Laboratorio 3: Newton–Raphson y Gradiente Descendente

Profesores:
Carlos Andrés Lozano,
Germán Adolfo Montoya,

Profesor de Laboratorio:
Juan Andrés Mendez

6 de marzo de 2025

1 Objetivo

El propósito de este laboratorio es introducir a los estudiantes en técnicas avanzadas de optimización no lineal, implementando desde cero métodos iterativos tales como Newton–Raphson y Gradiente Descendente. A través de ejercicios prácticos, los estudiantes desarrollarán una comprensión profunda de estos algoritmos fundamentales para el cálculo multivariable y la optimización computacional.

2 Fundamentación Matemática

2.1 Teoría de Optimización No Lineal

La optimización no lineal constituye una rama fundamental del análisis matemático que busca encontrar extremos (mínimos o máximos) de funciones continuas no lineales, sujetas o no a restricciones. Los métodos iterativos representan un enfoque sistemático para aproximar soluciones cuando no existe una solución analítica directa.

2.2 Método de Newton-Raphson

2.2.1 Fundamento Teórico

El método de Newton-Raphson se fundamenta en la expansión de Taylor de segundo orden de una función en la vecindad de un punto. Para una función $f : \mathbb{R}^n \to \mathbb{R}$ de clase $\mathcal{C}^2$, su aproximación cuadrática alrededor de un punto $\mathbf{x}_k$ viene dada por:

Expansión de Taylor de Segundo Orden

$$f(\mathbf{x}) \approx f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T(\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^T H(f(\mathbf{x}_k))(\mathbf{x} - \mathbf{x}_k) \tag{1}$$

donde $\nabla f(\mathbf{x}_k)$ es el gradiente y $H(f(\mathbf{x}_k))$ es la matriz Hessiana evaluada en $\mathbf{x}_k$.

En un extremo local, el gradiente se anula. El método de Newton-Raphson encuentra estos puntos mediante la resolución iterativa de $\nabla f(\mathbf{x}) = \mathbf{0}$. Diferenciando la aproximación cuadrática e igualando a cero:

$$\nabla f(\mathbf{x}_k) + H(f(\mathbf{x}_k))(\mathbf{x} - \mathbf{x}_k) = \mathbf{0} \tag{2}$$

$$\mathbf{x} - \mathbf{x}_k = -H(f(\mathbf{x}_k))^{-1}\nabla f(\mathbf{x}_k) \tag{3}$$

$$\mathbf{x} = \mathbf{x}_k - H(f(\mathbf{x}_k))^{-1}\nabla f(\mathbf{x}_k) \tag{4}$$

2.2.2 Formulación Algorítmica

Método de Newton-Raphson para Encontrar Extremos

El método de Newton-Raphson, cuando se aplica para encontrar extremos de una función $f(x)$, se basa en hallar los puntos donde la derivada se anula $f'(x) = 0$. Por tanto, aplicamos Newton-Raphson a la función $g(x) = f'(x)$.
La iteración se define como:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

donde:

- $f'(x_k)$ es la primera derivada de f evaluada en x_k (buscamos donde es cero)
- $f''(x_k)$ es la segunda derivada de f evaluada en x_k (determina la dirección y magnitud del paso)

Interpretación geométrica: En cada iteración, construimos una parábola osculatriz (aproximación cuadrática) a la función en el punto actual y hallamos donde esta parábola alcanza su extremo.
Clasificación del extremo: Una vez que el método converge a un punto x^* donde $f'(x^*) \approx 0$, podemos clasificarlo como:

- Si $f''(x^*) > 0$: Mínimo local
- Si $f''(x^*) < 0$: Máximo local
- Si $f''(x^*) = 0$: Punto de inflexión o requiere análisis adicional

