Differential Evolution

MÓDULO 4

Técnicas computacionales avanzadas para modelar fenómenos sociales

Concentración en Economía Aplicada y Ciencia de Datos

ITESM-SF | CDMX, Feb-Jun 2026

INTRODUCCIÓN

¿Qué es Differential Evolution?

El algoritmo **Differential Evolution** (DE) fue desarrollado por Price (1996) y Price et al. (2006). Es un algoritmo basado en vectores que utiliza cruza y mutación de soluciones para encontrar óptimos globales.

DE puede ser considerado un desarrollo de los algoritmos genéticos donde hay ecuaciones de actualización explícitas. A diferencia de otros métodos, DE usa números reales para las soluciones, de forma que no hay necesidad de codificar y decodificar.

Ventajas Clave

- Usa números reales directamente
- Ecuaciones explícitas de actualización
- Eficiente para optimización global
- Fácil implementación

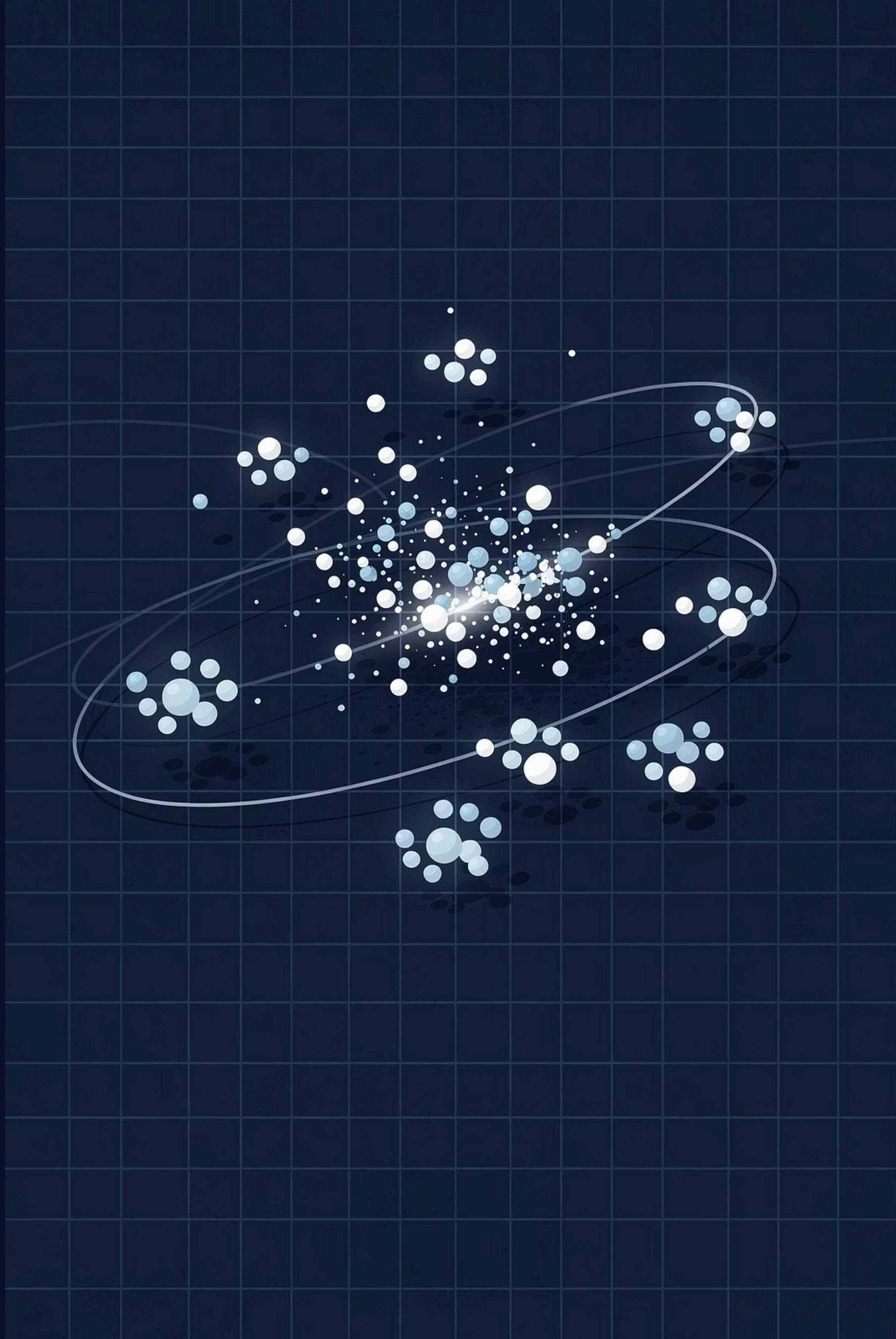
Estructura de la Población

El algoritmo crea de forma aleatoria una población de N vectores d-dimensionales con valores reales (soluciones candidatas) dentro de la región del espacio de búsqueda $[b_L, b_U]$.

La solución x_i en la generación t quedaría representada de la siguiente manera:

$$x_i = (x_{1,i}^t, x_{2,i}^t, \ldots, x_{d,i}^t)$$

que consiste de d componentes en el espacio d dimensional.



