计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(2):160−169 [doi: 10.15888/j.cnki.csa.008968]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于二阶近邻的异常检测①

卢梦茹, 周昌军, 刘华文, 徐晓丹

(浙江师范大学 数学与计算机科学学院, 金华 321004)
通信作者: 周昌军, E-mail: zhou-chang231@163.com

摘　要: 对盈千累万且错综复杂的数据集进行分析, 是一个非常具有挑战性的任务, 检测数据中的异常值的技术在该任务中发挥着举足轻重的作用. 通过聚类捕获异常的方式, 在日趋流行的异常检测技术中是最为常用的一类方法. 文中提出了一种基于二阶近邻的异常检测算法 (anomaly detection based second-order proximity, SOPD), 主要包括聚类和异常检测两个阶段. 在聚类过程中, 通过二阶近邻的方式获取相似性矩阵; 在异常检测过程中, 根据簇中的点与簇中心的关系, 计算聚类生成的每一个簇中的所有的点与该簇中心的距离, 捕捉异常状态, 并把每个数据点的密度考虑进去, 排除簇边界情况. 二阶近邻的使用, 使得数据的局部性以及全局性得以被同时考虑, 进而使得聚类得到的簇数减少, 增加了异常检测的精确性. 通过大量实验, 将该算法与一些经典的异常检测算法进行比较, 结果表明, SOPD 算法整体上性能较好.

关键词: 异常检测; 二阶近邻; 相似性矩阵; 密度; 全局性; 机器学习; 数据挖掘

引用格式: 卢梦茹,周昌军,刘华文,徐晓丹.基于二阶近邻的异常检测.计算机系统应用,2023,32(2):160–169. http://www.c-s-a.org.cn/1003-3254/8968.html

Anomaly Detection Based on Second-order Proximity

LU Meng-Ru, ZHOU Chang-Jun, LIU Hua-Wen, XU Xiao-Dan

(College of Mathematics and Computer Science, Zhejiang Normal University, Jinhua 321004, China)

Abstract: The analysis of numerous and intricate data sets is a highly challenging task, in which the technique to detect outliers in data plays a pivotal role. Capturing anomalies by clustering is the most common method among the increasingly popular anomaly detection techniques. This study proposes an anomaly detection algorithm based on second-order proximity (SOPD), which includes clustering and anomaly detection stages. During clustering, the similarity matrix is obtained by second-order proximity. During anomaly detection, the relationships between points in the cluster and the center of the cluster are employed to calculate the distance of all the points in each cluster generated by clustering from the center of the cluster and capture the anomalous state. The density of each data point is also taken into account to exclude the cases of cluster boundaries. The use of second-order proximity enables the locality and globality of the data to be considered simultaneously, which reduces the number of the obtained clusters and increases the accuracy of anomaly detection. Moreover, this study compares this algorithm with some classical anomaly detection algorithms through massive experiments, and the result shows that the SOPD-based algorithm performs well overall.

Key words: anomaly detection; second-order proximity; similarity matrix; density; globality; machine learning; data mining

数据分析过程中, 总是存在一些与其他数据大相径庭的非正常对象, 这种异常现象影响着数据挖掘的准确性, 甚至对其应用领域造成巨大的损失. 异常的产生可能会表现为人为错误、技术或机械方面的故障、

① 基金项目: 国家自然科学基金 (61976195)
收稿时间: 2022-07-14; 修改时间: 2022-09-07; 采用时间: 2022-09-14; csa 在线出版时间: 2022-11-29
CNKI 网络首发时间: 2022-11-30

噪声、系统变化和欺诈行为等[1]. 比如: 顾客的某次消费明显不同于以往正常的花销记录; 中转设备发生故障时, 通信或数据的传输出现错误; 路由器在短时间内突然收到大量交换包[2]. 为了避免这些异常的现象对数据挖掘的影响, 异常检测技术得到广泛应用. 虽然, 大部分的异常检测方法在技术上基本相同, 但是, 它们会有许多不同的名字, 比如离群值检测、异常值检测、新颖值检测、噪声检测或异常挖掘等[3]. 异常检测是数据挖掘中的一个重要问题, 目的是把那些与其他数据点显著不同的独特或不寻常的数据对象检测出来[4].

异常检测的算法可以大致分为 4 大类: 基于统计的异常检测算法[5,6]、基于距离的异常检测算法[7–9]、基于密度的异常检测算法[10–12] 和基于聚类的异常检测算法[13–16]. 基于统计的异常检测算法分为有参数和非参数两大类检测算法, 与之前的算法相比, 提升了异常检测的性能. 但是, 其数据分布需要提前确定, 基于距离的异常检测算法弥补了这一缺点, 不需要考虑数据分布. 基于距离的异常检测算法主要是基于目标对象邻域的稀疏状态与其异常状态之间的相关性假设[9]. 基于密度的异常检测算法根据数据点的密度检测异常, 如果某个数据点的密度低于其周围邻居的密度, 那么该数据点就是异常点[12]. 基于密度的异常检测算法主要从局部的角度来考虑异常情况, 可以识别更多有意义的局部异常数据点.

