2024年秋季《人工智能导论》

实验二

群智能算法编程

实验报告

学院：<u>自动化学院</u>

专业：<u>智能科学与技术</u>

学号：__________________

姓名：__________________

2024年11月

题目：

已知：$\begin{array}{l} f(x_1,x_2)=15+x_1^2+x_2^{\,2}-10(\cos 2\pi x_1+\cos 2\pi x_2) \\ -5\le x_i\le 5 \qquad i=1,2 \end{array}$，利用粒子群算法和 Matlab 编程。

（1）绘制函数三维曲面图、二维截面图（等高线云图）。

（2）求出函数的最小值（单号同学）/最大值（双号同学）。

一、实验目的

通过 MATLAB 对给定的二维非线性函数进行可视化和优化分析。具体目标包括：绘制函数的三维曲面图和二维等高线图，以直观展示函数的分布特性；利用粒子群优化算法（PSO）求解函数的全局最小值和最大值。通过该实验，加深对智能优化算法及其应用的理解。

二、实验原理

粒子群优化算法（Particle Swarm Optimization，PSO）是一种基于群体智能的全局优化算法，受到鸟群或鱼群运动行为的启发，用于求解复杂的优化问题。

（一）基本原理

PSO 通过一群“粒子”在搜索空间中移动来寻找最优解：

（1）粒子：每个粒子表示一个潜在解，其位置由向量 $x=(x_1,x_2)$ 表示。

（2）速度更新：粒子根据其历史最优位置（个体最优 p_{best}）和全体粒子的全局最优位置（全局最优 g_{best}）更新速度和位置。

（3）适应度函数：评估粒子当前位置的优劣，目标是找到使适应度函数（目标函数）最小或最大的解。

（二）关键公式

（1）速度更新公式：

$$v_i(t+1)=w\cdot v_i(t)+c_1\cdot r_1\cdot\left(p_{best,i}-x_i\right)+c_2\cdot r_2\cdot\left(g_{best}-x_i\right)$$

$v_i(t)$：粒子 i 在第 t 代的速度。

w：惯性权重，控制粒子搜索的新旧平衡。

c_1,c_2：学习因子，控制粒子对自身和群体最优解的依赖程度。

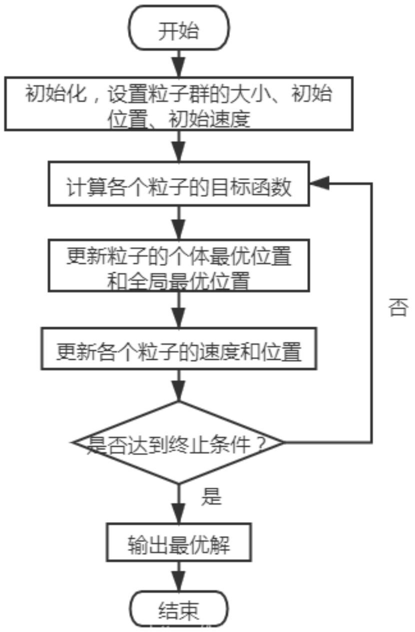
r_1,r_2：随机数，取值在 $[0,1]$。

（2）位置更新公式：

$$x_i(t+1)=x_i(t)+v_i(t+1)$$

三、算法实现

（一）算法流程图



（二）实现步骤

（1）定义目标函数

objFunc = @(x) 15 + x(1)^2 + x(2)^2 - 10 * (cos(2 * pi * x(1)) + cos(2 * pi * x(2)));

目标函数$f(x_1,x_2)$被定义为一个匿名函数，接受输入向量x=[x_1,x_2]，返回其函数值。

（2）设置粒子群算法参数

options = optimoptions('particleswarm', 'SwarmSize', 50, 'MaxIterations', 200);

SwarmSize：粒子的数量，表示搜索空间中同时移动的解的个数（50）。

MaxIterations：最大迭代次数，决定算法的搜索轮数（200）。

（3）定义搜索范围

lb = [-5, -5]; % 下界

ub = [5, 5]; % 上界

搜索空间范围被限制为 $x_1,x_2 \in [-5,5]$.

（4）调用 PSO 求解最小值

options = optimoptions('particleswarm', 'SwarmSize', 50, 'MaxIterations', 200);

objFunc = @(x) 15 + x(1)^2 + x(2)^2 - 10 * (cos(2 * pi * x(1)) + cos(2 * pi * x(2)));

lb = [-5, -5]; % 下界

ub = [5, 5];   % 上界

[bestSolMin, minVal] = particleswarm(objFunc, 2, lb, ub, options);

• 参数：

objFunc：目标函数。

options：设置 PSO 算法参数。

• 返回值：

bestSolMin：找到的最优解（即x_1,x_2的值）。

minVal：目标函数在最优解处的最小值。

（5）求解最大值

objFuncMax = @(x) -(15 + x(1)^2 + x(2)^2 - 10 * (cos(2 * pi * x(1)) + cos(2 * pi * x(2)))); % 此处为了与上方 objFunc 作区分

