24-10-24

Particle Swarm
Optimization Algorithm:-

```
import numpy as np



# Objective function: A simple quadratic function (sum of
squares) def objective_function(x):

    return np.sum(x**2)



# Particle Swarm Optimization (PSO) Algorithm

    def particle_swarm_optimization(n, iterations, lower_bound, upper_bound, w,
        c1, c2): # Initialize particles' positions and velocities

    dim = 3 # Number of design parameters (x0, x1, x2)

    positions = np.random.uniform(lower_bound, upper_bound, (n, dim))
    velocities = np.random.uniform(-1, 1, (n, dim))


    # Initialize personal bests (pbest) and global best (gbest)
    pbest = positions.copy()
    pbest_values = np.array([objective_function(p) for p in
    positions]) gbest = pbest[np.argmin(pbest_values)]

    gbest_value = np.min(pbest_values)



    # Main PSO loop

        for _ in
            range(iterations):
            for i in range(n):
            # Update velocity: v = w * v + c1 * r1 * (pbest - position) + c2 * r2 * (gbest -
            position) r1 = np.random.rand(dim)

            r2 = np.random.rand(dim)
```

```
        velocities[i] = w * velocities[i] + c1 * r1 * (pbest[i] - positions[i]) + c2 * r2 * (gbest -
positions[i])


        # Update position: x = x
        + v positions[i] +=
        velocities[i]


        # Apply bounds

        positions[i] = np.clip(positions[i], lower_bound, upper_bound)


        # Evaluate the objective function for the new
        position fitness = objective_function(positions[i])


        # Update personal best if the current position is
        better if fitness < pbest_values[i]:

            pbest[i] = positions[i]


            pbest_values[i] = fitness


    # Update global best
    min_fitness_index = np.argmin(pbest_values)
        if pbest_values[min_fitness_index] <
            gbest_value: gbest =
            pbest[min_fitness_index]
        gbest_value =

  pbest_values[min_fitness_index] return gbest,

  gbest_value

# Gather user input for the algorithm
print("Welcome to Particle Swarm
Optimization!")
n = int(input("Enter the number of particles (population size): "))
iterations = int(input("Enter the number of iterations: "))
```

```
lower_bound = float(input("Enter the lower bound for the design parameters:
")) upper_bound = float(input("Enter the upper bound for the design
parameters: ")) w = float(input("Enter the inertia weight (w): "))
c1 = float(input("Enter the cognitive coefficient (c1):
")) c2 = float(input("Enter the social coefficient (c2):
"))


# Run the PSO algorithm

best_solution, best_value = particle_swarm_optimization(n, iterations,
lower_bound, upper_bound, w, c1, c2)


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

OUTPUT:

```
Welcome to Particle Swarm Optimization!
Enter the number of particles (population size): 20
Enter the number of iterations: 100
Enter the lower bound for the design parameters: -10
Enter the upper bound for the design parameters: 10
Enter the inertia weight (w): 0.7
Enter the cognitive coefficient (c1): 1.5
Enter the social coefficient (c2): 1.5

Optimization Results:
Best Solution (Design Parameters): [6.26915659e-06 3.67385208e-07 1.17464080e-05]
Best Objective Value: 1.774153964226925e-10
```