Differential Evolution

Módulo 4 : Técnicas computacionales avanzadas para modelar fenómenos sociales
Concentración en Economía Aplicada y Ciencia de Datos
ITESM

6 de noviembre de 2022

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

Differential Evolution

- El algoritmo crea de forma aleatoria una población de N d-dimensional vectores con valores reales (soluciones candidatas) dentro de la región del espacio de búsqueda $[\mathrm{b}_L, \mathrm{b}_U]$.
- La solución x_i en la generación t quedaría representada de la siguiente manera

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

que consiste de d componentes en el espacio d dimensional.

Differential Evolution

- Para cada solución x_i el algoritmo lleva acabo **mutación** y **recombinación** para producir una solución candidata temporal llamada **trial vector**.
- 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 y viene a ser una nueva solución candidata. El proceso se repite hasta que se alcanza el criterio de paro.

Generación de soluciones candidatas

- El proceso inicia tomando una solución candidata x_i de la población.
- Después de tomar x_i, el algoritmo selecciona tres soluciones candidatas diferentes x_a, x_b y x_c de la población (todas distinas a x_i).
- Llamemos respectivamente a dichos vectores como V_1, V_2 y V_b donde V_b representa a un *vector base*.

Entonces, el algoritmo calcula la diferencia vectorial como:

