ (10 experimentos)

Resultado Teórico: La función tiene 4 mínimos globales equivalentes en $(\pm 1,\pm\sqrt{2})$ con valor $f^{i}=0$.

Resultado Principal

- **BFGS**: 100% de éxito (10/10 experimentos), promedio 24 iteraciones, precisión 10^{-19}
- **Gradient Descent**: 50% de fallo (5/10 experimentos divergieron), promedio 54 iteraciones cuando funciona, precisión 10^{-14}

Conclusión: BFGS es el método recomendado para este problema, siendo superior en robustez ($2\times$), eficiencia ($2.3\times$) y precisión ($270\times$).

Definición de la Función, Dominio y Signo

Definición de la Función

La función objetivo a minimizar es:

$$f(x,y)=(x^2-1)^2+(y^2-2)^2$$

Esta función es una composición de funciones polinomiales de grado cuatro.

Tipo de problema: Este es un problema de **optimización sin restricciones** (*unconstrained optimization*). No existen restricciones de igualdad ni desigualdad, por lo que no aplican las condiciones de Karush-Kuhn-Tucker (KKT) ni el análisis de restricciones activas.

Dominio

El dominio de la función es:

$$D_f = R^2$$

La función está definida para todos los pares ordenados $(x, y) \in R^2$, ya que no existen restricciones algebraicas (como divisiones por cero, raíces de números negativos, o logaritmos de números no positivos).

Signo de la Función

Dado que $f(x, y)$ es la suma de dos términos elevados al cuadrado:

$$f(x, y) = (x^2 - 1)^2 + (y^2 - 2)^2 \geq 0$$

Por lo tanto:

$$f(x, y) \geq 0 \, \forall \, (x, y) \in R^2$$

La función alcanza su valor mínimo global de 0 cuando ambos términos son simultáneamente cero:

$$x^2 - 1 = 0 \text{ y } y^2 - 2 = 0$$

Análisis de las Variables de la Función

Tipo de Variables

Las variables x e y son **variables continuas** que toman valores en el conjunto de los números reales R.

- **Continuas**: Ambas variables pueden tomar cualquier valor real dentro de su dominio, sin restricciones de discreción o valores enteros.
- **No acotadas**: No existen límites superiores o inferiores para los valores que pueden tomar x e y.

Independencia de las Variables

Las variables x e y son **independientes** entre sí, ya que la función se puede escribir como la suma de dos funciones separables:

$$f(x,y)=g(x)+h(y)$$

donde:

- $g(x)=(x^2-1)^2$
- $h(y)=(y^2-2)^2$

Esta separabilidad implica que el comportamiento de la función respecto a x es independiente del valor de y y viceversa.

Análisis de Continuidad y Diferenciabilidad

Continuidad

La función $f(x,y)$ es **continua** en todo su dominio R^2.

Justificación: $f(x,y)$ es una composición y suma de funciones polinomiales, las cuales son continuas en R. Específicamente:

- x^2, y^2 son funciones polinomiales continuas
- (x^2-1) y (y^2-2) son continuas (suma/resta de funciones continuas)
- $(x^2-1)^2$ y $(y^2-2)^2$ son continuas (composición de funciones continuas)
- La suma de funciones continuas es continua

Diferenciabilidad

La función $f(x,y)$ es **infinitamente diferenciable** (clase C^∞) en todo R^2.

Justificación: Las funciones polinomiales son diferenciables en todo su dominio, y la composición y suma de funciones diferenciables es diferenciable. Podemos calcular derivadas parciales de cualquier orden.

Gradiente: $\nabla f(x,y)$

El gradiente de la función es el vector de derivadas parciales de primer orden:

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

Cálculo de las Derivadas Parciales

Derivada parcial respecto a x:

$$\frac{\partial f}{\partial x}=\frac{\partial}{\partial x}\left[\left(x^2-1\right)^2+\left(y^2-2\right)^2\right)$$

Aplicando la regla de la cadena:

$$\frac{\partial f}{\partial x}=2\left(x^2-1\right)\cdot 2x=4x\left(x^2-1\right)$$

Derivada parcial respecto a y:

$$\frac{\partial f}{\partial y}=\frac{\partial}{\partial y}\left[\left(x^2-1\right)^2+\left(y^2-2\right)^2\right)$$

Aplicando la regla de la cadena:

$$\frac{\partial f}{\partial y}=2\left(y^2-2\right)\cdot 2y=4y\left(y^2-2\right)$$

Gradiente

Por lo tanto, el gradiente es:

$$\nabla f(x,y)=\begin{bmatrix}4x\left(x^2-1\right)\\4y\left(y^2-2\right)\end{bmatrix}$$

Matriz Hessiana

La matriz Hessiana $H_f(x,y)$ contiene las derivadas parciales de segundo orden:

$$H_f(x,y)=\begin{bmatrix}\frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x\partial y}\\ \frac{\partial^2 f}{\partial y\partial x} & \frac{\partial^2 f}{\partial y^2}\end{bmatrix}$$

Cálculo de las Derivadas de Segundo Orden

Derivada segunda respecto a x:

$$\frac{\partial^2 f}{\partial x^2}=\frac{\partial}{\partial x}\left[4x\left(x^2-1\right)\right)=4\left(x^2-1\right)+4x\cdot 2x=12x^2-4$$

Derivada segunda respecto a y:

$$\frac{\partial^2 f}{\partial y^2}=\frac{\partial}{\partial y}\left[4y\left(y^2-2\right)\right)=4\left(y^2-2\right)+4y\cdot 2y=12y^2-8$$

Derivadas cruzadas:

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial}{\partial y}\left[4x\left(x^2-1\right)\right] = 0$$

$$\frac{\partial^2 f}{\partial y \partial x} = \frac{\partial}{\partial x}\left[4y\left(y^2-2\right)\right] = 0$$

Matriz Hessiana

$$H_f(x,y) = \begin{bmatrix} 12x^2-4 & 0 \\ 0 & 12y^2-8 \end{bmatrix}$$

La matriz Hessiana es **diagonal**, lo que confirma la independencia de las variables.