聚类作为一种无监督的方法, 根据相似度度量对数据进行分组, 其目标是获得高的簇内相似性 (即类簇内的数据是相似的) 和低的簇间相似性 (即来自不同类簇的数据是不同的)[13]. 那么, 基于聚类的异常检测算法是根据数据点之间的相似性将数据集中的数据点分类为多个聚类, 异常值是指不位于任何聚类中或离最近聚类的质心很远的数据点[17]. DBSCAN (density-based spatial clustering of applications with noise)[14] 可以发现任意形状的簇, 将数据有效的分开, 能够在低密度区域中识别噪声, 即噪声数据的密度必定低于簇的密度. Huang 等[15] 提出了一种新的基于聚类的异常检测算法 (relative outlier cluster factor, ROCF), 不需要提前指定异常值的个数, 只需要一个参数 k 表示邻居的数量, ROCF 也可以通过构造决策图自动计算出数据集的离群率. 整体而言, ROCF 检测算法在获取离群值和离群值簇时更加倾向于自适应性. 为了提升大规模数据集中的异常检测性能, Nozad 等[16] 提供了基于批处理的异常检测算法, 是一种利用聚类方式检测异常的算法, 在此过程中通过密度实现聚类, 该算法对具有相关性或不相关性特征的高斯簇的检测效果表现更好, 同样适用于遵循任意分布的凸形簇.

本文提出了一种基于二阶近邻的异常检测算法, 通过二阶近邻获取相似性矩阵完成聚类过程, 利用每个簇中的数据点与其所属簇中心的关系捕捉异常情况. 该算法的相关假设是: 正常的数据点更加靠近所属簇中心, 远离簇中心的数据点则表现为异常; 一般情况下, 异常数据点的密度比正常数据点低. 基于二阶近邻的异常检测算法主要分为两阶段: 聚类阶段和异常检测阶段. 在聚类阶段, 通过二阶近邻的方式获取相似性矩阵, 这一过程捕获到了数据在聚类过程中的全局结构. 第二阶段考虑了每个簇中的数据点与其所属簇中心之间的关系, 根据正常数据点与簇中心关系更加紧密, 异常数据点与簇中心相距较远这个假设, 计算每个簇中所有的数据点与其所属簇中心之间的距离; 根据密度较小的数据点更有可能成为异常点的假设, 计算每个数据点的密度. 结合距离值与密度值, 最终获取每个数据点的异常值分数.

本文的主要贡献有以下两点.

(1) 提出了一种基于二阶近邻的异常检测算法, 利用二阶近邻获得相似性矩阵, 综合考虑数据的局部性和全局性结构信息, 深入挖掘数据点与数据点之间的关系, 增强了捕获它们之间的相似性的能力, 对于较稀疏的数据集会有更好的性能;

(2) 在异常检测阶段, 根据簇中的数据点与所属簇中心之间的关系判断异常情况, 同时考虑到簇与簇之间的临界处可能会存在数据点较密集的情况, 根据密度较小的数据点更有可能成为异常点的假设, 利用数据点的局部密度排除边界情况, 减少错误识别异常值的可能性.

1 相关概念

如果近邻图中任意两个顶点x_i和x_j之间由一条边连接, 即边上的权值$W_{ij}>0$, 那么这两个点则为一阶近似[18]. 一阶近似描述了顶点之间的成对相似性, 意味着图中由一条边连接的顶点往往是相似的. 例如, 观看同一场音乐会的两个人必定有相似的兴趣点. 因此, 保持图节点之间的一阶近似性非常必要. 图 1 中的顶点 E 和顶点 F 之间存在一条边连接, 它们之间则为一阶近似. 但是, 现实世界中的信息网络一般比较稀疏, 许多相似

的点往往并不存在直接相连的边. 一阶近似只考虑到了局部结构, 二阶近邻[18] 则可以捕获全局网络结构, 弥补一阶近似的不足. 设$A_i = \{a_{i,1}, a_{i,2}, a_{i,3}, \cdots, a_{i,|V|}\}$表示顶点$x_i$与其他顶点之间的一阶近似性. 由邻域$A_i$和$A_j$的相似性来确定二阶近邻. 有许多共同邻居的两个顶点为二阶近邻, 即使它们之间没有一阶近似性, 但仍具有相似性. 例如, 一般情况下, 有许多共同朋友的两个人往往也是朋友. 二阶近邻作为一种定义两个顶点的相似性的良好度量工具, 表征数据的全局结构, 可以高度丰富顶点之间的关系, 达到深度挖掘数据的目的. 图 1 中的顶点 E 和顶点 G 之间并不存在相连的边, 但是, 它们共享邻居顶点 A、顶点 B、顶点 C 和顶点 D, 因此, 顶点 E 和顶点 G 之间是二阶近邻, 存在相似性.

