Sanjana Shetty
5E

Particle Swarm Optimization

```
# using rastrigin & sphere - standard functions for optimization

import random
import math  # cos() for Rastrigin
import copy  # array-copying convenience
import sys   # max float


# -------fitness functions---------

# Rastrigin function
def fitness_rastrigin(position):
    fitnessVal = 0.0
    for i in range(len(position)):
        xi = position[i]
        fitnessVal += (xi * xi) - (10 * math.cos(2 * math.pi * xi)) + 10
    return fitnessVal


# Sphere function
def fitness_sphere(position):
    fitnessVal = 0.0
    for i in range(len(position)):
        xi = position[i]
        fitnessVal += (xi * xi)
    return fitnessVal


# --------------------------


# Particle class
```

```
class Particle:
    def __init__(self, fitness, dim, minx, maxx, seed):
        self.rnd = random.Random(seed)

        # Initialize position of the particle with 0.0 value
        self.position = [0.0 for i in range(dim)]

        # Initialize velocity of the particle with 0.0 value
        self.velocity = [0.0 for i in range(dim)]

        # Initialize best particle position of the particle with 0.0 value
        self.best_part_pos = [0.0 for i in range(dim)]

        # Loop dim times to calculate random position and velocity
        # Range of position and velocity is [minx, maxx]
        for i in range(dim):
            self.position[i] = ((maxx - minx) * self.rnd.random() + minx)
            self.velocity[i] = ((maxx - minx) * self.rnd.random() + minx)

        # Compute fitness of particle
        self.fitness = fitness(self.position)  # current fitness

        # Initialize best position and fitness of this particle
        self.best_part_pos = copy.copy(self.position)
        self.best_part_fitnessVal = self.fitness  # best fitness


# Particle Swarm Optimization function
def pso(fitness, max_iter, n, dim, minx, maxx):
    # Hyper parameters
    w = 0.729  # inertia
    c1 = 1.49445  # cognitive (particle)
    c2 = 1.49445  # social (swarm)

    rnd = random.Random(0)

    # Create n random particles
    swarm = [Particle(fitness, dim, minx, maxx, i) for i in range(n)]

    # Compute the value of best_position and best_fitness in swarm
```

```
    best_swarm_pos = [0.0 for _ in range(dim)]
    best_swarm_fitnessVal = sys.float_info.max  # swarm best

    # Compute best particle of swarm and its fitness
    for i in range(n):  # check each particle
        if swarm[i].fitness < best_swarm_fitnessVal:
            best_swarm_fitnessVal = swarm[i].fitness
            best_swarm_pos = copy.copy(swarm[i].position)

    # Main loop of PSO
    Iter = 0
    while Iter < max_iter:
        # After every 10 iterations
        # print iteration number and best fitness value so far
        if Iter % 10 == 0 and Iter > 1:
            print("Iter = " + str(Iter) + " best fitness = %.3f" %
best_swarm_fitnessVal)

        for i in range(n):  # process each particle
            # Compute new velocity of current particle
            for k in range(dim):
                r1 = rnd.random()  # randomizations
                r2 = rnd.random()

                swarm[i].velocity[k] = (
                    (w * swarm[i].velocity[k]) +
                    (c1 * r1 * (swarm[i].best_part_pos[k] -
swarm[i].position[k])) +
                    (c2 * r2 * (best_swarm_pos[k] - swarm[i].position[k]))
                )

                # If velocity[k] is not in [minx, max]
                # then clip it
                if swarm[i].velocity[k] < minx:
                    swarm[i].velocity[k] = minx
                elif swarm[i].velocity[k] > maxx:
                    swarm[i].velocity[k] = maxx

            # Compute new position using new velocity
            for k in range(dim):
```

```
                swarm[i].position[k] += swarm[i].velocity[k]

            # Compute fitness of new position
            swarm[i].fitness = fitness(swarm[i].position)

