03-10-24
Genetic Algorithm
Algorithm:-

```
import numpy as np

# Objective function: A simple quadratic function (sum of squares)
def objective_function(x):
    return np.sum(x**2)

# Initialize population: Create a population of individuals with random genes
def initialize_population(pop_size, dim, lower_bound, upper_bound):
    return np.random.uniform(lower_bound, upper_bound, (pop_size, dim))

# Selection: Tournament Selection
def selection(population, fitness, tournament_size=3):
    selected = []
    for _ in range(len(population)):
        participants = np.random.choice(len(population), tournament_size, replace=False)
        best = participants[np.argmin(fitness[participants])]  # Select the best individual
        selected.append(population[best])
    return np.array(selected)

# Crossover: Single-point crossover
def crossover(parents, crossover_rate=0.8):
    offspring = []
    for i in range(0, len(parents), 2):
        if np.random.rand() < crossover_rate and i + 1 < len(parents):
            crossover_point = np.random.randint(1, parents.shape[1])
            child1 = np.concatenate((parents[i, :crossover_point], parents[i + 1, crossover_point:]))
            child2 = np.concatenate((parents[i + 1, :crossover_point], parents[i, crossover_point:]))
            offspring.append(child1)
            offspring.append(child2)
        else:
            offspring.append(parents[i])
            if i + 1 < len(parents):
                offspring.append(parents[i + 1])
    return np.array(offspring)

# Mutation: Random mutation
def mutation(offspring, mutation_rate=0.1, lower_bound=-10, upper_bound=10):
    for i in range(len(offspring)):
        if np.random.rand() < mutation_rate:
            mutation_point = np.random.randint(offspring.shape[1])
            offspring[i, mutation_point] = np.random.uniform(lower_bound, upper_bound)
```

```
    return offspring

# Genetic Algorithm Function
def genetic_algorithm(pop_size, generations, lower_bound, upper_bound, mutation_rate,
crossover_rate):
    dim = 3  # Number of design parameters (x0, x1, x2)

    # Initialize population
    population = initialize_population(pop_size, dim, lower_bound, upper_bound)

    # GA Main Loop
    for generation in range(generations):
        # Evaluate the fitness of the population
        fitness = np.array([objective_function(ind) for ind in population])

        # Select the best individuals
        selected_parents = selection(population, fitness)

        # Perform crossover to generate offspring
        offspring = crossover(selected_parents, crossover_rate)

        # Perform mutation on the offspring
        offspring = mutation(offspring, mutation_rate, lower_bound, upper_bound)

        # Create new population by replacing the old one with offspring
        population = offspring

    # Evaluate final population fitness
    final_fitness = np.array([objective_function(ind) for ind in population])
    best_index = np.argmin(final_fitness)

    return population[best_index], final_fitness[best_index]

# Gather user input for the algorithm
print("Welcome to Genetic Algorithm Optimization!")
pop_size = int(input("Enter the population size: "))
generations = int(input("Enter the number of generations: "))
lower_bound = float(input("Enter the lower bound for the design parameters: "))
upper_bound = float(input("Enter the upper bound for the design parameters: "))
mutation_rate = float(input("Enter the mutation rate (between 0 and 1): "))
crossover_rate = float(input("Enter the crossover rate (between 0 and 1): "))

# Run the Genetic Algorithm
```

```
best_solution, best_value = genetic_algorithm(pop_size, generations, lower_bound,
upper_bound, mutation_rate, crossover_rate)

# Display the result
print("\nOptimization Results:")
print("Best Solution (Design Parameters):", best_solution)
print("Best Objective Value:", best_value)
```

Output:-

```
Welcome to Genetic Algorithm Optimization!
Enter the population size: 20
Enter the number of generations: 100
Enter the lower bound for the design parameters: -10
Enter the upper bound for the design parameters: 10
Enter the mutation rate (between 0 and 1): 0.1
Enter the crossover rate (between 0 and 1): 0.8

Optimization Results:
Best Solution (Design Parameters): [-0.33895557  0.41085676 -0.02515901]
Best Objective Value: 0.28432713514912306
```