DIFFERENTIAL EVOLUTION (DE)

agungsetiabudi@ub.ac.id

Differential Evolution (DE)

- Differential Evolution adalah algoritma optimisasi stokastik berbasis populasi yang dikembangkan untuk menyelesaikan masalah optimisasi kontinu.
- DE berfokus pada eksplorasi ruang pencarian dengan cara yang berbeda dari algoritma evolusi lainnya, yaitu melalui pemanfaatan perbedaan (differential) antar solusi yang ada dalam populasi.

Keunggulan DE

- Cepat dalam mencapai konvergensi dibandingkan beberapa metode optimasi lainnya.
- Robust terhadap berbagai jenis masalah, baik linier maupun non-linear.
- Menggunakan parameter tuning yang minimal (hanya tiga parameter utama: ukuran populasi, faktor skala F, dan probabilitas crossover CR).

Langkah-Langkah DE

Differential Evolution bekerja dengan empat langkah utama:

1. Inisialisasi Populasi
2. Mutasi
3. Rekombinasi
4. Seleksi

Contoh Permasalahan

Misalkan kita ingin meminimalkan fungsi berikut:
$f(x, y) = x^2 + y^2$
Ruang pencarian yang digunakan adalah $[-5, 5]$ untuk masing-masing variabel x dan y.

1. Inisialisasi Populasi

- Misalkan kita memilih ukuran populasi (N = 4) dan memulai dengan 4 vektor acak:
  - $X_1 = [1.2, -3.4]$,
  - $X_2 = [-2.1, 1.0]$,
  - $X_3 = [0.5, -1.5]$,
  - $X_4 = [-4.0, 2.3]$
- Setiap vektor dalam populasi ini adalah solusi kandidat yang akan dievaluasi pada setiap iterasi.

Contoh program

```
def initialize_population(population_size, bounds):
    population = []
    for _ in range(population_size):
        individual = [np.random.uniform(low, high) for (low, high) in bounds]
        population.append(individual)
    return np.array(population)
```

2. Mutasi

- Di tahap mutasi, DE membuat vektor mutasi V_i untuk setiap vektor target X_i dalam populasi. Vektor mutasi diperoleh dengan rumus berikut:
  - $V_i = X_{r1} + F \cdot (X_{r2} - X_{r3})$
- di mana:
  - X_{r1}, X_{r2}, dan X_{r3} adalah tiga vektor acak yang berbeda dari X_i,
  - F adalah faktor skala, biasanya dalam rentang [0 - 1].

2. Mutasi

- Misalkan $F = 0.8$. Untuk menghitung V_1, misalkan kita memilih:
  - $X_{r1} = X_2 = [-2.1, 1.0]$
  - $X_{r2} = X_3 = [0.5, -1.5]$
  - $X_{r3} = X_4 = [-4.0, 2.3]$

2. Mutasi

- Maka, vektor mutasi V_1 dihitung sebagai berikut:
  - $V_1 = X_2 + F \cdot (X_3 - X_4)$
  - $= [-2.1, 1.0] + 0.8 \cdot ([0.5, -1.5] - [-4.0, 2.3])$
  - $= [-2.1, 1.0] + 0.8 \cdot [4.5, -3.8]$
  - $= [-2.1, 1.0] + [3.6, -3.04]$
  - $= [1.5, -2.04]$
- Jadi, vektor mutasi V_1 untuk X_1 adalah $[1.5, -2.04]$.

Contoh program

```
r1, r2, r3 = np.random.choice(indices, 3, replace=False)
mutant_vector = population[r1] + F * (population[r2] - population[r3])
```

3. Rekombinasi (Crossover)

- Langkah berikutnya adalah menghasilkan vektor uji U_i melalui rekombinasi antara vektor target X_i dan vektor mutasi V_i.
- Crossover dilakukan elemen demi elemen berdasarkan probabilitas crossover CR.
- Misalkan $CR = 0.9$. Untuk menghasilkan U_1, kita melakukan hal berikut:
  - Jika nilai acak $r < CR$, elemen dari V_i digunakan.
  - Jika $r \geq CR$, elemen dari X_i digunakan.

3. Rekombinasi (Crossover)

- Misalkan untuk $X_1 = [1.2, -3.4]$ dan $V_1 = [1.5, -2.04]$, kita mendapatkan nilai acak $r = 0.8$ untuk elemen pertama dan $r = 0.5$ untuk elemen kedua. Karena $r < CR$ untuk kedua elemen, kita mengambil elemen-elemen dari V_1 sepenuhnya:
  - $U_1 = [1.5, -2.04]$

Contoh program

```
trial_vector = []
for j in range(len(bounds)):
    if np.random.rand() < CR:
        trial_vector.append(mutant_vector[j])
    else:
        trial_vector.append(population[i][j])
trial_vector = np.array(trial_vector)
```

4. Seleksi

- Dalam tahap seleksi, kita memilih antara vektor target X_i dan vektor uji U_i untuk dimasukkan ke generasi berikutnya.
- Seleksi dilakukan berdasarkan nilai fungsi objektif, yaitu solusi dengan nilai fungsi objektif yang lebih baik akan dipertahankan.

4. Seleksi

- Misalkan kita ingin meminimalkan $f(x, y) = x^2 + y^2$:
- $f(X_1) = f(1.2, -3.4) = 1.2^2 + (-3.4)^2 = 1.44 + 11.56 = 13.0$
- $f(U_1) = f(1.5, -2.04) = 1.5^2 + (-2.04)^2 = 2.25 + 4.1616 = 6.4116$
- Karena $f(U_1) < f(X_1)$, kita memilih $U_1 = [1.5, -2.04]$ sebagai bagian dari populasi baru.

Contoh program

```
if objective_function(trial_vector) < objective_function(population[i]):
    new_population.append(trial_vector)
else:
    new_population.append(population[i])
```

Iterasi

- Langkah-langkah ini diulangi untuk semua vektor dalam populasi, dan proses diulangi selama beberapa generasi hingga tercapai kondisi penghentian, seperti jumlah maksimum generasi atau konvergensi solusi.

Contoh Iterasi Singkat

- Misalkan kita melakukan satu iterasi untuk semua vektor di populasi:
  1. **Mutasi**: Buat vektor mutasi untuk setiap vektor target X_i.
  2. **Rekombinasi**: Buat vektor uji U_i dari kombinasi X_i dan V_i.
  3. **Seleksi**: Bandingkan nilai fungsi objektif dari X_i dan U_i, pilih yang lebih baik.
- Dengan terus mengulangi langkah ini, populasi akan bergerak menuju area optimum di ruang pencarian.