Ejemplo: Para la función $f(x) = 3x^3 - 10x^2 - 56x + 50$:
Calculamos:

$$f'(x) = 9x^2 - 20x - 56 \tag{5}$$
$$f''(x) = 18x - 20 \tag{6}$$

La iteración de Newton-Raphson queda:

$$x_{k+1} = x_k - \frac{9x_k^2 - 20x_k - 56}{18x_k - 20}$$

Para diferentes valores iniciales x_0 en el intervalo $[-6, 6]$, esta iteración convergerá a los distintos extremos de la función. La convergencia será cuadrática cuando estemos cerca de la solución, lo que significa que el número de dígitos correctos aproximadamente se duplica en cada iteración.
Un factor de convergencia α puede introducirse para controlar la magnitud del paso:

$$x_{k+1} = x_k - \alpha\frac{f'(x_k)}{f''(x_k)}$$

Valores de $\alpha < 1$ pueden mejorar la estabilidad cuando estamos lejos de la solución, a costa de una convergencia más lenta.

Método de Newton-Raphson

Entrada: Función f, punto inicial $\mathbf{x}_0$, factor de paso α, tolerancia ϵ, número máximo de iteraciones N_{max}.
Salida: Aproximación al extremo local $\mathbf{x}^*$

1. **Inicializar:** $k \leftarrow 0$
2. **Repetir:**
   a) Calcular $\nabla f(\mathbf{x}_k)$
   b) Calcular $H(f(\mathbf{x}_k))$
   c) Verificar si $H(f(\mathbf{x}_k))$ es invertible
   d) Calcular $\mathbf{d}_k = -H(f(\mathbf{x}_k))^{-1}\nabla f(\mathbf{x}_k)$
   e) Actualizar $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{d}_k$
   f) $k \leftarrow k + 1$
3. **Hasta que** $\|\nabla f(\mathbf{x}_k)\| < \epsilon$ o $k > N_{max}$
4. **Retornar** $\mathbf{x}_k$

2.2.3 Caracterización de Extremos

Para determinar si un punto crítico $\mathbf{x}^*$ (donde $\nabla f(\mathbf{x}^*) = \mathbf{0}$) es un mínimo, máximo o punto de silla, se analiza la matriz Hessiana:

Criterio de Segunda Derivada

Sea $\mathbf{x}^*$ un punto crítico de $f : \mathbb{R}^n \to \mathbb{R}$ y $H(\mathbf{x}^*)$ la matriz Hessiana evaluada en $\mathbf{x}^*$:

- Si $H(\mathbf{x}^*)$ es definida positiva, entonces $\mathbf{x}^*$ es un mínimo local.
- Si $H(\mathbf{x}^*)$ es definida negativa, entonces $\mathbf{x}^*$ es un máximo local.
- Si $H(\mathbf{x}^*)$ tiene valores propios positivos y negativos, entonces $\mathbf{x}^*$ es un punto de silla.

2.2.4 Análisis de Convergencia

La tasa de convergencia del método de Newton-Raphson es cuadrática en la vecindad de la solución, lo que significa:

Convergencia Cuadrática

Si f es suficientemente suave y $\mathbf{x}_0$ está suficientemente cerca de $\mathbf{x}^*$, entonces existe una constante $M > 0$ tal que:

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \leq M\|\mathbf{x}_k - \mathbf{x}^*\|^2 \tag{7}$$

Esta convergencia cuadrática hace que el método sea extraordinariamente eficiente cuando se encuentra cerca de la solución. Sin embargo, puede diverger si el punto inicial

está alejado del extremo o si la Hessiana es mal condicionada.

2.3 Método de Gradiente Descendente

2.3.1 Fundamento Teórico

El método de gradiente descendente se basa en el principio de que, para una función diferenciable $f : \mathbb{R}^n \to \mathbb{R}$, el gradiente $\nabla f(\mathbf{x})$ apunta en la dirección de máximo incremento de f. Por lo tanto, $-\nabla f(\mathbf{x})$ indica la dirección de máximo decremento.