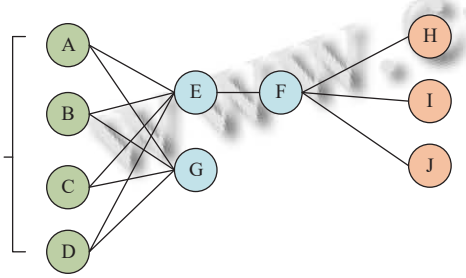


图 1 二阶近邻关系图

基于亲和传播的聚类 (affinity propagation, AP)[19] 是在 2007 年被提出的一种重要的聚类算法, 主要结合相似性和消息传递进行聚类, 将数据点对之间的相似性作为消息传递的一个输入度量. 真实值消息在数据点之间交换, 直到最优候选范例和相应的集群逐渐出现. 具体的消息传递过程如图 2 所示. 图 2(a) 表示的责任 (responsibility) $R(x_i, x_j)$是从数据点x_i发送到候选聚类中心x_j, 指出每个数据点选择候选聚类中心x_j而不是其他候选聚类中心的强烈程度. 图 2(b) 表示的可用性 (availability) $A(x_i, x_j)$是从候选聚类中心x_j发送到数据点x_i, 指出候选聚类中心x_j成为数据点x_i的聚类中心的可能程度. AP 算法在聚类时不需要事先初始化聚类中心, 每个数据点都有可能成为聚类中心, 消息传递的过程经过不断迭代可以自动确定聚类中心. 但是, 原始的亲和传播聚类算法仅使用数据样本的欧几里得距离作为相似度计算的唯一标准, 增加了算法在高维稀疏性数据集上的局限性.

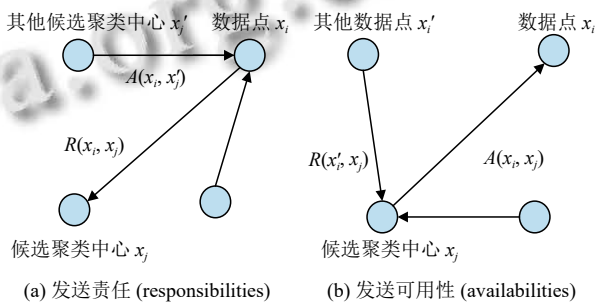


图 2 消息传递过程

2 算法描述

基于二阶近邻的异常检测算法通过二阶近邻获得相似性矩阵实现聚类, 以聚类结果为基础, 根据簇中的点与簇中心的关系, 计算每一个簇中的所有的点与该簇中心的距离, 捕获异常情况. 同时考虑每个数据点的局部密度, 避免在簇与簇之间的边界处错误识别异常值的情况. 基于距离值和局部密度值得到每个数据点的异常值分数, 并根据异常值分数排序, 取其值较大的数据点作为异常点. 该算法的流程图如图 3 所示.

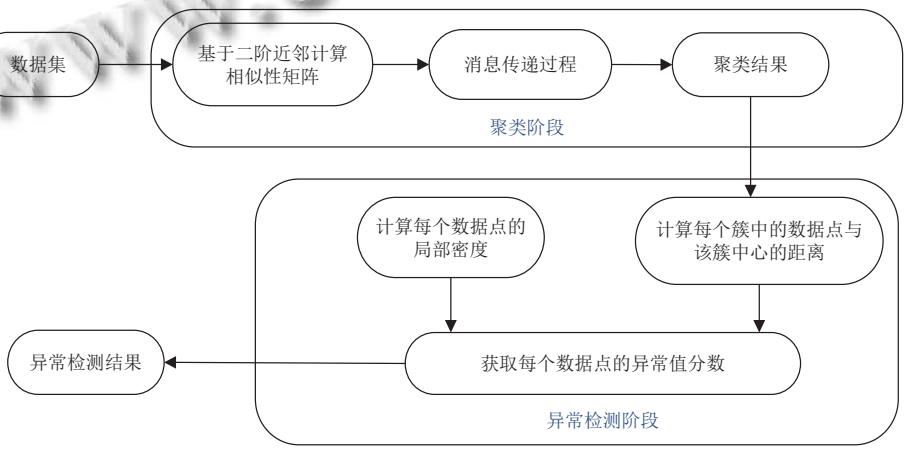


图 3 基于二阶近邻的异常检测流程

2.1 聚类阶段

本节主要对整个数据集中的数据进行聚类. 在此过程中联合一阶近似性和二阶近邻捕获相似性, 深入探究数据点与数据点之间的紧密关系, 然后经过消息传递过程实现聚类. 这是异常检测的初始阶段.