[bestSolMax, maxValNeg] = particleswarm(objFuncMax, 2, lb, ub, options);

maxVal = -maxValNeg; % 恢复最大值

（6）输出结果

fprintf('函数的最小值为：%.4f，出现在点 (x1, x2) = (%.4f, %.4f)\n', minVal, bestSolMin(1), bestSolMin(2));

fprintf('函数的最大值为：%.4f，出现在点 (x1, x2) = (%.4f, %.4f)\n', maxVal, bestSolMax(1), bestSolMax(2));

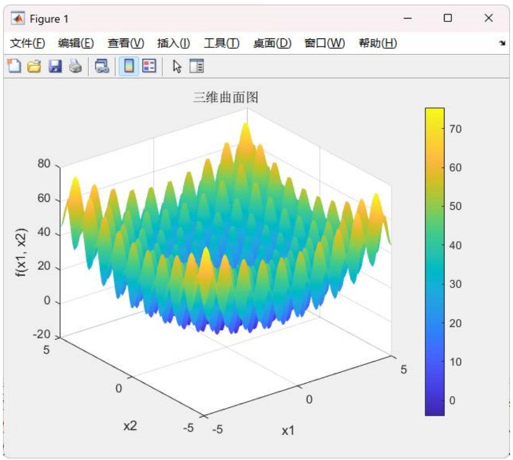
四、程序代码 [详见附件“liziqun.m”]

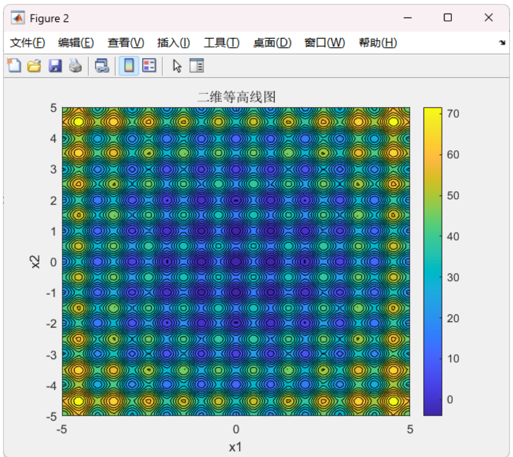
```
clc; clear; close all;
% 定义变量范围
x1 = linspace(-5, 5, 100);
x2 = linspace(-5, 5, 100);
[X1, X2] = meshgrid(x1, x2);
% 定义目标函数
f = @(x1, x2) 15 + x1.^2 + x2.^2 - 10 * (cos(2 * pi * x1) + cos(2 * pi * x2));
Z = f(X1, X2);
% 绘制三维曲面图
figure;
surf(X1, X2, Z);
title('三维曲面图');
xlabel('x1'); ylabel('x2'); zlabel('f(x1, x2)');
shading interp;
colorbar;
% 绘制二维等高线图
```

```
figure;
contourf(X1, X2, Z, 20);
title('二维等高线图');
xlabel('x1'); ylabel('x2');
colorbar;
% 粒子群算法求最小值
options = optimoptions('particleswarm', 'SwarmSize', 50, 'MaxIterations', 200);
objFunc = @(x) 15 + x(1)^2 + x(2)^2 - 10 * (cos(2 * pi * x(1)) + cos(2 * pi * x(2)));
lb = [-5, -5]; % 下界
ub = [5, 5];     % 上界
[bestSolMin, minVal] = particleswarm(objFunc, 2, lb, ub, options);
% 粒子群算法求最大值
objFuncMax = @(x) -(15 + x(1)^2 + x(2)^2 - 10 * (cos(2 * pi * x(1)) + cos(2 * pi * x(2))));
[bestSolMax, maxValNeg] = particleswarm(objFuncMax, 2, lb, ub, options);
maxVal = -maxValNeg; % 恢复最大值
% 打印结果
fprintf('函数的最小值为：%.4f，出现在点 (x1, x2) = (%.4f, %.4f)\n', minVal, bestSolMin(1), bestSolMin(2));
fprintf('函数的最大值为：%.4f，出现在点 (x1, x2) = (%.4f, %.4f)\n', maxVal, bestSolMax(1), bestSolMax(2));
```

五、结果分析

（一）运行结果





命令行窗口

```
Optimization ended: relative change in the objective value
over the last OPTIONS.MaxStallIterations iterations is less than OPTIONS.FunctionTolerance.
Optimization ended: relative change in the objective value
over the last OPTIONS.MaxStallIterations iterations is less than OPTIONS.FunctionTolerance.
函数的最小值为：-5.0000，出现在点 (x1, x2) = (0.0000, 0.0000)
函数的最大值为：75.7066，出现在点 (x1, x2) = (4.5230, 4.5230)
```

（二）实验总结

本实验通过 MATLAB 实现了目标函数的三维曲面图和二维等高线图的绘制，并利用粒子群优化算法（PSO）成功求解了函数的最小值和最大值。在可视化部分，直观展示了函数在定义域内的变化趋势，为极值分析提供了参考依据。在优化部分，通过设置合理的参数（如粒子数量和迭代次数），PSO 算法准确地找到了全局最优解，验证了其在复杂非线性函数优化中的高效性和全局搜索能力。

本实验充分体现了粒子群算法的简单性与实用性，为后续优化算法研究提供了良好的基础。