초기 개념 및 전체 흐름

GA(Genetic Algorithm)은 다음의 과정을 반복해 최적해를 탐색

- 초기 개체군 생성 : 무작위로 생성된 해(유전자. 현재는 x 값)를 사용
- 선택 : 좋은 개체들을 선택
- 교차 : 두 부모의 유전자를 섞어 자식을 만듦
- 돌연변이 : 유전자에 무작위 변화를 주어 탐색 범위를 확장

함수별 세부 설명 및 수식

```
def func(x):
    return np.sin(x) - .2 * abs(x)
```

- 개체의 유전자인 (x)에 대해 적합도 값(함수값 y)을 계산

$$f(x) = sin(x) - 0.2|x|$$

- x의 값에 따라 변동하는 sin 함수의 특성과, x의 절대값에 비례하는 페널티를 결합하여 최적해를 찾는 기준을 제공

```python
def _utils_constraints(g, min, max):
    if max and g > max:
        g = max
    if min and g < min:
        g = min
    return g
```

- 유전자 값 `g`가 주어진 최소값 `min`과 최대값 `max` 범위를 벗어나지 않도록 보정

$$g_{\text{new}} = \begin{cases} \max & \text{if } g > \max \\ \min & \text{if } g < \min \\ g & \text{otherwise} \end{cases}$$

- **예시**: 만약 $g = 12$ 이고, $\max = 10$이면 $g_{\text{new}} = 10$ 선택

```
def select_tournament(population, tournament_size):
    new_offspring = []
    for _ in range(len(population)):
        candidates = [random.choice(population) for _ in
        range(tournament_size)]
        new_offspring.append(max(candidates, key = lambda ind:
        ind.fitness))
    return new_offspring
```

- 전체 인구 집단에서 토너먼트 방식을 통해 적합도가 가장 높은 개체를 선택
  여러 개체 $\{I_1, I_2, \ldots, I_k\}$중에서,

$$I_{\text{selected}} = \arg \max_{I \in \{I_1, \ldots, I_k\}} \text{fitness}(I)$$

  - 토너먼트 크기 k (여기서는 3)를 설정하고, 무작위로 선택된 k개의 후보 중 적합도(예: $f(x)$)가 가장 높은 개체를 선택
- **ex:** 3개의 후보 중 가장 $\sin(x) - 0.2|x|$(함수값) 값이 높은 개체가 선택됨

```
def crossover_blend(g1, g2, alpha, min = None, max = None):
    shift = (1. + 2. * alpha) * random.random() - alpha
    new_g1 = (1. - shift) * g1 + shift * g2
    new_g2 = shift * g1 + (1. - shift) * g2
    return _utils_constraints(new_g1, min, max), _utils_constraints(new_g2, min, max)
```

- 두 부모 유전자 g_1와 g_2를 섞어 두 자식 유전자 값을 생성

$$\text{shift} = (1 + 2\alpha) \cdot r - \alpha$$

  여기서 r은 $[0, 1]$ 사이의 무작위 값, 현재 $\alpha = 1$

- 두 자식의 유전자 값은 다음과 같이 계산됨

$$g_{\text{child1}} = (1 - \text{shift}) \cdot g_1 + \text{shift} \cdot g_2$$
$$g_{\text{child2}} = \text{shift} \cdot g_1 + (1 - \text{shift}) \cdot g_2$$

- 위에서 계산된 두 자식 유전자에 대해 앞서 설정한 범위$[-10, 10]$범위 내에 있도록 설정

```
def mutate_gaussian(g, mu, sigma, min = None, max = None):
    mutated_gene = g + random.gauss(mu, sigma)
    return _utils_constraints(mutated_gene, min, max)
```

- 기존 유전자 g에 가우시안 분포를 따르는 무작위 노이즈를 더해 돌연변이를 발생시킴

$$g_{\text{mutated}} = g + N(\mu, \sigma^2)$$

- 여기서 $N(\mu, \sigma^2)$는 평균 μ와 분산 σ^2를 갖는 정규분포에서 뽑은 난수
- 계산 후, g_{mutated}가 $[-10, 10]$범위를 벗어나지 않도록 설정정
- **ex:** 만약 $g = 5$, $\mu = 0$, $\sigma = 1$이라면, 무작위로 선택된 값 (ex: -0.8)을 더해 $g_{\text{mutated}} = 4.2$

왜 가우시안 분포를 활용할까?

- 평균=0, 분산=1이기에 확률이 0 근처에 몰려 있어, **작은 변화를 유도**함
- 대부분의 변화는 작지만, 때때로 큰 변화도 발생할 수 있어 **전체 탐색 공간을 효과적으로 탐색**할 수 있음

3. 전체 유전 알고리즘의 실행 과정 요약

1. **초기 개체군 생성:**
    - x 값이 무작위로 선택되어 개체군이 생성됨됨
    - 각 개체의 적합도는 $f(x) = \sin(x) - 0.2|x|$로 계산
2. **세대 반복 (총 10세대):**
    - **선택:** 토너먼트 방식으로 $k = 3$개의 후보 중 적합도가 가장 높은 개체를 선택
    - **교차:** 선택된 개체들을 짝지어, 위의 교차 수식을 통해 두 자식 유전자를 생성
    - **돌연변이:** 각 자식 유전자에 대해 가우시안 분포를 이용해 변화를 주며, 범위를 벗어나지 않도록 보정