

Método de Optimización de Nelder-Mead

Optimización sin Derivadas

Modelos de Optimización para Aplicaciones Forestales

5/12/2025

- 1 Introducción y Contexto
- 2 Conceptos Fundamentales
- 3 El Método de Nelder-Mead
- 4 Algoritmo Completo
- 5 Propiedades y Análisis

Definición

La optimización busca encontrar el mejor elemento (mínimo o máximo) de un conjunto de alternativas disponibles.

Problema general:

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x})$$

donde $f : \mathbb{R}^n \rightarrow \mathbb{R}$ es la función objetivo.

En caso que la función objetivo tenga un gradiente muy complicado de evaluar, se puede aplicar un método sin derivadas, que no es otra cosa que un conjunto de reglas de evaluación y búsqueda de puntos en el dominio, hasta encontrar un mínimo. Este conjunto de reglas no nos garantizan que encontremos el mínimo absoluto de la función objetivo.

Origen

Desarrollado en 1965 por John Nelder y Roger Mead^a en el artículo:
A simplex method for function minimization

^aNational Vegetable Research Station, UK

Características históricas:

- Uno de los primeros métodos de optimización directa
- Diseñado para computadoras de la época (recursos limitados)
- Extremadamente popular hasta el día de hoy
- Implementado en prácticamente todo software científico
- Sencillo de hacer una implementación propia en cualquier lenguaje de programación

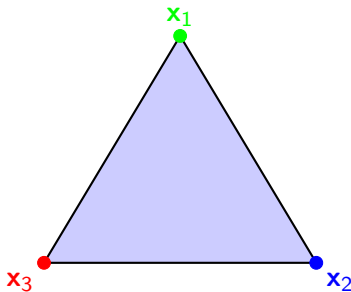
Definition

Un **símplex** en \mathbb{R}^n es la envolvente convexa de $n + 1$ puntos afínmente independientes.

Símplex en 2D

Ejemplos visuales:

- \mathbb{R}^1 : Segmento (2 puntos)
- \mathbb{R}^2 : Triángulo (3 puntos)
- \mathbb{R}^3 : Tetraedro (4 puntos)
- \mathbb{R}^n : $(n + 1)$ vértices



Ordenamiento de Vértices

En cada iteración, los vértices se ordenan según el valor de la función:

$$f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \cdots \leq f(\mathbf{x}_n) \leq f(\mathbf{x}_{n+1})$$

Nomenclatura

- \mathbf{x}_1 : **Mejor vértice** (mínimo valor de f)
- \mathbf{x}_{n+1} : **Peor vértice** (máximo valor de f)
- \mathbf{x}_n : **Segundo peor vértice**

Centroide

El centroide de los n mejores puntos:

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$$

Estrategia

El método mantiene un símplex que se **mueve**, **expande** y **contrae** en el espacio de búsqueda, intentando desplazarse hacia regiones con menores valores de f .

Principio geométrico:

- 1 Identificar el peor vértice del símplex
- 2 Intentar reemplazarlo por uno mejor
- 3 Usar operaciones geométricas simples
- 4 Repetir hasta converger

El símplex evoluciona por el espacio de soluciones buscando el óptimo

Las Cuatro Operaciones Principales

- ➊ **Reflexión** ($\alpha = 1$): Reflejar el peor punto respecto al centroide
- ➋ **Expansión** ($\gamma = 2$): Si la reflexión es muy buena, expandir más
- ➌ **Contracción** ($\beta = 0,5$): Si la reflexión falla, contraer
- ➍ **Reducción** ($\delta = 0,5$): Si todo falla, reducir todo el simplex

Parámetros estándar

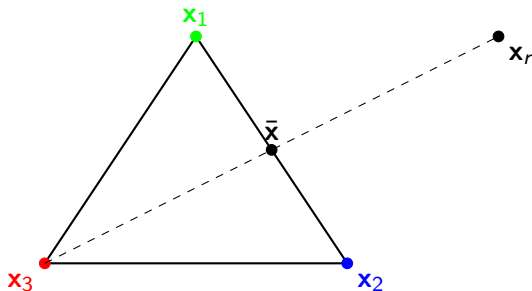
$$\alpha = 1 \quad \gamma = 2 \quad \beta = 0,5 \quad \delta = 0,5$$

Operación 1: Reflexión

Objetivo: Reflejar el peor punto a través del centroide de los restantes n-puntos

$$\mathbf{x}_r = \bar{\mathbf{x}} + \alpha(\bar{\mathbf{x}} - \mathbf{x}_{n+1})$$

donde $\alpha > 0$ (típicamente $\alpha = 1$)