Análisis de Convexidad

Condiciones de Convexidad

Una función es convexa si su Hessiana es semidefinida positiva (todos sus valores propios son no negativos) en todo su dominio.

Valores Propios de la Hessiana

Dado que la Hessiana es diagonal, sus valores propios son simplemente los elementos de la diagonal:

$$\lambda_1 \quad ¿\, 12x^2-4$$
$$\lambda_2 \quad ¿\, 12y^2-8$$

Análisis de Convexidad

Para que la función sea convexa globalmente, necesitamos que ambos valores propios sean no negativos para todo $(x,y) \in R^2$:

$$\lambda_1 = 12x^2-4 \geq 0 \quad \Rightarrow x^2 \geq \frac{1}{3} \Rightarrow |x| \geq \frac{1}{\sqrt{3}}$$

$$\lambda_2 = 12y^2-8 \geq 0 \quad \Rightarrow y^2 \geq \frac{2}{3} \Rightarrow |y| \geq \sqrt{\frac{2}{3}}$$

Conclusión: La función **NO es convexa globalmente** en R^2, sino que es convexa solo en la región:

$$R = \left\{(x,y) : |x| \geq \frac{1}{\sqrt{3}} \text{ y } |y| \geq \sqrt{\frac{2}{3}}\right)$$

En las regiones donde $|x| < \frac{1}{\sqrt{3}}$ o $|y| < \sqrt{\frac{2}{3}}$, la función es **no convexa** (la Hessiana no es semidefinida positiva). En particular, cerca del origen $(0,0)$, donde ambos valores propios son negativos, la función presenta comportamiento localmente cóncavo.

Clasificación Completa de Regiones

1. **Región convexa** (Hessiana semidefinida positiva, ambos $\lambda_i \geq 0$):

$$R_{\text{convexa}} = \left\{ (x,y) : |x| \geq \frac{1}{\sqrt{3}} \text{ Y } |y| \geq \sqrt{\frac{2}{3}} \right\}$$

   En esta región, ambos valores propios son no negativos, garantizando convexidad local.

2. **Región cóncava** (Hessiana semidefinida negativa, ambos $\lambda_i \leq 0$):

$$R_{\text{cóncava}} = \left\{ (x,y) : |x| \leq \frac{1}{\sqrt{3}} \text{ Y } |y| \leq \sqrt{\frac{2}{3}} \right\}$$

   En esta región, ambos valores propios son no positivos, creando comportamiento localmente cóncavo. El punto $(0,0)$ (máximo local) está en el centro de esta región.

3. **Regiones silla** (Hessiana indefinida, valores propios de signos opuestos):

   - Región donde $|x| < \frac{1}{\sqrt{3}}$ Y $|y| \geq \sqrt{\frac{2}{3}}$: $\lambda_1 < 0, \lambda_2 \geq 0$
   - Región donde $|x| \geq \frac{1}{\sqrt{3}}$ Y $|y| < \sqrt{\frac{2}{3}}$: $\lambda_1 \geq 0, \lambda_2 < 0$

   En estas regiones, la función no es ni convexa ni cóncava. Los puntos silla $(0, \pm\sqrt{2})$ y $(\pm 1, 0)$ se encuentran en estas regiones.

4. **Fronteras de convexidad**:

   - Líneas verticales: $x = \pm \frac{1}{\sqrt{3}} \approx \pm 0.577$
   - Líneas horizontales: $y = \pm \sqrt{\frac{2}{3}} \approx \pm 0.816$

Implicaciones

La no convexidad global implica que:

- Los métodos de optimización basados en gradiente podrían converger a diferentes soluciones dependiendo del punto inicial (existen 4 mínimos globales equivalentes)

- Existen puntos silla que podrían ralentizar o afectar la convergencia
- Se requiere un análisis cuidadoso de los puntos estacionarios
- **Nota importante**: Aunque la función no es convexa globalmente, el análisis de la Hessiana (sección 8) demuestra que no existen mínimos locales que no sean globales. Todos los mínimos encontrados son mínimos globales.

Determinación del Mínimo Teórico

Puntos Estacionarios

Nota terminológica importante:

Según las definiciones en optimización:

- **Punto crítico (en cálculo)**: Aquellos donde la derivada es indefinida (no existe)
- **Punto estacionario**: Aquellos donde el gradiente es cero ($\nabla f = 0$)

En este problema:

- La función $f(x, y)$ es de clase C^∞ (infinitamente diferenciable en todo R^2)
- Por lo tanto, **NO existen puntos críticos** (la derivada existe en todos los puntos)
- Los puntos que buscamos son **puntos estacionarios** donde $\nabla f(x, y) = 0$

Justificación de ausencia de puntos críticos: La función $f(x, y) = (x^2 - 1)^2 + (y^2 - 2)^2$ es una composición de polinomios. Específicamente:

- Los términos x^2, y^2 son polinomios (funciones C^∞)
- Las sumas $(x^2 - 1)$ y $(y^2 - 2)$ son polinomios
- Las composiciones $(x^2 - 1)^2$ y $(y^2 - 2)^2$ son polinomios de grado 4
- La suma de polinomios es un polinomio

Como los polinomios son infinitamente diferenciables en todo R, la función f es infinitamente diferenciable en todo R^2. Por lo tanto, no existen puntos donde la derivada sea indefinida.