2.1.1 二阶近邻求相似性

相似性表示数据点与数据点之间的密切程度, 聚类往往通过相似性度量所有数据点之间的关系, 对于数据的分布则没有太大的要求. 聚类过程中将相似性较高的那些数据点聚集在一起形成一个簇, 而相似性较低的数据点之间则处于一个远离的状态, 据此可以很好地将不同类的数据分别聚集成不同的簇. 计算相似性的方式有很多种, 此处将数据点x_i和数据点x_j之间的初始相似性$S(x_i,x_j)$用欧几里得距离表示, 即:

$$S(x_i,x_j)=-\|x_i-x_j\|^2 \tag{1}$$

欧几里得距离原本是正值, 但是两个数据点之间的距离越远, 它们的相似性越小, 因此在距离值前面加上了一个负号, 使得初始相似性为负值. 所有数据点之间的初始相似性存放在相似性矩阵S中, 矩阵S中的值所表示的是数据点与数据点之间的一种成对近似性, 即是所谓的一阶近似性, 能够很好地表示数据的局部信息, 因此, 矩阵S也被称为一阶近似性矩阵. 因而, 为了更进一步刻画不同数据点之间的相近性水平, 通过使用更高阶的近似性来探究数据集内部的数据点与数据点的紧密程度. 在这里, 利用每个数据点的邻域关系, 通过二阶近邻来描绘一阶近似性矩阵S中拥有共同邻居的数据点, 增强这些数据点与其二阶近邻之间的相似性. 即以一阶近似性矩阵为基础, 通过向数据点周边的邻居进行扩展, 根据数据点与数据点之间共享邻居的程度, 获取它们之间的近似性并存入到二阶近邻矩阵中. 二阶近邻弥补了一阶近似性的不足, 充分地捕获了数据的全局结构信息. 因此, 联合一阶近似性和二阶近邻重构相似性矩阵, 能够更深入地度量数据点之间的相似性. 在基于随机游走的概率集合中, 一阶和二阶近邻分别是 1 步和 2 步随机游走的概率转移矩阵[18], 因此二阶近邻矩阵由 2 步随机游走的概率转移矩阵所表示, 即$S^2=S\times S$, 那么一阶近似性矩阵和二阶近邻矩阵的联合信息则可以通过如下方式计算:

$$S=\alpha\times S+(1-\alpha)\times S^2 \tag{2}$$

通过为一阶近似性矩阵和二阶近邻矩阵分配合适的权值, 重新构造相似性矩阵S, 综合考虑了数据的局部性和全局性结构信息. 其中, α是一个参数, 表示一阶近似性矩阵的权值, $(1-\alpha)$则是为二阶近邻矩阵所分配的权值, α的取值范围一般是 (0.5, 1), 参数α的值越大, 一阶近似性矩阵的权重就越大. $\alpha\times S$表示相似性矩阵的一阶近似性, $(1-\alpha)\times S^2$表示的是相似性矩阵的二阶近邻. 为了保证一阶近似性矩阵和二阶近邻矩阵中的数据具有相同的分布形式, 提高聚类的精确度, 重新构造相似性矩阵之前, 需要对一阶近似性矩阵和二阶近邻矩阵进行归一化处理, 在这里使用的是按行归一化方式, 并始终保持矩阵原来的对称性.

2.1.2 消息传递

把通过二阶近邻获得的相似性矩阵传入到消息传递过程中, 不断更新迭代得到最后的聚类结果. 最初数据集中的每一个数据点都可以被看作候选聚类中心, 即需要向消息传递过程中输入一个表示数据点x_j的$S(x_j,x_j)$, $S(x_j,x_j)$的值越大, 数据点x_j越有可能成为一个聚类中心. $S(x_j,x_j)$的值被称为偏好值, 用 P 表示, 评估了数据集中的每一个点成为聚类中心的可能性. 它可以取所有数据点之间的相似性的最小值, 也可以取所有数据点之间的相似性的平均值等.

责任$R(x_i,x_j)$和可用性$A(x_i,x_j)$作为消息传递中的两种消息交换, 在数据点之间的消息更新过程中起着至关重要的作用. $R(x_i,x_j)$反映的是数据点x_j适合成为数据点x_i的候选聚类中心的累积证据, 考虑到了其他有可能成为数据点x_i的候选聚类中心的数据点. $A(x_i,x_j)$反映的是数据点x_i选择数据点x_j作为自己的候选聚类中心的累积证据, 考虑到了其他有可能选择数据点x_j作为候选聚类中心的数据点. 更新之前, 可用性值初始化为 0, 即$A(x_i,x_j)=0$. 责任计算方式如下:

$$R(x_i,x_j)=S(x_i,x_j)-\max_{x'_j\ \text{s.t.}\ x_j\neq x'_j}\{A(x_i,x'_j)+S(x_i,x'_j)\} \tag{3}$$

在第一次迭代中, 因为可用性为 0, $R(x_i,x_j)$表示为数据点x_i和候选聚类中心x_j之间的输入相似性减去数据点x_i与其他候选聚类中心之间的最大相似性. 在之后的迭代中, 当一些数据点被分配给其他候选聚类中心时, 它们的可用性将根据更新规则降到 0 以下. 当$x_i=x_j$时, $R(x_j,x_j)$设置为表示数据点x_j可能成为聚类中心的输入偏好值$S(x_j,x_j)$减去数据点x_i与其他候选聚类中心之间的最大相似性. $R(x_j,x_j)$也被称为自我责任 (self-responsibility), 反映了数据点x_j成为一个聚类中心

的累积证据.