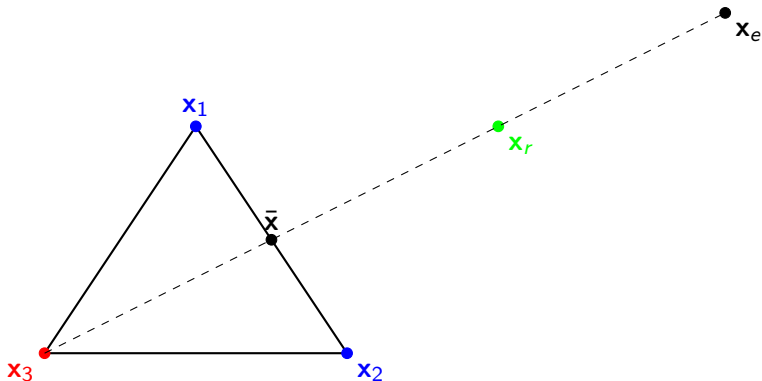


Operación 2: Expansión

Objetivo: Si la reflexión es muy exitosa, expandir aún más en esa dirección

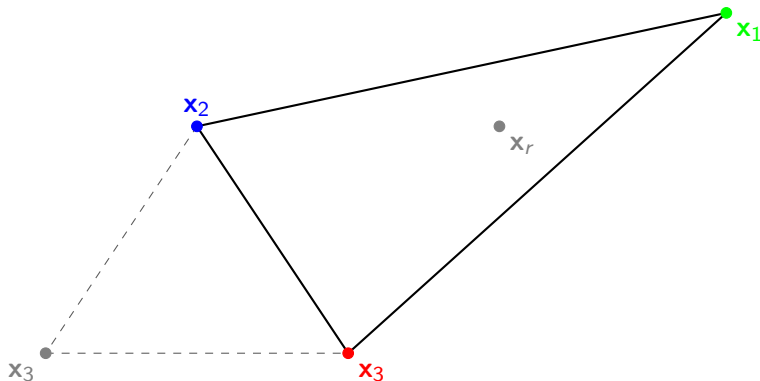
$$\mathbf{x}_e = \bar{\mathbf{x}} + \gamma(\mathbf{x}_r - \bar{\mathbf{x}})$$

donde $\gamma > 1$ (típicamente $\gamma = 2$)



Operación 2: Expansión

Se acepta la expansión cuando: $f(\mathbf{x}_r) < f(\mathbf{x}_1)$. Caso contrario, se acepta el punto reflejado



Operación 3: Contracción

Objetivo: Si la reflexión no mejora suficientemente, contraer el símplex

Hay dos tipos:

Contracción Externa

Cuando $f(\mathbf{x}_r) < f(\mathbf{x}_{n+1})$:

$$\mathbf{x}_c = \bar{\mathbf{x}} + \beta(\mathbf{x}_r - \bar{\mathbf{x}})$$

Contracción Interna

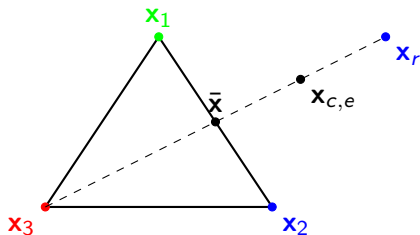
Cuando $f(\mathbf{x}_r) \geq f(\mathbf{x}_{n+1})$:

$$\mathbf{x}_c = \bar{\mathbf{x}} - \beta(\bar{\mathbf{x}} - \mathbf{x}_{n+1})$$

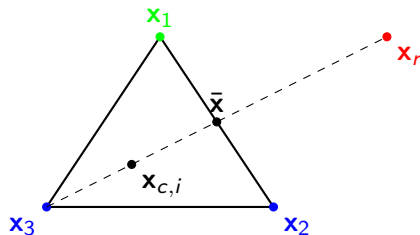
donde $0 < \beta < 1$ (típicamente $\beta = 0,5$)

Operación 3: Contracción

Contracción externa

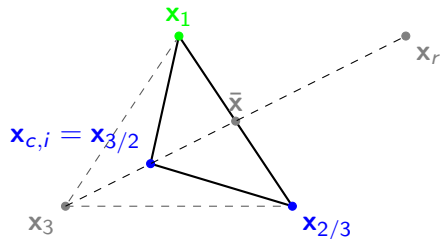
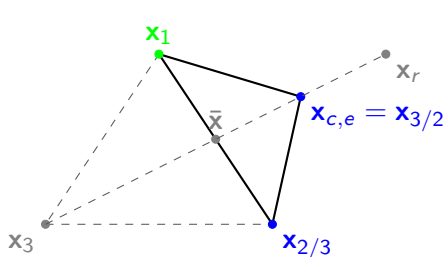