Los puntos estacionarios se encuentran donde el gradiente es cero:

$$\nabla f(x, y) = \begin{bmatrix} 4x(x^2 - 1) \\ 4y(y^2 - 2) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Esto requiere:

$$4x(x^2-1)=0 \quad \Rightarrow x=0 \text{ o } x=\pm 1$$
$$4y(y^2-2)=0 \quad \Rightarrow y=0 \text{ o } y=\pm\sqrt{2}$$

Lista de Puntos Estacionarios

Combinando todas las posibilidades, obtenemos 9 puntos estacionarios:

1. $(0,0)$
2. $(0,\sqrt{2})$
3. $(0,-\sqrt{2})$
4. $(1,0)$
5. $(1,\sqrt{2})$
6. $(1,-\sqrt{2})$
7. $(-1,0)$
8. $(-1,\sqrt{2})$
9. $(-1,-\sqrt{2})$

Análisis de los Puntos Estacionarios

Clasificación mediante el Criterio de la Hessiana

Para clasificar cada punto estacionario, evaluamos la Hessiana en cada punto y analizamos sus valores propios.

Punto (0, 0)

$$H_f(0,0)=\begin{bmatrix}-4 & 0\\ 0 & -8\end{bmatrix}$$

Valores propios: $\lambda_1=-4<0$, $\lambda_2=-8<0$ → **Máximo local**

$$f(0,0)=(0-1)^2+(0-2)^2=1+4=5$$

Punto $(0,\pm\sqrt{2})$

$$H_f(0,\pm\sqrt{2})=\begin{bmatrix}-4 & 0\\ 0 & 16\end{bmatrix}$$

Valores propios: $\lambda_1=-4<0$, $\lambda_2=16>0$ → **Punto silla**

$$f(0,\pm\sqrt{2})=(0-1)^2+(2-2)^2=1$$

Punto $(\pm 1,0)$

$$H_f(\pm 1,0)=\begin{bmatrix}8 & 0\\ 0 & -8\end{bmatrix}$$

Valores propios: $\lambda_1=8>0$, $\lambda_2=-8<0$ → **Punto silla**

$$f(\pm 1,0)=(1-1)^2+(0-2)^2=4$$

Punto $(\pm 1,\pm\sqrt{2})$

$$H_f(\pm 1,\pm\sqrt{2})=\begin{bmatrix}8 & 0\\ 0 & 16\end{bmatrix}$$

Valores propios: $\lambda_1=8>0$, $\lambda_2=16>0$ → **Mínimo local** (que también es **mínimo global**, como se demuestra en la Sección 9.1)

$$f(\pm 1,\pm\sqrt{2})=(1-1)^2+(2-2)^2=0$$

Resumen de la Clasificación

Clasificación de puntos estacionarios

Punto	Tipo	f(x,y)
$(0,0)$	Máximo local	5
$(0,\sqrt{2})$	Punto silla	1
$(0,-\sqrt{2})$	Punto silla	1
$(1,0)$	Punto silla	4
$(-1,0)$	Punto silla	4
$(1,\sqrt{2})$	**Mínimo global**	**0**
$(1,-\sqrt{2})$	**Mínimo global**	**0**
$(-1,\sqrt{2})$	**Mínimo global**	**0**
$(-1,-\sqrt{2})$	**Mínimo global**	**0**

Análisis del Óptimo

Mínimos Globales

La función tiene **cuatro mínimos globales** con el mismo valor:

$$x^{¿} \in \{(1,\sqrt{2}),(1,-\sqrt{2}),(-1,\sqrt{2}),(-1,-\sqrt{2})\}$$

$$f(x^{¿})=0$$

Demostración de que son mínimos globales

1. **Cota inferior**: Como $f(x,y)=(x^2-1)^2+(y^2-2)^2$ es la suma de dos términos al cuadrado, se cumple:

$$f(x,y)=(x^2-1)^2+(y^2-2)^2 \geq 0 \ \forall (x,y) \in R^2$$

   Por lo tanto, el valor mínimo posible de la función es 0.

2. **Alcanzabilidad**: Este valor mínimo de 0 se alcanza cuando ambos términos son simultáneamente cero:

$$(x^2-1)^2=0 \quad \Rightarrow x^2=1 \Rightarrow x=\pm 1$$
$$(y^2-2)^2=0 \quad \Rightarrow y^2=2 \Rightarrow y=\pm\sqrt{2}$$

   Esto produce exactamente los 4 puntos: $(\pm 1, \pm\sqrt{2})$.

3. **Conclusión**: Como $f(x^*)=0=inf_{(x,y)\in R^2} f(x,y)$ y este ínfimo se alcanza en los 4 puntos mencionados, estos son **mínimos globales**.

Implicación importante: Debido a que $f(x,y)\geq 0$ para todo $(x,y) \in R^2$, no pueden existir mínimos locales con valores mayores que 0. Cualquier punto estacionario que sea un mínimo local debe tener valor 0, y por tanto, es un mínimo global.

