**安徽大学集成电路学院**

**《算法设计与分析》第八次报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 算法设计与分析 | | | 日期 |  |
| 专业 |  | 学号 |  | 姓名 |  |
| 【实验目的】  本实验旨在通过使用MATLAB实现基于邻接矩阵和特征值分析的聚类算法，达到以下目标：  1. 掌握使用邻接矩阵表示数据点之间关系的方法。  2. 理解并实现基于特征值分析的聚类数自动选择技术。  3. 学习如何结合特征向量和K-means算法进行数据聚类。  4. 实践可视化技术，包括散点图和条形图的绘制，以直观展示聚类结果和特征值间隔。 | | | | | |
| 【实验原理（预习）】  本实验基于谱聚类的基本思想，利用邻接矩阵的特征值和特征向量进行数据聚类。主要原理如下：  1. 邻接矩阵构建：  对于给定的n个数据点，构建一个n×n的邻接矩阵A。矩阵中的每个元素表示第i个点和第j个点之间的欧氏距离：    这个矩阵是对称的，即。  2. 特征值分解：  对邻接矩阵A进行特征值分解：    其中是包含特征值的对角矩阵，U是对应的特征向量矩阵。  3. 最佳聚类数选择：  计算特征值间隔（Eigengap），定义为相邻特征值的相对差：    选择最大间隔对应的索引作为最佳聚类数k。  4. 降维与聚类：  选取前k个特征向量形成矩阵。每一行代表原始数据点在k维空间中的新表示。  5. K-means聚类：  在降维后的空间中应用K-means算法，将n个点分为k类。  理论基础：  谱聚类的核心思想是利用图的拉普拉斯矩阵的特征向量来表示数据，本实验中直接使用邻接矩阵作为简化。  特征值间隔法（Eigengap heuristic）基于这样的观察：当存在明显的聚类结构时，前k个最小特征值会相对接近，而第k个和第k+1个特征值之间会出现较大间隔。  使用特征向量进行降维可以揭示数据的内在结构，有助于后续的聚类过程。  通过这种方法，我们可以自动确定合适的聚类数，并在降维空间中更有效地进行聚类，特别是对于非球形或复杂结构的数据集。 | | | | | |

|  |  |  |
| --- | --- | --- |
| 【实验内容与记录（题号、操作步骤、数据记录与处理、附图编号、代码等）】  matlab代码   |  | | --- | | Kruskals.m | | clc; clear; close all;  % 邻接矩阵聚类  n = 12;  [x, y] = ginput(n);  plot(x, y, 'r\*')  hold on  % 计算邻接矩阵  A = zeros(n);  for i = 1:n      for j = i:n          A(i, j) = sqrt((x(i) - x(j)) ^ 2 + (y(i) - y(j)) ^ 2);          A(j, i) = A(i, j); % 确保矩阵是对称的      end  end  % 计算特征值和特征向量  [U, V] = eig(A);  V = diag(V); % 提取对角线元素  % 计算特征值间隔  Gap = abs(diff(V) ./ V(2:end));  % 自动选择最佳聚类数  [~, best\_k] = max(Gap);  best\_k = best\_k + 1; % 因为Gap的长度比n小1  % 使用K-means进行聚类  Y = U(:, 1:best\_k);  [idx, ~] = kmeans(Y, best\_k);  % 绘制聚类结果  colors = hsv(best\_k);  for i = 1:best\_k      cluster = idx == i;      scatter(x(cluster), y(cluster), 50, colors(i,:), 'filled')  end  title(sprintf('最佳聚类数: %d', best\_k))  legend('原始数据点', 'Location', 'bestoutside')  hold off  % 显示Gap值  figure;  bar(Gap);  title('特征值间隔');  xlabel('间隔索引');  ylabel('间隔值'); |     得到了正确的结果 |

|  |
| --- |
| 【小结与讨论】  本实验通过实现基于邻接矩阵和特征值分析的聚类算法，我获得了以下关键认识和讨论点：  1. 算法有效性：  实验结果表明，基于特征值分析的聚类方法能够有效地识别数据中的自然分组。通过自动选择最佳聚类数，算法展现了其适应性和灵活性。  2. 计算复杂度：  对于大规模数据集，计算完整的邻接矩阵和其特征值可能会成为计算瓶颈。在实际应用中，可能需要考虑使用稀疏矩阵表示或近似算法来提高效率。  3. 参数敏感性：  虽然本算法能够自动选择聚类数，但其性能仍可能受到初始数据点选择和K-means初始化的影响。多次运行实验并比较结果可能有助于评估算法的稳定性。  总的来说，本实验不仅加深了我们对聚类算法的理解，也提高了使用MATLAB进行数据分析和算法实现的能力。通过理论学习与实践相结合，我们更好地掌握了如何将抽象的数学概念转化为具体的编程实现，这对于未来的算法设计与分析课程学习具有重要意义。 |