在上述责任更新过程中, 所有的候选聚类中心竞争同一个数据点的所有权, 即成为这个数据点的聚类中心. 而可用性的更新则来自那些数据点中是否每一个候选聚类中心都可以成为一个更好的聚类中心的证据:

$$A(x_i,x_j)=\min\{0,R(x_j,x_j)+\sum_{x_i'\ \text{s.t.}\ x_i'\notin\{x_i,x_j\}}\max\{0,R(x_i',x_j)\}\} \tag{4}$$

$A(x_i,x_j)$为$R(x_j,x_j)$加上候选聚类中心从其他数据点获取到的正的责任的和. 自我可用性 (self-availability) $A(x_j,x_j)$的更新有所不同:

$$A(x_j,x_j)=\sum_{x_i'\ \text{s.t.}\ x_i'\neq x_j}\max\{0,R(x_i',x_j)\} \tag{5}$$

这个消息基于其他数据点发送给数据点x_j的正的责任, 反映了数据点x_j成为聚类中心的累积证据.

$R(x_i,x_j)$和$A(x_i,x_j)$消息更新之后, 数据点x_i的聚类中心可以通过以下方式来确定:

$$E(x_j)=\arg\max_{x_j}\{R(x_i,x_j)+A(x_i,x_j)\} \tag{6}$$

消息传递的过程中通过不断迭代更新, 在形成簇的同时自动确定了每个簇的中心点, 即聚类中心, 完成了聚类.

2.2 异常检测阶段

本节主要是根据第 2.1 节得到的聚类结果进行异常检测. 文中所提出的算法的相关假设中提到正常的数据点更加靠近所属簇中心, 而异常的数据点在所属簇中会距离簇中心较远. 基于上述假设, 那些紧密围绕在簇中心的数据点通常是正常值, 而异常的数据点则稀疏地围绕在簇的边界处. 由此说明正常的数据点往往与所属簇的簇中心有着密切的关系. 因此, 每个数据点的异常情况都可以很好地通过它本身与所属簇中心的距离近似表示. 假设第 2.1 节中基于二阶近邻得到的聚类结果为$\{C_1,C_2,C_3,\cdots,C_m\}$, C_m表示的是第 m 个簇, $C_m=\{x_{1m},x_{2m},x_{3m},\cdots,x_{km}\}$表示第 m 个簇中包含 k 个数据点, 那么, 每个簇中的每个数据点与该簇中心的距离为:

$$dist(x_{km},c_m)=\|x_{km}-c_m\| \tag{7}$$

其中, x_{km}表示的是第 m 个簇中的第 k 个数据点, c_m表示第 m 个簇的簇中心. $dist(x_{km},c_m)$的值越大, 数据点x_{km}越有可能成为一个异常值. 但是, 每个簇与簇之间的边界处可能会存在一些较密集的数据点, 如果只考虑距离问题, 则极有可能会使正常的数据点被误认为是异常点, 极大地影响到异常检测的准确度. 为了排除这种边界情况, 减少错误识别率, 基于密度较小的点更有可能成为异常点的假设, 同时把数据点的密度考虑进去. 最初, 数据点的密度一般是指其特定范围内所包含的数据点的个数. 在基于密度的聚类的定义中, 每一个簇都是由一组密集的数据点所构成, 不同的簇则被那些数据点较稀疏的区域划分开. 因此, 数据点周围越密集, 其密度越大, 越不可能成为异常值. 但是, 这种仅仅考虑数据点周围的邻居个数作为其密度的方式, 只在数据点较密集的情况下比较适用. 为了更好地适应不同类型的数据集, 文中通过采用基于高斯核函数的方式生成数据点的局部密度, 数据点x_i的局部密度ρ_i可以通过以下公式来计算:

$$\rho_i=\sum_{x_i\neq x_j}\mathrm{e}^{-(dist(x_i,x_j)/d_c)^2} \tag{8}$$

其中, $dist(x_i,x_j)$表示数据点x_i和数据点x_j之间的距离, x_j表示整个数据集中除了数据点x_i的每一个数据点. d_c为截断距离, 一般根据经验从所有升序排序的距离值中取一个合适的值作为截断距离. 那么, 作为异常检测过程中的一个重要判断依据, 数据点x_i的异常值分数则可以通过以下公式来获得:

$$score(x_i)=\frac{dist(x_{km},c_m)}{\rho_i} \tag{9}$$

数据点x_i即为数据点x_{km}, 其异常值分数与距离$dist(x_{km},c_m)$成正比, 与密度ρ_i成反比. 即与所属簇中心的距离越远, 且其局部密度越小的数据点更有可能被识别成为异常点. 最后把所有数据点的异常值分数按从大到小的顺序进行排序, 异常点则为异常值分数值较大的数据点.