Interpretación Geométrica

Estos cuatro mínimos corresponden a las cuatro combinaciones de signos que satisfacen:

- $x^2=1 \Rightarrow x=\pm 1$
- $y^2=2 \Rightarrow y=\pm\sqrt{2}$

La existencia de múltiples mínimos globales es consistente con la estructura simétrica de la función y su no convexidad en ciertas regiones.

Características del Óptimo

- **Valor óptimo**: $f^*=0$
- **Número de soluciones óptimas**: 4 (simétricamente distribuidas)
- **Naturaleza**: Mínimos globales estrictos localmente (cada uno es el único mínimo en su vecindad)

Descripción de los Algoritmos Utilizados

Método del Descenso del Gradiente (Gradient Descent)

Descripción General

El método del Descenso del Gradiente es un algoritmo iterativo de optimización de primer orden que se mueve en la dirección opuesta al gradiente para encontrar un mínimo local de una función.

Fundamento Matemático

El gradiente $\nabla f(x)$ apunta en la dirección de mayor crecimiento de la función. Por lo tanto, el negativo del gradiente $-\nabla f(x)$ apunta en la dirección de mayor decrecimiento, lo que lo convierte en una dirección de descenso.

Algoritmo

Dado un punto inicial $x^{(0)}$ y una tasa de aprendizaje $\alpha > 0$:

1. **Inicialización**: $k = 0$, $x = x^{(0)}$
2. **Iteración**: Mientras $\| \nabla f(x^{(k)}) \| > \epsilon$ y $k < k_{max}$:

$$x^{(k+1)} = x^{(k)} - \alpha \nabla f(x^{(k)})$$

$$k = k + 1$$

3. **Terminación**: Retornar $x^{(k)}$ como aproximación del mínimo

Parámetros

- **learning_rate** (α): Controla el tamaño del paso en cada iteración
  - Valores muy grandes: Pueden causar divergencia u oscilaciones
  - Valores muy pequeños: Convergencia lenta
  - Típicamente: $0.001 \leq \alpha \leq 0.1$
- **tol** (ϵ): Tolerancia para el criterio de parada basado en la norma del gradiente
  - Cuando $\| \nabla f(x) \| < \epsilon$, se considera que se ha alcanzado un punto crítico
  - Típicamente: $10^{-6} \leq \epsilon \leq 10^{-4}$
- **max_iter** (k_{max}): Número máximo de iteraciones para evitar bucles infinitos

Ventajas

- Simple de implementar
- Bajo costo computacional por iteración

- Requiere solo el cálculo del gradiente (derivadas de primer orden)
- Garantiza descenso en cada iteración (con α apropiado)

Desventajas

- Convergencia puede ser lenta, especialmente cerca del óptimo
- Sensible a la elección de la tasa de aprendizaje
- Puede atascarse en mínimos locales o puntos silla
- No es eficiente para funciones mal condicionadas

Justificación de Selección

Se seleccionó este algoritmo porque:

1. **Simplicidad**: Es el método de optimización basado en gradiente más fundamental
2. **Referencia**: Sirve como línea base para comparar con métodos más sofisticados
3. **Interpretabilidad**: Su comportamiento es fácil de entender y visualizar
4. **Aplicabilidad**: Funciona bien para funciones suaves y diferenciables como nuestra función objetivo

Método Cuasi-Newton BFGS

Descripción General

BFGS (Broyden-Fletcher-Goldfarb-Shanno) es un método cuasi-Newton que aproxima la matriz Hessiana inversa para encontrar direcciones de búsqueda más eficientes que el gradiente puro.

Fundamento Matemático

Los métodos de Newton utilizan información de segundo orden (la Hessiana) para encontrar la dirección de búsqueda:

$$x^{(k+1)} = x^{(k)} - H_f^{-1}\left(x^{(k)}\right)\nabla f\left(x^{(k)}\right)$$

Sin embargo, calcular y invertir la Hessiana es costoso. BFGS construye iterativamente una aproximación B_k de H_f^{-1} usando solo evaluaciones del gradiente.

Algoritmo

1. **Inicialización**: $k=0$, $x^{(0)}$, $B_0 = I$ (matriz identidad)
2. **Iteración**: Para $k=0,1,2,\ldots$:
    a. Calcular dirección de búsqueda: $p_k = -B_k \nabla f\left(x^{(k)}\right)$

b. Búsqueda de línea: Encontrar α_k que minimice $f\left(x^{(k)}+\alpha_k p_k\right)$

c. Actualizar posición: $x^{(k+1)}=x^{(k)}+\alpha_k p_k$

d. Calcular diferencias:

- $s_k=x^{(k+1)}-x^{(k)}$
- $y_k=\nabla f\left(x^{(k+1)}\right)-\nabla f\left(x^{(k)}\right)$

e. Actualizar aproximación de la Hessiana inversa (fórmula BFGS):

$$B_{k+1}=B_k+\frac{\left(s_k^T y_k+y_k^T B_k y_k\right)\left(s_k s_k^T\right)}{\left(s_k^T y_k\right)^2}-\frac{B_k y_k s_k^T+s_k y_k^T B_k}{s_k^T y_k}$$

3. **Terminación**: Cuando $\| \nabla f\left(x^{(k)}\right) \| <\epsilon$ o $k \geq k_{max}$

Implementación

Se utiliza la implementación de `scipy.optimize.minimize` con `method='BFGS'`, que incluye:

- Búsqueda de línea robusta (condiciones de Wolfe)
- Manejo numérico estable de la actualización BFGS
- Criterios de convergencia sofisticados

