****

**《数据挖掘》实验报告**

|  |  |
| --- | --- |
| **专 业** | **人工智能** |
| **学 号** |  |
| **姓 名** |  |

**2024-2025-1学期**

**南京邮电大学**

**实验3 基于距离度量的****异常点检测**

一、实验目的：

1. 通过距离度量方式对异常点数据进行甄别。熟悉欧氏距离(Euclidean Distance)、马氏距离(Mahalanobis Distance)等距离度量函数；

2. 掌握如何识别数据中存在的异常点并进行分析；

3. 熟悉华为openGauss数据库的部署、使用。

4. 熟悉python数据处理工具箱以及基于欧氏、马氏等距离度量的异常点监测代码编写。

5. 熟悉利用python驱动openGauss数据库。

6. 熟悉Linux操作系统基本指令。

二、实验设备

1. 硬件：微型计算机。

2. 软件：操作系统openEuler、python3.7及以上版本、openGuass数据库、Data Studio数据库管理工具。

三、实验内容与步骤

1. 在虚拟机+openEuler操作系统或华为云服务器上安装部署openGauss数据库。

2. 配置python3.7环境及Python-psycopg驱动包。

3. 在windows系统配置Data Studio数据库管理工具，远程连接数据库，导入数据。

4. 编写基于距离的异常点监测程序

（1）本实验提供数据集，共138个数据点，其中有10个异常值，找出10个异常点。

（2）读取数据，设置合适阈值，分别用欧氏、马氏距离进行数据分析。

5. 使用psycopg包连接数据库，获取数据，运行异常点监测程序。

四、实验要求

使用python连接驱动openGauss数据库，编程实现欧氏、马氏距离对于数据异常点的检测与匹配。

五、实验步骤、原理以及实验结果分析**（共70’；其中实验步骤30’，需给出阶段性运行结果或者系统部署界面截图；原理介绍20’；实验结果截图与分析20’）**

实验步骤：

1. 部署 openGauss 数据库、Data Studio 导入数据
2. 安装 openGauss
3. 参考实验指导书或华为云文档，在 openEuler 或云服务器（ECS）上部署 openGauss 数据库。
4. 切换到 omm 用户启动数据库服务：gs\_om -t start命令
5. Data Studio 远程连接

在 Windows 客户端安装 Data Studio，输入服务器 IP、端口(6000)、数据库名(postgres)、用户(omm / 自定义)及密码，成功连接后可看到数据库的树状结构。

1. 导入data1.txt
2. 在 Data Studio 中为数据库创建新表 data(x float, y float)。
3. 右键该表 -> 导入数据 -> 指定文件 data1.txt -> 选择 UTF-8，分隔符,，若无表头则取消“包含表头”勾选 -> 点击导入，查看插入结果是否与源文件行数一致(共 138 条)。
4. 基于欧氏距离、马氏距离的异常点检测

欧式：  
import numpy as np  
from scipy.spatial.distance import euclidean  
  
*# 读取数据*data = np.loadtxt('data1.txt', delimiter=',')  
  
*# 计算每个点与其前后20个点的欧氏距离*def compute\_local\_euclidean\_distances(data, window=20):  
 n = data.shape[0]  
 distances = np.zeros(n)  
 for i in range(n):  
 start = max(0, i - window)  
 end = min(n, i + window + 1)  
 local\_points = np.concatenate([data[start:i], data[i + 1:end]])  
 distances[i] = np.mean([euclidean(data[i], point) for point in local\_points])  
 return distances  
  
*# 找出一个离群点*def find\_single\_outlier(distances):  
 return np.argmax(distances)  
  
*# 主函数*def detect\_outliers(data, n\_outliers=10):  
 original\_indices = np.arange(data.shape[0])  
 outliers = []  
  
 for \_ in range(n\_outliers):  
 distances = compute\_local\_euclidean\_distances(data)  
 outlier\_index = find\_single\_outlier(distances)  
 outliers.append(original\_indices[outlier\_index])  
  
 *# 从数据集和索引数组中删除离群点* data = np.delete(data, outlier\_index, axis=0)  
 original\_indices = np.delete(original\_indices, outlier\_index)  
  
 return np.array(outliers)  
  
  
*# 检测离群点*outliers = detect\_outliers(data.copy())  
print("使用马氏距离检测到的前10个离群点索引:", np.sort(outliers + 1)) *# 加1是为了与1-based索引匹配*

实验结果：



马氏：  
import numpy as np  
from scipy.spatial.distance import mahalanobis  
  
*# 读取数据*data = np.loadtxt('data1.txt', delimiter=',')  
  
*# 计算马氏距离*def compute\_mahalanobis\_distances(data):  
 n = data.shape[0]  
 cov\_matrix = np.cov(data.T)  
 inv\_cov\_matrix = np.linalg.inv(cov\_matrix)  
 dist\_matrix = np.zeros((n, n))  
 for i in range(n):  
 for j in range(i + 1, n):  
 dist = mahalanobis(data[i], data[j], inv\_cov\_matrix)  
 dist\_matrix[i, j] = dist\_matrix[j, i] = dist  
 return dist\_matrix  
  
