粒子群优化算法 (PSO)

1 介绍

粒子群优化算法（Particle Swarm Optimization，PSO）是一种基于群体智能的优化算法，由Kennedy和Eberhart于1995年提出。该算法受鸟群觅食行为的启发，通过群体中个体间的信息共享来寻找全局最优解。

2 算法原理

2.1 粒子表示

粒子群优化算法中的每个个体称为粒子，每个粒子表示一个潜在的解决方案。粒子在搜索空间中移动，以寻找最优解。

2.2 初始化

初始时，在搜索空间内随机生成一群粒子，每个粒子具有随机的位置和速度。粒子的位置表示一个候选解，速度表示粒子在下一次迭代时的位置变化量。

2.3 适应度评估

每个粒子根据适应度函数评估其当前位置的优劣。适应度函数由具体问题决定，用于衡量解的好坏。

2.4 个体极值和全局极值

每个粒子记录自己迄今为止找到的最佳位置，称为个体极值（pBest）。整个粒子群中找到的最佳位置称为全局极值（gBest）。

2.5 速度和位置更新

在每次迭代中，粒子根据以下公式更新自己的速度和位置：

$$v_i(t+1) = w \cdot v_i(t) + c1 \cdot r1 \cdot (pBest_i - x_i(t)) + c2 \cdot r2 \cdot (gBest - x_i(t)) \tag{1}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2}$$

其中：

- $v_i(t)$ 为粒子 i 在迭代 t 时的速度。

- $x_i(t)$ 为粒子 i 在迭代 t 时的位置。
- w 为惯性权重，控制粒子之前速度的影响。
- $c1$ 和 $c2$ 分别为认知系数和社会系数，控制粒子向个体极值和全局极值学习的程度。
- $r1$ 和 $r2$ 为[0,1]之间的随机数，增加算法的随机性。

2.6 终止条件

当达到预定的最大迭代次数或全局极值的变化在一段时间内小于某个阈值时，算法终止。

3 示例

以下是使用粒子群优化算法求解一个二维函数的最小值的示例。选择Rosenbrock函数作为目标函数：

$$f(x,y) = (a-x)^2 + b(y-x^2)^2 \tag{3}$$

其中，通常设定 $a=1$ 和 $b=100$，函数的全局最小值在点 $(1,1)$ 处，其函数值为0。

3.1 步骤一：初始化

- 惯性权重 $w=0.5$
- 认知系数 $c1=1.5$
- 社会系数 $c2=1.5$
- 粒子数 $N=30$
- 最大迭代次数 $T=100$
- 粒子的位置和速度初始化在[-10, 10]范围内随机取值，速度初始化为零向量

3.2 步骤二：适应度评估

计算每个粒子的位置 (x_i, y_i) 处的适应度值 $f(x_i, y_i)$。

3.3 步骤三：更新个体极值和全局极值

- 对每个粒子，比较其当前适应度值和历史最佳适应度值，更新个体极值 $pBest_i$。
- 比较所有粒子的适应度值，找出当前群体的全局极值 $gBest$。

3.4 步骤四：更新速度和位置

使用以下公式更新粒子的速度和位置：

$$v_i(t+1) = w \cdot v_i(t) + c1 \cdot r1 \cdot (pBest_i - x_i(t)) + c2 \cdot r2 \cdot (gBest - x_i(t)) \tag{4}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{5}$$

其中 $r1$ 和 $r2$ 为[0,1]之间的随机数。

3.5 步骤五：迭代更新

重复步骤二到步骤四，直到达到最大迭代次数 T 或全局极值的变化小于某个阈值。

3.6 结果

假设经过100次迭代后，找到的全局极值 $gBest$ 位置接近 $(1, 1)$，其函数值接近0，即为优化问题的解。

3.7 代码示例

以下是一个简单的Python实现，使用PSO求解Rosenbrock函数的最小值：

```
import numpy as np

# Rosenbrock函数
def rosenbrock(x, y):
    return (1 - x)**2 + 100 * (y - x**2)**2

# 参数初始化
w = 0.5
c1 = 1.5
c2 = 1.5
N = 30
T = 100
x_bound = [-10, 10]
y_bound = [-10, 10]

# 粒子位置和速度初始化
particles = np.random.rand(N, 2) * (x_bound[1] - x_bound[0]) + x_bound[0]
velocities = np.zeros((N, 2))

# 记录个体极值和全局极值
pBest = particles.copy()
gBest = particles[np.argmin([rosenbrock(p[0], p[1]) for p in particles])]
```

```
# 迭代更新
for t in range(T):
    for i in range(N):
        fitness = rosenbrock(particles[i, 0], particles[i, 1])
        if fitness < rosenbrock(pBest[i, 0], pBest[i, 1]):
            pBest[i] = particles[i]
        if fitness < rosenbrock(gBest[0], gBest[1]):
            gBest = particles[i]

    for i in range(N):
        r1, r2 = np.random.rand(), np.random.rand()
        velocities[i] = (w * velocities[i]
                         + c1 * r1 * (pBest[i] - particles[i])
                         + c2 * r2 * (gBest - particles[i]))
        particles[i] += velocities[i]

# 输出最优解
print("最佳位置:", gBest)
print("最小值:", rosenbrock(gBest[0], gBest[1]))
```

通过上述步骤和代码示例，可以更清楚地理解粒子群优化算法在具体问题中的应用。