Parámetros

- **tol**: Tolerancia para convergencia del gradiente
- **max_iter**: Número máximo de iteraciones
- No requiere especificar learning_rate (se determina automáticamente mediante búsqueda de línea)

Ventajas

- **Convergencia superlineal**: Mucho más rápida que el descenso del gradiente cerca del óptimo
- **Adaptativo**: La búsqueda de línea ajusta automáticamente el tamaño del paso
- **Curvatura**: Utiliza información de segundo orden sin calcular explícitamente la Hessiana
- **Eficiente**: Requiere menos iteraciones que métodos de primer orden
- **Robusto**: La implementación de SciPy incluye salvaguardas numéricas

Desventajas

- Más complejo de implementar desde cero
- Mayor costo computacional por iteración (actualización de B_k)
- Requiere almacenar una matriz $n \times n$ (donde n es la dimensión)
- Puede fallar si la función no es suficientemente suave

Justificación de Selección

Se seleccionó BFGS porque:

1. **Eficiencia**: Es uno de los métodos cuasi-Newton más efectivos y ampliamente utilizados
2. **Estado del arte**: Representa el estándar industrial para optimización no lineal sin restricciones
3. **Comparación**: Permite contrastar un método sofisticado de segundo orden con el simple descenso del gradiente
4. **Biblioteca**: La implementación en SciPy es robusta y bien probada
5. **Aplicabilidad**: Nuestra función es suave (clase C^{∞}), ideal para BFGS

Comparación de Resultados

Criterios de Comparación

Los métodos se comparan según:

1. **Número de iteraciones**: ¿Cuántos pasos requiere cada método para converger?
2. **Tiempo de ejecución**: ¿Cuánto tiempo toma la optimización?
3. **Valor final de la función**: ¿Qué tan cerca está del mínimo teórico?
4. **Punto final**: ¿A cuál de los cuatro mínimos globales converge?
5. **Sensibilidad al punto inicial**: ¿Cómo afecta $x^{(0)}$ al resultado?
6. **Impacto del learning rate**: (Solo para Gradient Descent) ¿Cómo afecta α a la convergencia?

Experimentos Diseñados

Nota: Los experimentos cubren el rango completo $[-100, 100)$ para ambas variables, según los requisitos del proyecto.

Configuración de Experimentos

Exp.	Punto Inicial	Learning Rate	Objetivo
exp1	$(0.5, 0.5)$	0.1	Región no convexa central
exp2	$(-1.5, -1.0)$	0.05	Región convexa
exp3	$(100, 100)$	0.05	Extremo superior derecho
exp4	$(-100, -100)$	0.05	Extremo inferior izquierdo
exp5	$(100, -100)$	0.05	Extremo inferior derecho
exp6	$(-100, 100)$	0.05	Extremo superior izquierdo
exp7	$(0.1, 0.1)$	0.05	Cerca del máximo local
exp8	$(0.05, \sqrt{2})$	0.03	Cerca de punto silla
exp9	$(1.0, 0.05)$	0.03	Cerca de otro punto silla
exp10	$(50, -75)$	0.05	Punto intermedio alejado

Resultados de los Experimentos

Resultados Experimentales Obtenidos

Tras ejecutar los 10 experimentos diseñados, se obtuvieron los siguientes resultados:

Tabla de Convergencia General

Exp.	Punto Inicial	GD Min	GD Iter	GD $f(x)$	BFGS Min	BFGS Iter
exp1	$(0.5, 0.5)$	1	31	2.4×10^{-14}	1	8
exp2	$(-1.5, -1.$	4	29	3.6×10^{-14}	4	10
exp3	**(100, 100)**	**DIV**	**5000**	**NaN**	**1**	**42**
exp4	**(-100, -100)**	**DIV**	**5000**	**NaN**	**4**	**42**
exp5	**(100, -**	**DIV**	**5000**	**NaN**	**2**	**42**

Exp.	Punto Inicial	GD Min	GD Iter	GD $f(x)$	BFGS Min	BFGS Iter
	100)					
exp6	**(-100, 100)**	**DIV**	**5000**	**NaN**	**3**	**42**
exp7	$(0.1, 0.1)$	1	44	3.8×10^{-14}	1	8
exp8	$(0.05, \sqrt{2})$	1	83	5.9×10^{-14}	1	6
exp9	$(1.0, 0.05)$	1	42	1.8×10^{-14}	1	4
exp10	**(50, -75)**	**DIV**	**4000**	**NaN**	**2**	**37**

Leyenda de Mínimos:

- Mínimo 1: $(1, \sqrt{2})$
- Mínimo 2: $(1, -\sqrt{2})$
- Mínimo 3: $(-1, \sqrt{2})$
- Mínimo 4: $(-1, -\sqrt{2})$
- DIV: Divergencia por overflow

Hallazgos Importantes

1. Falla Masiva del Gradient Descent en Puntos Alejados

El resultado más crítico del estudio: **Gradient Descent falló en 5 de 10 experimentos (50% tasa de fallo)**.

Experimentos con divergencia:

- exp3 $(100, 100)$: Divergencia por overflow
- exp4 $(-100, -100)$: Divergencia por overflow
- exp5 $(100, -100)$: Divergencia por overflow
- exp6 $(-100, 100)$: Divergencia por overflow
- exp10 $(50, -75)$: Divergencia por overflow

Patrón identificado: Todos los puntos iniciales con $|x) \geq 50$ o $|y) \geq 75$ causaron divergencia.