Dirección de Descenso

Sea f diferenciable en $\mathbf{x}$. Si $\nabla f(\mathbf{x}) \neq \mathbf{0}$, entonces $\mathbf{d} = -\nabla f(\mathbf{x})$ es una dirección de descenso, es decir:

$$\nabla f(\mathbf{x})^T \mathbf{d} < 0 \tag{8}$$

2.3.2 Formulación Algorítmica

Método de Gradiente Descendente

Entrada: Función f, punto inicial $\mathbf{x}_0$, tamaño de paso α, tolerancia ϵ, número máximo de iteraciones N_{max}.
Salida: Aproximación al mínimo local $\mathbf{x}^*$

1. **Inicializar:** $k \leftarrow 0$
2. **Repetir:**
   a) Calcular $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$
   b) Actualizar $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \mathbf{g}_k$
   c) $k \leftarrow k + 1$
3. **Hasta que** $\|\nabla f(\mathbf{x}_k)\| < \epsilon$ o $k > N_{max}$
4. **Retornar** $\mathbf{x}_k$

2.3.3 Análisis de Convergencia

Para funciones con gradiente Lipschitz continuo, el método de gradiente descendente con tamaño de paso fijo adecuado converge linealmente:

Convergencia Lineal

Si ∇f es Lipschitz continuo con constante L y f está acotada inferiormente, entonces para $\alpha \in (0, \frac{2}{L})$, existe una constante $\rho \in (0, 1)$ tal que:

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*) \leq \rho(f(\mathbf{x}_k) - f(\mathbf{x}^*)) \tag{9}$$

donde $\mathbf{x}^*$ es un mínimo local de f.

2.3.4 Variantes y Extensiones

Existen numerosas variantes del gradiente descendente que mejoran su eficiencia:

- **Gradiente Descendente con Momento:** Incorpora ïnercia.ªl movimiento mediante un término de momento, acelerando la convergencia en direcciones consistentes.
- **Gradiente Descendente con Tamaño de Paso Adaptativo:** Ajusta dinámicamente el tamaño del paso para mejorar la convergencia.
- **Métodos de Segundo Orden (Newton-Raphson):** Utilizan información de la curvatura para adaptar la dirección de búsqueda.

2.4 Comparación entre Newton-Raphson y Gradiente Descendente

Característica	**Newton-Raphson**	**Gradiente Descendente**
Tasa de convergencia	Cuadrática cerca del óptimo	Lineal
Costo computacional por iteración	Alto (cálculo e inversión de la Hessiana)	Bajo (solo cálculo del gradiente)
Sensibilidad al condicionamiento	Alta (problemas si la Hessiana es mal condicionada)	Moderada
Robustez global	Baja (puede diverger si está lejos de la solución)	Alta (convergencia asegurada con paso adecuado)
Aplicabilidad a problemas de gran escala	Limitada	Excelente

Cuadro 1: Comparación de los métodos Newton-Raphson y Gradiente Descendente

3 Problemas

3.1 Problema 1: Newton-Raphson en 2D para Polinomios Cúbicos

3.1.1 Descripción del Problema

Implementar el método de Newton-Raphson para encontrar extremos locales de la función polinómica

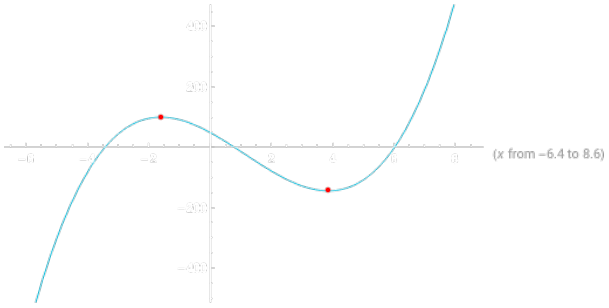

Figura 1: Local minima and maxima

$$f(x) = 3x^3 - 10x^2 - 56x + 50 \tag{10}$$

dentro del intervalo $[-6, 6]$, identificando tanto mínimos como máximos locales.

3.1.2 Instrucciones

1. Implementar el algoritmo de Newton-Raphson para funciones unidimensionales.
2. Calcular analíticamente la primera y segunda derivada de $f(x)$.
3. Experimentar con diferentes valores iniciales x_0 en el intervalo $[-6, 6]$.
4. Utilizar diferentes valores para el factor de convergencia α (por ejemplo, $\alpha = 0{,}6$).
5. Graficar la función junto con los puntos encontrados, destacando mínimos y máximos.
6. Analizar el comportamiento de la convergencia para diferentes valores iniciales.

