particleswarmoptm

November 20, 2024

[ ]:

```
#particle swarm optimization algorithm to minimize objective fn
print("Name:Sudarshan Komar","USN:1BM22CS291",sep="\n")

import numpy as np

# Objective function (e.g., Sphere function)
def objective_function(x):
    return sum(xi**2 for xi in x)

class Particle:
    def __init__(self, dim):
        self.position = np.random.rand(dim) * 10 - 5  # Random position in␣
 ↪range [-5, 5]
        self.velocity = np.random.rand(dim) * 2 - 1   # Random velocity
        self.best_position = self.position.copy()
        self.best_value = objective_function(self.position)

def pso(num_particles, dimensions, max_iterations):
    w = 0.5  # Inertia weight
    c1 = 1.5  # Cognitive coefficient
    c2 = 1.5  # Social coefficient

    # Initialize particles
    particles = [Particle(dimensions) for _ in range(num_particles)]
    global_best_position = particles[0].best_position.copy()
    global_best_value = particles[0].best_value

    for t in range(max_iterations):
        for particle in particles:
            # Update velocity
            r1, r2 = np.random.rand(dimensions), np.random.rand(dimensions)
            particle.velocity = (w * particle.velocity +
                                 c1 * r1 * (particle.best_position - particle.
 ↪position) +
                                 c2 * r2 * (global_best_position - particle.
 ↪position))
```

```
            # Update position
            particle.position += particle.velocity

            # Evaluate fitness
            value = objective_function(particle.position)

            # Update personal best
            if value < particle.best_value:
                particle.best_value = value
                particle.best_position = particle.position.copy()

            # Update global best
            if value < global_best_value:
                global_best_value = value
                global_best_position = particle.position.copy()

    # Print only the final best result
    print(f"Best Position: {global_best_position}, Best Value: {global_best_value}")
    return global_best_position, global_best_value

# Parameters
num_particles = 30
dimensions = 2
max_iterations = 1000

best_position, best_value = pso(num_particles, dimensions, max_iterations)
```

```
Name:Sudarshan Komar
USN:1BM22CS291
Best Position: [5.92457810e-110 3.05564784e-109], Best Value:
9.687989953612073e-218
```