Causa raíz: El gradiente crece cúbicamente con la distancia:

$$\| \nabla f(x, y) \| \approx 4\sqrt{x^6 + y^6} \text{para} |x), |y) \gg 1$$

En $(100, 100)$: $\| \nabla f \| \approx 5.7 \times 10^7$

Con $\alpha = 0.05$, el paso es $\Delta x \approx 2.8 \times 10^6$, causando overflow explosivo.

Cálculo del learning rate óptimo teórico:

Para garantizar convergencia en Gradient Descent con learning rate fijo, se requiere que:

$$\alpha < \frac{2}{\lambda_{max}(H)}$$

donde $\lambda_{max}(H)$ es el mayor valor propio de la Hessiana en cualquier punto de la trayectoria.

En puntos extremos como $(100, 100)$:

$$\lambda_{max} = 12 \times 100^2 - 4 = 119996$$

Por lo tanto, para garantizar convergencia desde cualquier punto en $[-100, 100) \times [-100, 100)$:

$$\alpha < \frac{2}{119996} \approx 1.67 \times 10^{-5}$$

Implicación práctica:

- Con $\alpha = 1.67 \times 10^{-5}$, cada paso sería minúsculo
- Desde $(100, 100)$ hasta $(1, \sqrt{2})$ (distancia ≈ 140), se requerirían aproximadamente **8-10 millones de iteraciones**
- El tiempo de ejecución sería prohibitivo (días o semanas de cómputo)

Conclusión: Gradient Descent con learning rate fijo es **matemáticamente inviable** para el rango completo $[-100, 100)$. Se requiere obligatoriamente learning rate adaptativo.

Contraste dramático: BFGS convergió exitosamente en **TODOS** los casos, incluyendo los 5 donde GD falló.

2. Tasa de Éxito Real

Tasa de Éxito por Método

Método	Éxitos	Fallos	Tasa de Éxito
Gradient Descent	5/10	5/10	**50%**
BFGS	10/10	0/10	**100%**

Conclusión crítica: Gradient Descent simple **NO es confiable** para el rango $[-100, 100)$.

3. Eficiencia Comparativa (Solo Casos Exitosos)

Iteraciones:

- GD: Promedio = 54 iteraciones (solo 5 casos exitosos)
- BFGS: Promedio = 24 iteraciones (10 casos, todos exitosos)
- **BFGS es ~2.3× más rápido** cuando GD funciona
- **BFGS es infinitamente mejor** considerando las divergencias de GD

Precisión:

- GD: Mejor = 1.8×10^{-14} (solo casos exitosos)
- BFGS: Mejor = 6.7×10^{-19}
- **BFGS logra ~270× mejor precisión**

4. Comportamiento en Casos Problemáticos

Cerca del Máximo Local (exp7: $x_0 = (0.1, 0.1)$):

- GD: 44 iteraciones ✓ **EXITOSO**
- BFGS: 8 iteraciones ✓ **EXITOSO**
- Ambos escapan exitosamente del máximo local

Cerca de Puntos Silla (exp8, exp9):

- GD: 42-83 iteraciones ✓ **EXITOSO**
- BFGS: 4-6 iteraciones ✓ **EXITOSO**
- Los puntos silla no impiden convergencia, solo la ralentizan

Puntos Extremos (exp3-exp6: $|x| = 100$ o $|y| = 100$):

- GD: **100% DIVERGENCIA** × (4 de 4 experimentos)
- BFGS: **100% ÉXITO** ✓ (4 de 4 experimentos, 42 iteraciones)
- **Hallazgo crítico**: GD es **incapaz** de manejar puntos alejados con LR fijo

Visualización del Modelo y las Instancias de los Algoritmos

Gráficos de Contorno

Los gráficos de contorno muestran:

- **Curvas de nivel**: Líneas de igual valor de la función $f(x, y)$
- **Trayectoria**: Puntos visitados por cada algoritmo (conectados por líneas)

- **Mínimo teórico**: Marcado con una estrella verde

Interpretación Visual

- **Gradient Descent** (rojo): Trayectoria más larga, pasos más pequeños cerca del óptimo
- **BFGS** (azul): Trayectoria más corta y directa
- **Curvas de nivel**: Muestran la topología de la función, incluyendo los cuatro mínimos globales

Información Revelada

Los gráficos permiten visualizar:

1. La naturaleza no convexa de la función en ciertas regiones
2. La estructura simétrica con cuatro mínimos
3. La presencia de puntos silla y el máximo local en el origen
4. La eficiencia relativa de cada método
5. El comportamiento de convergencia en diferentes regiones del espacio

Conclusiones

Sobre la Función

- La función tiene estructura cuártica no convexa globalmente
- Existen 4 mínimos globales equivalentes y varios puntos silla
- La separabilidad simplifica el análisis pero la no convexidad introduce complejidad

Sobre los Métodos

- **Gradient Descent**: Simple pero ineficiente, adecuado para problemas simples o como método base
- **BFGS**: Superior en casi todos los aspectos, es la elección preferida para este tipo de problemas

Recomendaciones

Para minimizar funciones suaves no lineales:

1. Usar métodos cuasi-Newton (BFGS) cuando sea posible
2. Probar múltiples puntos iniciales para explorar diferentes cuencas de atracción

3. Considerar la topología de la función (convexidad, múltiples mínimos)
4. Ajustar cuidadosamente los hiperparámetros en métodos de primer orden