$$V_d = V_1 - V_2 \tag{1}$$

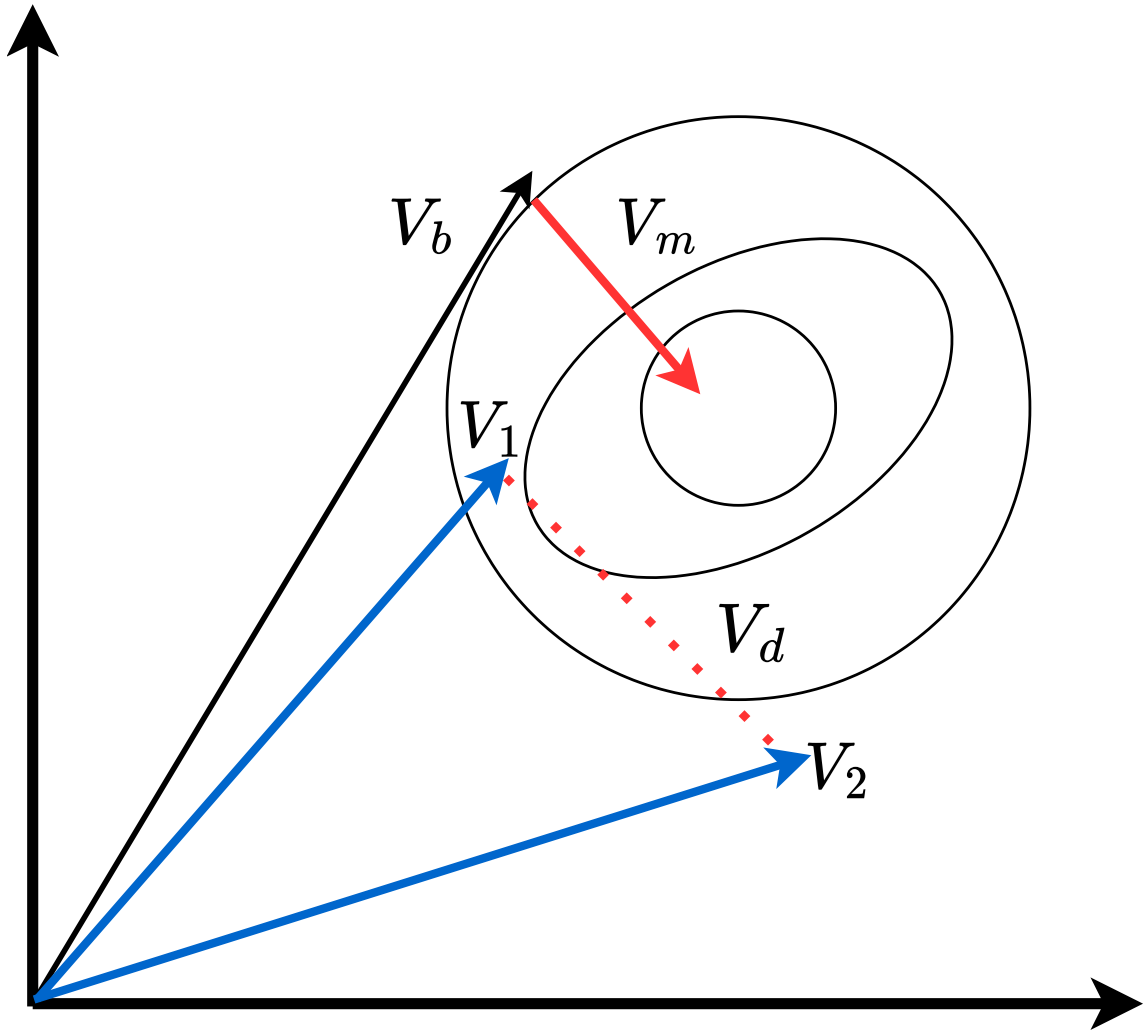
Generación de soluciones candidatas

Entonces se crea un **vector mutante** V_m al sumar V_b el vector V_d multiplicado por un factor de escalamiento λ,

$$V_m = V_b + \lambda V_d \tag{2}$$

Este proceso es llamado **mutación diferencial**.

Generación de soluciones candidatas



Generación de soluciones candidatas

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

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

para $j \in [i, n]$.
r_j es un número aleatorio distribuido uniformemente en el intervalo $[0, 1]$ y J_r es un entero distribuido uniformemente en el intervalo $[1, n]$.
El rol de J_r es la de asegurar que de ninguna manera V_t será un clon de x_i.

Generación de soluciones candidatas

- El **trial vector** se convierte en una solución sí y sólo sí $f_c(V_t) < f_c(x_i)$, donde f_c es la función a minimizar.
- En este caso, V_t remplaza a x_i en la población.
- La eficiencia del algoritmo queda controlada por dos parámetros: el factor de escalamiento, λ, y la probabilidad de crossover, p_c.

Algoritmo Diferential Evolution (Scardua, 2021)

Algorithm 1: Classic Differential Evolution

$\alpha \in (0, 0.9]$ es el parámetro de tamaño de paso
p_c es el parámetro de probabilidad de crossover
$P \leftarrow$ población aleatoria de N individuos $x_i \in \mathbb{R}^n$
while no se alcanza el criterio de paro **do**
  for $i \in [1, N]$ **do**
    $x_i \leftarrow$ pick the i-th candidate solution
    $x_a \leftarrow$ randomly select x_a where $a \neq i$
    $V_1 \leftarrow x_a$
    $x_b \leftarrow$ randomly select x_b where $b \neq a \neq i$
    $V_2 \leftarrow x_b$
    $x_c \leftarrow$ randomly select x_c where $c \neq b \neq a \neq i$
    $V_b \leftarrow x_c$
    $V_d \leftarrow V_1 - V_2$
    $V_m \leftarrow V_b + \alpha V_d$
    $i_r \leftarrow$ random integer $\in [1, n]$
    for each dimension $j \in [1, n]$ **do**
      $r_j \leftarrow$ random number $\in [0, 1]$
      if $r_j < p_c$ or $j = i_r$ **then**
        $V_i(j) = V_m(j)$
      else
        $V_t(j) = x_i(j)$
      end if
    end for
    if $f_c(V_t) < f_c(x_i)$ **then**
      $x_i \leftarrow V_t$
    end if
  end for
end while

Elección de parámetros

- De acuerdo a Yang (2020), el factor de escalamiento, λ, 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.
- Un buen rango es $\lambda \in [0.45, 0.95]$, con una primera selección entre $\lambda \in [0.7, 0.9]$.
- Para la probabilidad de crossover, p_c, un buen rango es $p_c \in [0.1, 0.8]$.

References I

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.