  
*# 找出一个离群点*def find\_single\_outlier(dist\_matrix):  
 mean\_distances = np.mean(dist\_matrix, axis=1)  
 return np.argmax(mean\_distances)  
  
  
*# 主函数*def detect\_outliers(data, distance\_method='euclidean', n\_outliers=10):  
 original\_indices = np.arange(data.shape[0])  
 outliers = []  
  
 for \_ in range(n\_outliers):  
 if distance\_method == 'mahalanobis':  
 dist\_matrix = compute\_mahalanobis\_distances(data)  
 else:  
 raise ValueError("Unsupported distance method")  
  
 outlier\_index = find\_single\_outlier(dist\_matrix)  
 outliers.append(original\_indices[outlier\_index])  
  
 *# 从数据集和索引数组中删除离群点* data = np.delete(data, outlier\_index, axis=0)  
 original\_indices = np.delete(original\_indices, outlier\_index)  
  
 return np.array(outliers)  
  
*# 使用马氏距离检测离群点*mahalanobis\_outliers = detect\_outliers(data.copy(), 'mahalanobis') + 1  
print("使用马氏距离检测到的前10个离群点索引:", np.sort(mahalanobis\_outliers))

实验结果：



六、思考与总结**（20’，每题10’）**

1. 欧氏和马氏距离分别有何特点和差异？

欧氏距离和马氏距离是两种常用的距离度量方法。

* **欧氏距离**：计算两点之间的直线距离，适用于各特征之间尺度相似的数据。它的优点是简单易懂，计算速度快，但对特征尺度差异敏感，可能导致距离失真。
* **马氏距离**：考虑了特征之间的相关性和方差，通过协方差矩阵来标准化数据，使其不受特征尺度的影响。马氏距离适用于特征间存在相关性的数据，但计算较复杂，需要求协方差矩阵的逆。

**主要差异：**

1. **尺度敏感性**：欧氏距离对特征尺度敏感，马氏距离通过协方差矩阵消除了这一影响。
2. **考虑相关性**：欧氏距离忽略特征间相关性，马氏距离考虑了特征之间的相关性。
3. **计算复杂度**：欧氏距离计算简单，马氏距离计算更复杂，需要求协方差矩阵的逆。

2. 两种度量距离适用于何种异常点监测？

在异常点监测中，欧氏距离和马氏距离各自有不同的适用场景：

**欧氏距离：**

* **适用场景**：欧氏距离适用于特征之间尺度一致且相对独立的情况。在数据集中的特征尺度相似，且没有显著的相关性时，欧氏距离能够有效地识别异常点。
* **异常点监测**：当数据集中的样本之间的关系主要是基于直接的几何距离时，欧氏距离能够通过度量样本与其他点之间的直接距离来发现离群点（例如，数据点与其邻近点之间的距离过大时即为异常点）。它适合于简单的二维或三维数据集。

**马氏距离：**

* **适用场景**：马氏距离适用于特征之间具有相关性或者特征的尺度差异较大的情况。它能有效处理特征之间的相关性问题，对于不同尺度的特征进行标准化，因此在高维数据中表现更好。
* **异常点监测**：马氏距离能够更好地识别基于特征协方差关系的异常点。对于有显著特征依赖或尺度差异的高维数据，马氏距离能够通过考虑协方差矩阵来衡量样本间的距离，从而更准确地识别离群点。它适合用于处理复杂的数据集，尤其是在数据特征间相关性较强时。

七、实验过程遇到的问题以及心得体会**（10’）**

问题：

1. **数据库部署和配置问题**：在虚拟机或华为云服务器上安装openGauss数据库可能会遇到环境配置、依赖库缺失或网络连接问题。确保操作系统和数据库版本兼容、虚拟机资源足够以及网络配置正确是非常重要的。
2. **Python环境配置**：在Windows系统上配置Python 3.7环境和安装psycopg驱动包时，可能会遇到版本不匹配或依赖安装问题。确保Python版本、驱动包和数据库连接配置的正确性是解决问题的关键。
3. **Data Studio配置问题**：在配置Data Studio进行远程数据库连接时，可能会遇到连接失败、认证失败等问题。检查数据库连接信息（如IP、端口、用户名和密码）是否正确，以及网络是否允许数据库的远程连接。
4. **异常点检测中的阈值选择**：在使用欧氏距离和马氏距离进行异常点检测时，选择合适的阈值非常重要。如果阈值设置过高，可能无法检测到异常点；如果过低，可能会误判正常点为异常点。需要通过实验调优阈值。
5. **数据质量问题**：数据集中的噪声或不完整数据可能会影响异常点检测结果。数据预处理（如去重、缺失值处理等）是确保结果准确性的关键。

**心得体会**：

这个实验帮助我理解了如何在实际环境中配置和部署数据库，以及如何通过编程实现异常点检测。通过欧氏距离和马氏距离对数据进行分析，学习到如何选择适合的算法和阈值进行异常点监测，也提高了我对数据预处理和数据库管理的理解。