2.3 总的算法框架

文中所提出的基于二阶近邻的异常检测算法的实现步骤的详细描述如算法 1 所示. 该算法主要分为聚类 (步骤 1–5) 和异常检测 (步骤 6–8) 两个阶段. 聚类阶段, 计算数据点与数据点之间的相似性构造初始相似性矩阵S, 按行归一化相似性矩阵S以及该矩阵的平方$S\times S$, 并保持S和$S\times S$原来的对称性, 在此基础上, 通过一阶近似性和二阶近邻对初始相似性矩阵重新构造; 然后, 把最终构造好的相似性矩阵S、参数α、偏好值

参数 P 传入到消息传递的过程中, 计算责任和可用性的值并重复迭代直到收敛, 获取最终聚类结果, 得到不同的簇. 异常检测阶段, 根据数据点与簇中心的距离关系以及对数据点的密度大小的考虑, 计算每个数据点与所属簇中心的距离值以及该数据点的局部密度, 得到每一个数据点的异常值分数, 对异常值分数按照降序排序, 取异常值分数较大的数据点作为异常点.

算法1. 基于二阶近邻的异常检测算法

输入: 数据集, 参数α, 偏好值参数P

输出: 异常值分数$score$, 异常数据点

阶段1. 聚类

1. 通过式(1)计算数据点之间的相似性并构造初始相似性矩阵S;
2. 通过式(2)以二阶近邻的方式重新构造初始相似性矩阵S;
3. 通过式(3)–式(5)实现消息传递;
4. 重复步骤3直至收敛得到最后的责任值和可用性值;
5. 利用式(6)确定最后的聚类中心, 获取聚类结果$\{C_1,C_2,C_3,\cdots,C_m\}$;

阶段2. 异常检测

6. 通过式(7)计算每个簇中的每个数据点与该聚类中心的距离为 $dist(x_{km},c_m)$;
7. 利用式(8)计算每个数据点的局部密度ρ_i;
8. 由式(9)得到异常值分数$score(x_i)$, 把所有数据点的异常值分数按从大到小的顺序进行排序, 输出其中值较大的作为异常点.

基于二阶近邻的异常检测算法中取偏好值参数 P 为−0.2, 参数α的取值范围一般为 (0.5, 1), 为达到最优的结果, 在该算法中取参数α为 0.9. 在消息更新的过程中, 某些情况下会受到数值振荡的影响, 为了避免这种影响, 每种消息都表示为λ乘以上一次迭代得到的值, 加上$1-\lambda$乘以当前更新得到的值, 影响因子λ的取值范围为 (0, 1), 为获得较好的结果, 在这里取λ的值为 0.9.

算法 1 的计算复杂度主要体现在两个阶段: 聚类阶段 (步骤 1–5) 和异常检测阶段 (从步骤 6–8), 第 1 阶段中, 步骤 1 和步骤 2 计算数据点之间的相似性并构造相似性矩阵需要花费$O(N\times N)$, N表示的是数据集中的数据点的个数, 从步骤 3 到步骤 5 的消息传递过程, 需要不断迭代更新, 需要花费更多的时间, 其复杂度为$O(N^2\times T)$, T表示的是消息传递过程的迭代次数, 而$O(N\times N)$远低于$O(N^2\times T)$, 因此构造相似性矩阵的时间复杂度可以忽略, 聚类阶段的时间复杂度为$O(N^2\times T)$; 异常检测阶段中, 步骤 6 计算簇中的数据点与所属簇中心的距离时应考虑到簇的个数和每个簇中的数据点的数量, 需要花费$O(N-M)$, M为聚类过程形成的簇的个数, 步骤 7 计算数据点的局部密度的时间复杂度为$O(N\times N)$, 步骤 8 需要计算每一个数据点的异常值分数, 因此时间复杂度仅为$O(N)$, $O(N-M)$和$O(N)$与$O(N\times N)$相比, 可以忽略不计, 因此这一阶段的时间复杂度应为$O(N\times N)$. 总的时间复杂度为$O(N^2\times T)+O(N\times N)$, 忽略较低的$O(N\times N)$, 最后的时间复杂度应为$O(N^2\times T)$.

3 实验分析

3.1 实验数据集与实验设置

为了验证文中提出的异常检测算法的有效性, 我们基于 10 个真实的数据集做了一系列的比较实验. 这些数据集包括 WPBC、Ovarian、Wholesale、ForestTypes、Haberman、Ionosphere、Leaf、MUSK、SHS、TAE. 除了 Ovarian 数据集 (https://github.com/therneau/), 其他数据集都来自于 UCI 机器学习仓库 (http://archive.ics.uci.edu/ml/). 所有数据集的信息的简要描述如表 1 所示, 其中“异常标记”表示数据集中的数据点被看作是异常值的标签, 不同数据集有其各自的异常标记信息. 比如, WPBC 数据集中的 47 个被标记为“R”的数据点、Wholesale 数据集中被标记为“Channel 2”的 142 个数据点和 Ovarian 数据集中被标记为“Yes”的 162 个数据点都被看作是异常值.