3.2 Problema 2: Análisis de Extremos Locales y Globales

3.2.1 Descripción del Problema

Utilizar el método de Newton-Raphson para encontrar todos los extremos locales de la función

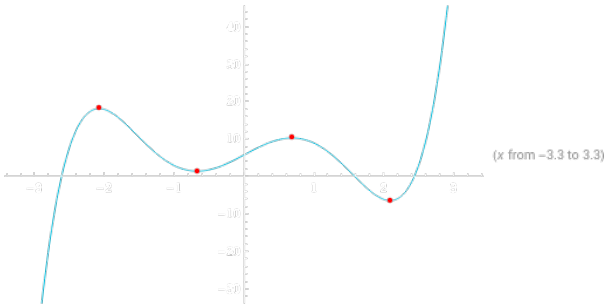

Figura 2: Critical points

$$f(x) = x^5 - 8x^3 + 10x + 6 \tag{11}$$

en el intervalo $[-3, 3]$, e identificar el máximo global y el mínimo global.

3.2.2 Instrucciones

1. Calcular analíticamente la primera y segunda derivada de $f(x)$.
2. Aplicar el método de Newton-Raphson desde diferentes puntos iniciales para encontrar todos los posibles extremos.
3. Clasificar los puntos encontrados como mínimos o máximos locales.
4. Identificar entre todos los extremos el máximo global y el mínimo global.
5. Graficar la función con todos los extremos locales (en negro) y destacar el máximo global y el mínimo global (en rojo).
6. Analizar la convergencia del método para esta función.

3.3 Problema 3: Newton-Raphson Multidimensional

3.3.1 Parte a: Función de Rosenbrock (3D)

Descripción del Problema Aplicar el método de Newton-Raphson para encontrar el mínimo de la función de Rosenbrock

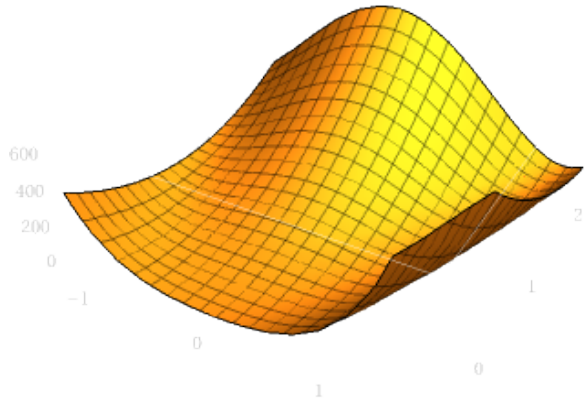

Figura 3: NR-3d surface

$$f(x, y) = (x - 1)^2 + 100(y - x^2)^2 \tag{12}$$

interpretada como una superficie en $\mathbb{R}^3$.

Instrucciones

1. Calcular analíticamente el gradiente y la matriz Hessiana de $f(x, y)$.
2. Implementar el método de Newton-Raphson para funciones bidimensionales.
3. Utilizar como punto inicial $(x_0, y_0) = (0, 10)$.
4. Graficar la superficie $z = f(x, y)$ en el espacio tridimensional.
5. Representar los puntos iterativos sobre la superficie y destacar el mínimo final en color rojo.
6. Analizar la convergencia hacia el mínimo conocido de la función (1,1).

3.3.2 Parte b: Función en 4D

Descripción del Problema Diseñar una función en cuatro dimensiones, por ejemplo,

$$f(x, y, z) = (x - 1)^2 + (y - 2)^2 + (z - 3)^2 \tag{13}$$

y aplicar el método de Newton-Raphson para encontrar su mínimo global.