Contracción interna



Operación 3: Contracción

si $f(x_c) < f(x_{n+1})$ se acepta la contracción



Operación 4: Reducción (Shrinkage)

Objetivo: Cuando todas las operaciones anteriores fallan, reducir todo el símplex hacia el mejor punto

$$\mathbf{x}_i = \mathbf{x}_1 + \delta(\mathbf{x}_i - \mathbf{x}_1), \quad i = 2, \dots, n + 1$$

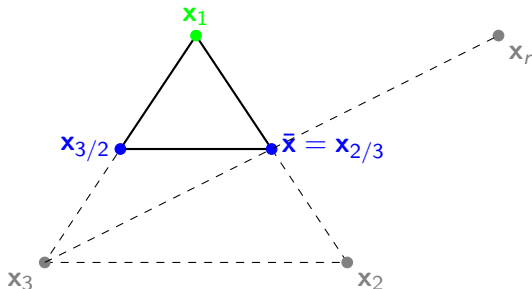
donde $0 < \delta < 1$ (típicamente $\delta = 0,5$)

Observación

Esta es la operación más costosa computacionalmente porque requiere evaluar f en n nuevos puntos (todos los vértices excepto \mathbf{x}_1).

El símplex se hace más pequeño, concentrándose alrededor del mejor punto

Operación 4: Reducción (Shrinkage)



Algoritmo de Nelder-Mead

- 1: **Inicializar:** Construir símplex inicial
- 2: **repeat**
- 3: Ordenar: $f(\mathbf{x}_1) \leq \dots \leq f(\mathbf{x}_{n+1})$
- 4: Calcular: $\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$
- 5: **Reflexión:** $\mathbf{x}_r = \bar{\mathbf{x}} + \alpha(\bar{\mathbf{x}} - \mathbf{x}_{n+1})$
- 6: **if** $f(\mathbf{x}_r) < f(\mathbf{x}_1)$ **then**
- 7: Intentar **Expansión**
- 8: **else if** $f(\mathbf{x}_1) \leq f(\mathbf{x}_r) < f(\mathbf{x}_n)$ **then**
- 9: Aceptar reflexión
- 10: **else**
- 11: Intentar **Contracción**
- 12: **if** Contracción falla **then**
- 13: Aplicar **Reducción**
- 14: **end if**
- 15: **end if**
- 16: **until** Criterio de parada
- 17: **Retornar:** $\mathbf{x}^* \approx \mathbf{x}_1$

El algoritmo se detiene cuando se cumple alguna de estas condiciones:

❶ **Tamaño del símplex pequeño:**

$$\max_{i=2,\dots,n+1} \|\mathbf{x}_i - \mathbf{x}_1\| < \epsilon$$

❷ **Variación de función objetivo pequeña:**

$$\sqrt{\frac{1}{n+1} \sum_{i=1}^{n+1} (f(\mathbf{x}_i) - \bar{f})^2} < \epsilon$$

❸ **Número máximo de iteraciones:**

$$k > k_{max}$$

donde ϵ es una tolerancia pequeña (ej: 10^{-6})

Principales ventajas

- **No requiere derivadas**
 - Solo necesita evaluar $f(\mathbf{x})$
 - Ideal para problemas de caja negra
- **Simplicidad**
 - Fácil de entender e implementar
 - Pocas líneas de código
- **Robusto**
 - Maneja funciones con ruido
 - Tolerante a discontinuidades menores
- **Bajo costo por iteración**
 - Típicamente 1-2 evaluaciones de f por iteración
 - Solo n evaluaciones en reducción

Principales desventajas

- **Sin garantías de convergencia**
 - Puede fallar en problemas mal condicionados
 - Existen contraejemplos documentados
- **Degeneración del símplex**
 - El símplex puede colapsar a dimensión menor
 - Pierde capacidad de exploración
- **Convergencia lenta**
 - Especialmente cerca del óptimo
 - Mucho más lento que métodos con derivadas
- **Sensibilidad al símplex inicial**
 - La elección inicial afecta el resultado
 - Puede quedar atrapado en mínimos locales

Evaluaciones de función por iteración

- **Reflexión aceptada:** 1 evaluación
- **Expansión:** 2 evaluaciones
- **Contracción:** 2 evaluaciones
- **Reducción:** n evaluaciones (¡la más costosa!)

Complejidad total

- Mejor caso: $O(1)$ evaluaciones por iteración
- Peor caso: $O(n)$ evaluaciones por iteración
- Número de iteraciones: Depende del problema

Comparación: Los métodos basados en gradiente requieren $O(n)$ operaciones para calcular ∇f , pero convergen en menos iteraciones.