Consideraciones Finales

Este estudio demuestra la importancia de:

- Análisis teórico exhaustivo antes de aplicar algoritmos
- Comparación empírica de múltiples métodos
- Visualización para entender el comportamiento de los algoritmos
- Documentación clara de decisiones y resultados

Validación del Rango de Experimentación

Los experimentos realizados cubren el rango completo $[-100, 100)$ para ambas variables, conforme a los requisitos del proyecto:

Cobertura del espacio:

- Experimentos en los 4 cuadrantes
- Puntos extremos: $(\pm 100, \pm 100)$
- Puntos intermedios: diversos valores entre -100 y 100
- Casos problemáticos: cerca de puntos estacionarios no mínimos
- Regiones convexas y no convexas

Aclaraciones Terminológicas Importantes

Conforme a las definiciones estándar en optimización:

1. **Puntos críticos (en cálculo)**: Aquellos donde la derivada es indefinida (no existe). En este problema, la función es C^{∞} (infinitamente diferenciable), por lo que **NO existen puntos críticos**.
2. **Puntos estacionarios**: Aquellos donde el gradiente es cero ($\nabla f = 0$). En optimización con restricciones, también incluye puntos que satisfacen las condiciones de KKT. En este problema sin restricciones, coinciden con los puntos donde $\nabla f = 0$.
3. **Puntos críticos en optimización con restricciones**: Puntos donde la función objetivo es combinación lineal de las restricciones de igualdad y las restricciones de desigualdad activas (condiciones de KKT). **No aplica** a este problema por ser optimización sin restricciones.

En este informe se utiliza correctamente "puntos estacionarios" para referirse a los 9 puntos donde $\nabla f(x, y)=0$.

Lecciones Aprendidas de los Experimentos

Sobre la Robustez de los Algoritmos

Gradient Descent NO es robusto para puntos iniciales arbitrarios en $[-100, 100]$:

- **50% tasa de fallo** (5 de 10 experimentos divergieron)
- Falló en TODOS los puntos con $|x| \geq 50$ o $|y| \geq 50$
- Requiere learning rate adaptativo (no opcional, **necesario**)
- **Inaceptable para uso en producción** sin modificaciones

BFGS ES completamente robusto para todo el rango probado:

- **100% tasa de éxito** (10 de 10 experimentos convergieron)
- Auto-ajuste del tamaño de paso mediante búsqueda de línea
- Consistentemente eficiente independiente de la posición inicial
- **Recomendado para uso en producción**

Sobre el Learning Rate en Gradient Descent

Los experimentos revelaron la importancia **crítica y catastrófica** del learning rate.

Recomendaciones obligatorias para GD:

1. **Learning rate inversamente proporcional a la distancia**:

$$\alpha(x)=\frac{\alpha_0}{1+\| x \|^2} \text{ donde } \alpha_0 \approx 0.1$$

2. **Normalización del gradiente** (Gradient clipping):

$$\Delta x = -\alpha \frac{\nabla f}{max(1, \| \nabla f \| / M)} \text{ donde } M = 1000$$

3. **Búsqueda de línea** (como BFGS): Encontrar α que satisfaga condiciones de Wolfe

Sin estas modificaciones, GD es inutilizable para este problema.

Recomendación Final Basada en Evidencia

Para minimizar $f(x, y)=(x^2-1)^2+(y^2-2)^2$ con puntos iniciales en $[-100, 100]$:

Comparación Final de Métodos

Aspecto	Gradient Descent	BFGS
Tasa de éxito	50% (5/10)	100% (10/10)
Iteraciones (éxito)	54 promedio	24 promedio
Precisión	10^{-14}	10^{-19}
Robustez	Muy baja	Total
Implementación	Más simple	Más compleja
Necesita tuning	Sí (crítico)	No
Rango funcional	$\|x\),\|y\)<50$	Todo $[-100, 100)$

Veredicto: BFGS superior en 6 de 7 aspectos. La simplicidad de GD no compensa sus fallos masivos.

Referencias de Librerías y Documentación

Librerías Utilizadas

Este proyecto utilizó las siguientes librerías de Python para la implementación de algoritmos de optimización y análisis de resultados:

NumPy (Numerical Python)

Versión utilizada: 1.24.0 o superior

Propósito: Biblioteca fundamental para computación científica en Python, utilizada para:

- Manejo de arrays y matrices multidimensionales
- Operaciones algebraicas vectorizadas
- Cálculo de normas vectoriales (`np.linalg.norm`)
- Funciones matemáticas (exponenciales, trigonométricas, raíces)
- Generación de mallas de puntos para visualización

Funcionalidades específicas usadas:

- `np.array()`: Creación de vectores y matrices
- `np.linalg.norm()`: Cálculo de norma euclidiana del gradiente
- `np.linalg.eigvals()`: Cálculo de valores propios de la Hessiana
- `np.sqrt()`: Cálculo de raíces cuadradas
- `np.linspace()`, `np.meshgrid()`: Generación de mallas para visualización

- `np.argmin()`, `np.mean()`, `np.median()`: Estadísticas
- `np.isnan()`: Detección de valores NaN (divergencia)

Documentación oficial: https://numpy.org/doc/stable/

Referencia bibliográfica:

Harris, C.R., Millman, K.J., van der Walt, S.J. et al. (2020). Array programming with NumPy. *Nature*, 585, 357–362. DOI: 10.1038/s41586-020-2649-2

SciPy (Scientific Python)

Versión utilizada: 1.10.0 o superior

Propósito: Biblioteca para computación científica y técnica, construida sobre NumPy.