PROCESO

Mutación y Recombinación

Selección

Se toma una solución candidata x_i de la población

Mutación

Se crea un **trial vector** mediante mutación y recombinación

Evaluación

Si el trial vector es mejor, reemplaza a x_i

Iteración

El proceso se repite hasta alcanzar el criterio de paro

Para cada solución x_i, el algoritmo lleva a cabo **mutación** y **recombinación** para producir una solución candidata temporal. Si la evaluación de la función a minimizar del trial vector es menor que la de x_i, entonces el trial vector reemplaza a x_i en la población.

Generación de Soluciones Candidatas

Paso 1: Selección de Vectores

El proceso inicia tomando una solución candidata x_i de la población. Después, el algoritmo selecciona tres soluciones candidatas diferentes x_a, x_b y x_c de la población (todas distintas a x_i).

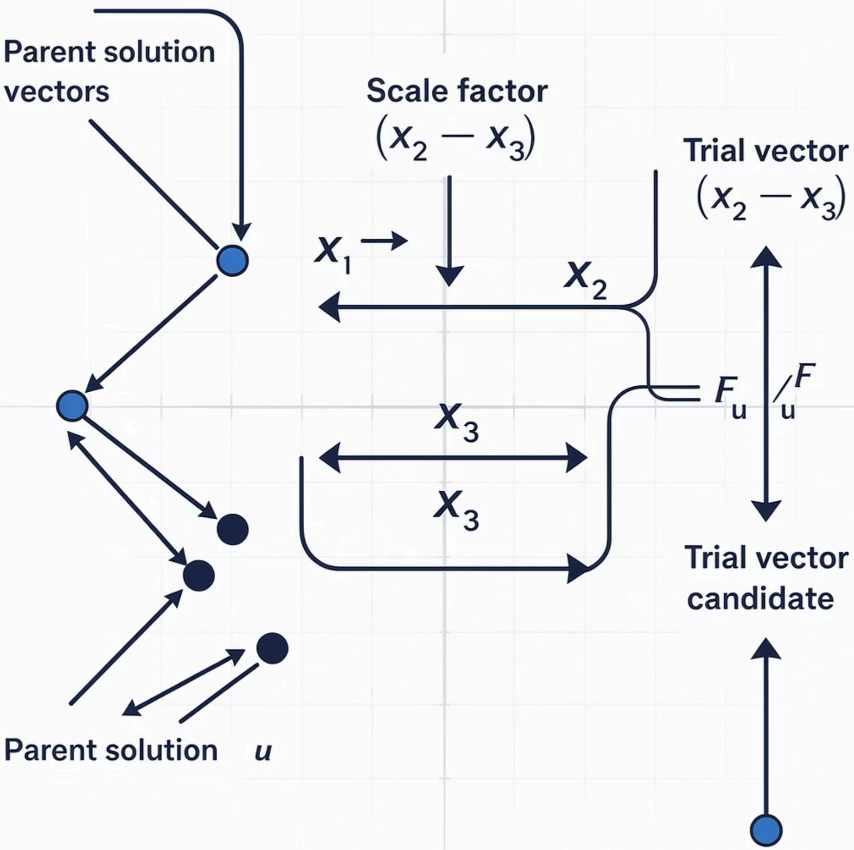
Llamemos respectivamente a dichos vectores como V_1, V_2 y V_b donde V_b representa a un *vector base*.

Paso 2: Diferencia Vectorial

Entonces, el algoritmo calcula la diferencia vectorial como:

$$V_d = V_1 - V_2 \quad (1)$$

Este vector diferencia captura la dirección y magnitud de búsqueda en el espacio de soluciones.



Mutación Diferencial

Se crea un **vector mutante** V_m al sumar V_b el vector V_d multiplicado por un factor de escalamiento λ:

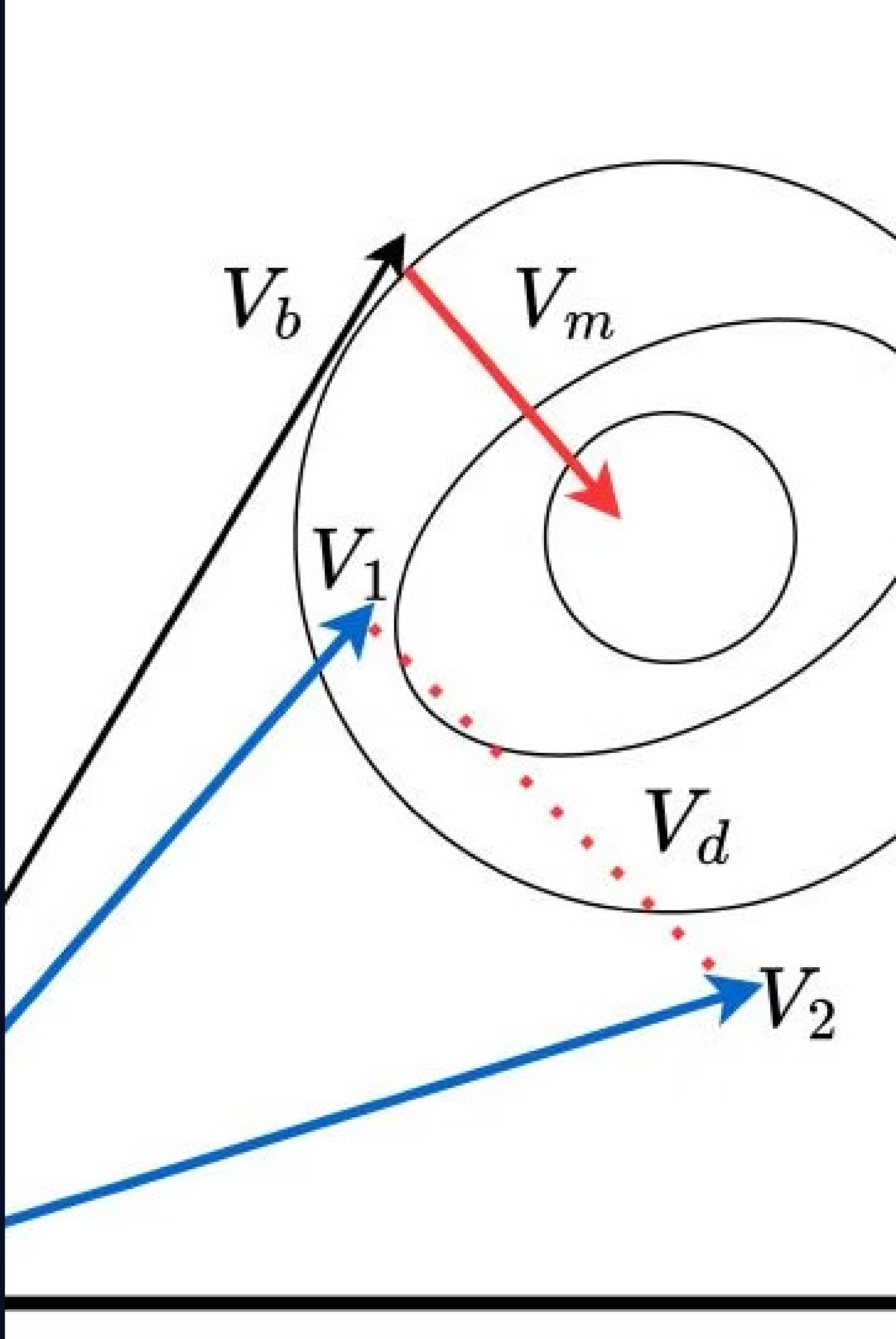
$$V_m = V_b + \lambda V_d \quad (2)$$

Este proceso es llamado **mutación diferencial** y es el corazón del algoritmo DE. El factor λ controla la amplitud de la mutación.

Visualización del Proceso

El diagrama muestra cómo se genera el vector mutante V_m (flecha roja) a partir del vector base V_b (flecha negra) y la diferencia vectorial V_d (línea punteada roja) calculada entre V_1 y V_2 (flechas azules).

Los círculos concéntricos representan las curvas de nivel de la función objetivo, mostrando cómo el algoritmo explora el espacio de búsqueda hacia el óptimo.



RECOMBINACIÓN

Creación del Trial Vector

Después que el vector mutante V_m ha sido creado, se produce un *trial vector* V_t como un crossover entre x_i y el vector mutante V_m de acuerdo a:

$$V_t(j) = \begin{cases} V_m(j) & \text{if } (r_j < p_c) \text{ o } j = J_r \\ x_i(j) & \text{en caso contrario} \end{cases} \qquad (3)$$

para $j \in [i, n]$.

r_j

Número aleatorio distribuido uniformemente en $[0, 1]$

J_r

Entero distribuido uniformemente en $[1, n]$

Rol de J_r

Asegurar que V_t no será un clon de x_i

Parámetros de Control

La eficiencia del algoritmo queda controlada por dos parámetros principales: el factor de escalamiento λ y la probabilidad de crossover p_c.

Factor de Escalamiento λ

De acuerdo a Yang (2020), es el parámetro más sensible. Un valor entre $\lambda \in [0, 2]$ es aceptable en teoría, pero $\lambda \in (0, 1)$ es más eficiente en la práctica.

Rango recomendado: $\lambda \in [0.45, 0.95]$

Primera selección: $\lambda \in [0.7, 0.9]$

Probabilidad de Crossover p_c

Controla la frecuencia con la que se toman componentes del vector mutante versus la solución actual.

Rango recomendado: $p_c \in [0.1, 0.8]$

REFERENCIAS

Bibliografía

Price, K., Storn, R. M., and Lampinen, J. A. (2006). *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media.

Price, K. V. (1996). Differential evolution: a fast and simple numerical optimizer. In *Proceedings of North American fuzzy information processing*, pages 524–527. IEEE.

Scardua, L. A. (2021). *Applied Evolutionary Algorithms for Engineers Using Python*. CRC Press.

Yang, X.-S. (2020). *Nature-inspired optimization algorithms*. Academic Press.