Instrucciones

1. Formular matemáticamente el algoritmo de Newton-Raphson en $\mathbb{R}^4$.
2. Calcular analíticamente el gradiente y la matriz Hessiana de la función propuesta.
3. **Implementar el método iterativo para funciones tetradimensionales desde cero en Python**. Solo se permite el uso de NumPy para operaciones matemáticas básicas, SymPy para cálculos simbólicos, y Matplotlib para visualización. **Está prohibido utilizar librerías avanzadas o funciones predefinidas que resuelvan directamente el problema**. El incumplimiento de esta restricción resultará en una penalización significativa en la calificación.
4. Definir un criterio de parada basado en la norma del gradiente (e.g., $\|\nabla f(\mathbf{x}_k)\| < \epsilon$).
5. Representar la convergencia mediante proyecciones bidimensionales o tridimensionales (opcional) utilizando Matplotlib.
6. Discutir las dificultades computacionales específicas del problema en alta dimensión.
7. Entregar el código Python completo (.py) con comentarios detallados que expliquen cada etapa del algoritmo implementado.

3.4 Problema 4: Gradiente Descendente en Optimización

3.4.1 Parte A: Implementación de Gradiente descendente en 3-D

Descripción del Problema Implementar el método de Gradiente Descendente para minimizar la función de pérdida

$$L(x, y) = (x - 2)^2 + (y + 1)^2 \tag{14}$$

simulando el entrenamiento de una red neuronal con dos parámetros.

3.4.2 Instrucciones

1. Calcular analíticamente el gradiente de la función de pérdida.
2. Implementar el algoritmo de Gradiente Descendente.
3. Experimentar con diferentes valores para el parámetro de paso α.
4. Graficar la trayectoria de los parámetros durante la optimización.
5. Destacar el valor óptimo final y compararlo con la solución analítica.
6. Analizar la sensibilidad del método al valor de α y discutir estrategias para su selección óptima.

3.4.3 Parte b: Comparación entre Newton-Raphson y Gradiente Descendente

Descripción del Problema Aplicar tanto el método de Newton-Raphson como el de Gradiente Descendente para minimizar la función

$$f(x, y) = (x - 2)^2(y + 2)^2 + (x + 1)^2 + (y - 1)^2 \tag{15}$$

partiendo del mismo punto inicial $(x_0, y_0) = (-2, -3)$. Esta función presenta curvatura variable y un valle estrecho, lo que permite evidenciar las diferencias en convergencia de ambos métodos.

Instrucciones

1. Calcular analíticamente el gradiente y la matriz Hessiana de la función propuesta.
2. Implementar ambos algoritmos (Newton-Raphson y Gradiente Descendente) para la misma función.
3. Utilizar el mismo punto inicial $(x_0, y_0) = (-2, -3)$ para ambos métodos.
4. Para cada método, experimentar con diferentes valores del parámetro de paso α y determinar el valor óptimo.
5. Graficar en una misma figura las trayectorias de convergencia de ambos métodos, superpuestas sobre los contornos de la función objetivo.
6. Realizar un análisis comparativo considerando:

- Número de iteraciones hasta la convergencia
- Tiempo de ejecución
- Precisión final del resultado
- Robustez frente a diferentes valores del parámetro de paso
- Costo computacional por iteración

7. Concluir cuál método es más adecuado para esta función específica y argumentar bajo qué circunstancias generales sería preferible uno u otro método.

8. Presentar una tabla comparativa que sintetice las ventajas y desventajas observadas para cada método en este problema particular.

La visualización debería incluir:

- Mapa de contorno de la función objetivo
- Trayectorias de ambos algoritmos (con marcadores diferentes)
- Punto inicial y puntos finales claramente identificados
- Convergencia del error (distancia al óptimo) en escala logarítmica para ambos métodos

3.5 Problema 5: Descenso de Gradiente y Descenso de Gradiente Basado en Momento

3.5.1 Descripción del Problema

Este ejercicio explora los fundamentos del Descenso de Gradiente, sus limitaciones y aplicaciones en redes neuronales. Se enfoca en la optimización matemática para enriquecer la comprensión de una de las técnicas de optimización más efectivas utilizadas en el aprendizaje automático.