Módulo específico usado: `scipy.optimize`

Funcionalidades usadas:

- `scipy.optimize.minimize()`: Función de optimización de propósito general
  - `method='BFGS'`: Algoritmo cuasi-Newton BFGS
  - `jac=grad_f`: Gradiente analítico
  - `tol`: Tolerancia para convergencia
  - `options={'maxiter': max_iter}`: Número máximo de iteraciones
  - `callback`: Función para registrar trayectoria

Algoritmo BFGS implementado en SciPy: Sigue el esquema de Nocedal & Wright (2006), con:

- Búsqueda de línea que satisface las condiciones de Wolfe
- Manejo numérico estable de la actualización BFGS
- Reinicio automático si la aproximación de la Hessiana pierde definitud positiva

Documentación oficial: https://docs.scipy.org/doc/scipy/reference/optimize.html

Referencia bibliográfica:

Virtanen, P., Gommers, R., Oliphant, T.E. et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17, 261–272. DOI: 10.1038/s41592-019-0686-2

Matplotlib

Versión utilizada: 3.7.0 o superior

Propósito: Biblioteca para creación de visualizaciones estáticas, animadas e interactivas.

Módulos usados:

- `matplotlib.pyplot`: Interfaz tipo MATLAB para gráficos
- `mpl_toolkits.mplot3d.Axes3D`: Gráficos tridimensionales

Funcionalidades usadas:

- `plt.subplots()`: Creación de figuras con múltiples subgráficos
- `ax.contour()`: Gráficos de curvas de nivel
- `ax.plot()`: Trayectorias de optimización
- `ax.plot_surface()`: Superficie 3D de la función
- `ax.scatter()`: Marcado de puntos estacionarios
- Configuración de ejes, títulos, leyendas y rejillas

Documentación oficial: https://matplotlib.org/stable/contents.html

Referencia bibliográfica:

Hunter, J.D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90-95. DOI: 10.1109/MCSE.2007.55

Otras Librerías Estándar de Python

- **time**: Medición de tiempo de ejecución (`time.time()`)
- **json**: Serialización y deserialización de datos (`json.load()`, `json.dump()`)
- **pathlib**: Manejo de rutas de archivos (`Path()`, `Path.mkdir()`, `Path.glob()`)

Referencias Teóricas de los Algoritmos

Método BFGS (Broyden-Fletcher-Goldfarb-Shanno)

Referencia principal:

Nocedal, J., & Wright, S. J. (2006). *Numerical Optimization* (2nd ed.). Springer Series in Operations Research. ISBN: 978-0-387-30303-1

Capítulos relevantes:

- Capítulo 6: Quasi-Newton Methods
- Capítulo 3: Line Search Methods
- Sección 6.1: The BFGS Method

Artículos originales:

- Broyden, C.G. (1970). "The convergence of a class of double-rank minimization algorithms". *IMA Journal of Applied Mathematics*, 6(1), 76-90.
- Fletcher, R. (1970). "A new approach to variable metric algorithms". *The Computer Journal*, 13(3), 317-322.
- Goldfarb, D. (1970). "A family of variable-metric methods derived by variational means". *Mathematics of Computation*, 24(109), 23-26.
- Shanno, D.F. (1970). "Conditioning of quasi-Newton methods for function minimization". *Mathematics of Computation*, 24(111), 647-656.

Método del Descenso del Gradiente

Referencia principal:

> Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press. ISBN: 978-0-521-83378-3

Capítulos relevantes:

- Capítulo 9: Unconstrained minimization
- Sección 9.3: Gradient descent method

Referencia clásica:

> Cauchy, A. (1847). "Méthode générale pour la résolution des systèmes d'équations simultanées". *Comptes Rendus de l'Académie des Sciences*, 25, 536-538.

Implementación y Código Fuente

Repositorio del proyecto:
https://github.com/Rlianny/Optimization_Models_Project_2025-

Archivos principales:

- `Implementation/Methods_Implementation.ipynb`: Notebook Jupyter con implementación completa
- `Implementation/Experiments/exp*.json`: Configuraciones de experimentos
- `Implementation/Results/results_*.json`: Resultados de las ejecuciones

Lenguaje de programación: Python 3.7+

Entorno de desarrollo: Jupyter Notebook / VS Code

Recursos Adicionales

Tutoriales y documentación de SciPy Optimize:

- SciPy Lecture Notes - Optimization: https://scipy-lectures.org/advanced/mathematical_optimization/
- SciPy Optimize Tutorial: https://docs.scipy.org/doc/scipy/tutorial/optimize.html

Recursos sobre métodos cuasi-Newton:

- Wright, S.J., & Nocedal, J. (1999). "Numerical Optimization". Springer. (Texto fundamental)
- Optimization Methods - Stanford University: https://web.stanford.edu/class/ee364a/ (Curso de Stephen Boyd)

Licencias

- **NumPy**: BSD License
- **SciPy**: BSD License
- **Matplotlib**: PSF License (compatible con BSD)
- **Python**: PSF License

Todas las librerías utilizadas son de código abierto y permiten uso académico y comercial sin restricciones significativas.

Nota final sobre reproducibilidad: Todos los experimentos pueden ser reproducidos ejecutando el notebook `Methods_Implementation.ipynb` con los archivos de configuración proporcionados en la carpeta `Experiments/`. Los resultados están almacenados en formato JSON para máxima portabilidad y legibilidad.