表 1 实验数据集的简要描述

数据集	数据点的数量	特征个数	异常值个数	异常标记
WPBC	198	33	47	R
Ovarian	253	15 154	162	Yes
Wholesale	440	8	142	Channel 2
ForestTypes	325	27	46	o
Haberman	306	4	81	2
Ionosphere	351	16	126	b
Leaf	340	15	10	36
MUSK	476	166	269	NON-MUSK
SHS	143	6	66	0
TAE	151	5	49	1

实验的过程中将基于二阶近邻的异常检测算法与 6 种经典的异常检测算法进行比较, 这些算法分别为随机离群值选择异常检测算法 SOS (stochastic outlier selection)[20]、主成分分析异常检测算法 PCA (principal component analysis)[21]、基于偏差的线性离群值检测方法 LMDD (linear method for deviation-based outlier detection) [22]、子空间离群值检测算法 SOD (subspace outlier detection)[23]、一级支持向量机异常检测算法 OCSVM (one-class support vector machines)[24]、局部相关积分异常检测算法 LOCI (local correlation integral) [25].

在 SOS 算法[20] 中, 一个数据点与另一个数据点之间的关系, 可以通过亲和性来量化. PCA 算法[21] 作为一种线性降维方法, 将样本在所有特征向量上的投影距离的和作为其异常值分数, 据此确定异常值. LMDD 算法[22] 把平滑因子的概念用到了异常检测中, 该概念表明通过从数据集中删除一个元素的子集, 可以减少一定的差异. SOD 算法[23] 是子空间离群值检测方法, 能够根据数据对象与子空间的邻居的偏离程度来检测高维特征空间中不同子空间中的离群值. OCSVM 算法[24] 是支持向量算法对无标记数据情况的自然扩展, 是一种无监督的异常检测算法. LOCI 算法[25] 提供了一个自动的、数据指定的截断来确定某个点是否为异常值. 以上 6 种异常检测算法的源码皆来自于 pyod 工具包 (https://pyod.readthedocs.io/en/latest/)[26].

每个异常检测算法一般会需要设置不同的参数, 它们的取值会对异常检测的性能产生影响, 因此, 为参数分配适当的值非常重要. 为了保证合理性, 在实验过程中, 各种异常检测算法的参数都会设置为 pyod 工具包中推荐的值. 基于二阶近邻的异常检测算法中的参数设置在算法描述部分中会有详细的说明. 整个实验是在环境为 64 位、8 GB 内存、Intel(R) Core(TM)i7-8550U CPU@1.80 GHz 2 GHz 的 Win10 上进行的.

为了评估不同异常检测算法的性能, 文中采用了 3 种常用的度量指标, 分别是平均精度 AP (average precision)、F_1的最大值$\max F_1$和接受者操作特性曲线 ROC (receiver operating characteristic) 下的面积 AUC (area under curve)[2]. 假设G_h和G_q分别表示真实值和 q 个候选异常值. 精度则为真实异常值在候选异常值中所占的比例:

$$P(q)=\frac{|G_h\cap G_q|}{q} \tag{10}$$

平均精度 AP 就是所有$P(q)$的值的平均值. 召回率则为:

$$R(q)=\frac{|G_h\cap G_q|}{|G_h|} \tag{11}$$

很容易可以看出, $P(q)$显示了候选异常值中的真实异常值的存在情况, $R(q)$则为真实异常值中被检测出来的异常值的情况. 那么, $F_1(q)$的值则可以通过$P(q)$和$R(q)$这两个值来获得, 它是$P(q)$和$R(q)$的调和平均数:

$$F_1(q)=\frac{2P(q)R(q)}{P(q)+R(q)} \tag{12}$$

$\max F_1$是指 q 个候选异常值的$F_1(q)$中的最大值.

接受者操作特性曲线 ROC 是一个由真阳性率和假阳性率构成的图形图, AUC 度量的是 ROC 曲线下的面积, 它作为一种衡量性能优劣的评价指标, 用来表示比较检测性能的数值结果. 它的值总是保持在 [0, 1] 区间内, 而且, 其值越大越好, 平均精度 AP 和$\max F_1$的值也是如此.

3.2 实验结果与讨论

我们分别在 10 个真实数据集上将基于二阶近邻的异常检测算法 (anomaly detection based second-order proximity, SOPD) 和其他的异常检测算法进行了性能测试和评估, 得到不同的实验结果. 在每个异常检测算法中, 每个数据集中的数据点的异常值分数都是自动生成, 根据异常值分数按从大到小的顺序对数据点进行排序, 取前 q 个数据点作为预测的异常值, q 表示为真实异常值的个数. 根据生成的异常值分数、预测的标签和真实的标签, 计算每个异常检测算法在不同数据集下的平均精度 AP 、$\max F_1$和 AUC, 评估它们的检测异常的性能.