            # Is new position a new best for the particle?
            if swarm[i].fitness < swarm[i].best_part_fitnessVal:
                swarm[i].best_part_fitnessVal = swarm[i].fitness
                swarm[i].best_part_pos = copy.copy(swarm[i].position)

            # Is new position a new best overall?
            if swarm[i].fitness < best_swarm_fitnessVal:
                best_swarm_fitnessVal = swarm[i].fitness
                best_swarm_pos = copy.copy(swarm[i].position)

        Iter += 1
    return best_swarm_pos


# ----------------------------
# Driver code for Rastrigin function

print("\nBegin particle swarm optimization on Rastrigin function\n")
dim = 3
fitness = fitness_rastrigin

print("Goal is to minimize Rastrigin's function in " + str(dim) + "
variables")
print("Function has known min = 0.0 at (", end="")
for i in range(dim - 1):
    print("0, ", end="")
print("0)")

num_particles = 50
max_iter = 100

print("Setting num_particles = " + str(num_particles))
print("Setting max_iter = " + str(max_iter))
print("\nStarting PSO algorithm\n")
```

```
best_position = pso(fitness, max_iter, num_particles, dim, -10.0, 10.0)

print("\nPSO completed\n")
print("\nBest solution found:")
print(["%.6f" % best_position[k] for k in range(dim)])
fitnessVal = fitness(best_position)
print("Fitness of best solution = %.6f" % fitnessVal)

print("\nEnd particle swarm for Rastrigin function\n")


print()
print()


# Driver code for Sphere function
print("\nBegin particle swarm optimization on Sphere function\n")
dim = 3
fitness = fitness_sphere

print("Goal is to minimize Sphere function in " + str(dim) + " variables")
print("Function has known min = 0.0 at (", end="")
for i in range(dim - 1):
    print("0, ", end="")
print("0)")

num_particles = 50
max_iter = 100

print("Setting num_particles = " + str(num_particles))
print("Setting max_iter = " + str(max_iter))
print("\nStarting PSO algorithm\n")

best_position = pso(fitness, max_iter, num_particles, dim, -10.0, 10.0)

print("\nPSO completed\n")
print("\nBest solution found:")
print(["%.6f" % best_position[k] for k in range(dim)])
fitnessVal = fitness(best_position)
print("Fitness of best solution = %.6f" % fitnessVal)
```

```
print("\nEnd particle swarm for Sphere function\n")
```

OUTPUT:

```
Goal is to minimize Rastrigin's function in 3 variables
Function has known min = 0.0 at (0, 0, 0)
Setting num_particles = 50
Setting max_iter = 100

Starting PSO algorithm

Iter = 10 best fitness = 8.463
Iter = 20 best fitness = 4.792
Iter = 30 best fitness = 2.223
Iter = 40 best fitness = 0.251
Iter = 50 best fitness = 0.251
Iter = 60 best fitness = 0.061
Iter = 70 best fitness = 0.007
Iter = 80 best fitness = 0.005
Iter = 90 best fitness = 0.000

PSO completed


Best solution found:
['0.000618', '0.000013', '0.000616']
Fitness of best solution = 0.000151

End particle swarm for Rastrigin function
```

```
Begin particle swarm optimization on Sphere function

Goal is to minimize Sphere function in 3 variables
Function has known min = 0.0 at (0, 0, 0)
Setting num_particles = 50
Setting max_iter = 100

Starting PSO algorithm

Iter = 10 best fitness = 0.189
Iter = 20 best fitness = 0.012
Iter = 30 best fitness = 0.001
Iter = 40 best fitness = 0.000
Iter = 50 best fitness = 0.000
Iter = 60 best fitness = 0.000
Iter = 70 best fitness = 0.000
Iter = 80 best fitness = 0.000
Iter = 90 best fitness = 0.000

PSO completed


Best solution found:
['0.000004', '-0.000001', '0.000007']
Fitness of best solution = 0.000000

End particle swarm for Sphere function
```