Se estudiarán los siguientes aspectos:

- **Descenso de Gradiente:** Algoritmo de optimización iterativo de primer orden para encontrar un mínimo local de una función diferenciable.
- **Limitaciones del Descenso de Gradiente:** Desafíos y limitaciones que presenta el algoritmo básico.
- **Aplicación en Redes Neuronales:** Empleo del Descenso de Gradiente para entrenar redes neuronales, mejorando iterativamente los parámetros del modelo para minimizar la función de pérdida.
- **Descenso de Gradiente Basado en Momento:** Variante que facilita una convergencia más rápida hacia el mínimo global y estabiliza las actualizaciones.

El objetivo principal es implementar el algoritmo de Descenso de Gradiente de tal manera que una red neuronal aprenda a aproximar la función seno.

3.5.2 Instrucciones

1. **Clonar el Repositorio:** Inicie clonando el repositorio `https://github.com/jabandersnatch/gradient-descent` en su máquina local para acceder al cuaderno Jupyter y otros recursos.
2. **Instalar las Dependencias:** Asegúrese de tener Python instalado, junto con bibliotecas como NumPy y Matplotlib, para ejecutar el cuaderno y los scripts.
3. **Explorar el Cuaderno:** Abra el cuaderno `neural_net_gradient_descent.ipynb` en Jupyter y siga las instrucciones para aprender e implementar los algoritmos.
4. **Implementar el Descenso de Gradiente básico:** Siguiendo las indicaciones del cuaderno, implemente el algoritmo de Descenso de Gradiente para entrenar una red neuronal simple.
5. **Implementar el Descenso de Gradiente Basado en Momento:** Extienda su implementación para incluir el concepto de momento, facilitando una convergencia más rápida.
6. **Analizar y Comparar:** Compare el rendimiento de ambos métodos en términos de velocidad de convergencia y calidad de la aproximación a la función seno.
7. **Entregar:** Presente un informe detallando su implementación, los resultados obtenidos y un análisis comparativo entre ambos métodos.

Recursos Proporcionados:

- **Cuaderno Jupyter:** `neural_net_gradient_descent.ipynb`, que incluye instrucciones paso a paso y celdas de código.
- **Script de Python:** Disponible para aquellos que prefieren un enfoque de codificación más tradicional.

4 Entregables y Criterios de Calificación

4.1 Entregables

Para cada ejercicio, se deberá entregar un archivo Python (*.py) que contenga:

1. Implementación del algoritmo correspondiente.
2. Comentarios explicativos detallando la lógica del código.
3. Visualizaciones gráficas de los resultados.
4. Análisis breve de los resultados obtenidos.

Adicionalmente, se deberá incluir un informe en formato PDF que contenga:

1. Formulación matemática de los problemas.
2. Descripción de la implementación.
3. Análisis de resultados, incluyendo las gráficas generadas.
4. Conclusiones y observaciones.

Tener en cuenta:

- **No es permitido** usar paquetes o funciones que realicen lo solicitado. Por ejemplo, instalar una librería en donde se llama el método *NewtonRaphson*, el cual recibe la función a analizar y encuentra los minimos o máximos. Las soluciones solicitadas **deben implementar los pseudocódigos suministrados por el laboratorio o la clase magistral.**
- No se reciben entregas por fuera del plazo máximo y tampoco por correo. Las entregas solo se reciben por **Bloque Neón**.
- Este laboratorio se puede entregar en parejas.
- **IMPORTANTE!!!** Junto a los archivos de la entrega adjuntar los nombres de los integrantes del equipo.

4.2 Criterios de Calificación

Criterio	**Puntuación**
Implementación correcta de los algoritmos	40 %
Calidad del código y comentarios	15 %
Visualizaciones y análisis gráfico	20 %
Análisis matemático y conclusiones	20 %
Originalidad y propuestas adicionales	5 %

Cuadro 2: Criterios de calificación del laboratorio

5 Referencias

- Nocedal, J., & Wright, S. J. (2006). *Numerical Optimization*. Springer.
- Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.
- Documentación de **Sympy**: `https://docs.sympy.org`
- Documentación de **NumPy**: `https://numpy.org/doc/`
- Documentación de **Matplotlib**: `https://matplotlib.org/stable/contents.html`