表 2 记录了 SOPD 算法与其他 6 种异常检测算法的平均精度 AP, 其中粗体表示实验结果中每个数据集所对应的最高的 AP 值. 从表中可以看出, SOPD 算法仅仅在 SHS 数据集上低于 SOD 算法的平均精度值, 其他数据集上, SOPD 算法的平均精度都高于另外 6 种异常检测算法. 尤其是在 Ionosphere 数据集上, SOPD 算法的平均精度高达 0.92. 除此之外, Ovarian 数据集上, OCSVM 算法的平均精度和 PCA 算法、SOD 算法一样都是 0.70, 为 6 种异常检测算法中最高的, 这是因为 OCSVM 算法可以捕获到不同数据集的分布情况, 尤其是在维度较高的大样本数据集上, 事先并不知道数据的分布状况, 此种情况下 OCSVM算法有较强的检测能力, 而 SOPD 算法的 AP 值为 0.73, 比 OCSVM 算法更高, 因此, SOPD 算法在未知数据分布的情况下, 对高维度数据集的异常检测性能也较好.

为了再次验证 SOPD 算法的优异性, 通过使用$\max F_1$进行性能度量, 具体实验比较结果如表 3 所示. 表 3 的比较结果显示, 在 Ovarian 数据集、SHS 数据集和 TAE 数据集上, SOPD 算法的$\max F_1$和另外 6 种异

常检测算法中的最高值一样, 除此之外, SOPD 算法在其他数据集上的max F_1都高于那些异常检测算法. 比如, Leaf 数据集中, SOPD 算法的max F_1为 0.33, 而其他检测算法的最高的max F_1仅为 0.26, SOPD 算法的结果仍较高. MUSK 数据集上, PCA 算法的max F_1为 0.75, 是 6 种异常检测算法中最高的, 但 SOPD 算法的max F_1仍比 PCA 算法高. 因此, SOPD 算法在所有数据集上都有较高的max F_1值, 尤其是在 Wholesale 数据集、ForestTypes 数据集和 Ionosphere 数据集上, SOPD 算法的性能明显优于其他异常检测算法. SOPD 算法主要是同时考虑了数据的局部性和全局性结构信息.

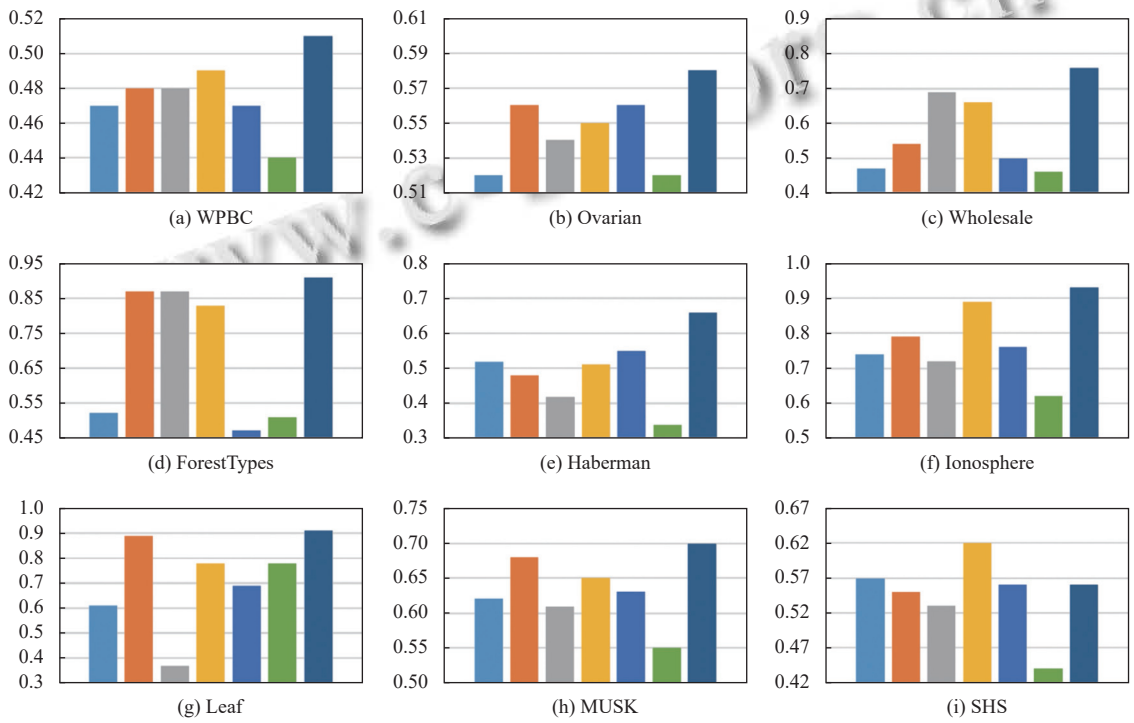
表 2 异常检测算法的平均精度

数据集	SOS	PCA	LMDD	SOD	OCSVM	LOCI	SOPD
WPBC	0.23	0.23	0.25	0.24	0.23	0.21	**0.28**
Ovarian	0.67	0.70	0.64	0.70	0.70	0.65	**0.73**
Wholesale	0.31	0.41	0.48	0.43	0.32	0.32	**0.54**
ForestTypes	0.20	0.71	0.66	0.59	0.14	0.34	**0.73**
Haberman	0.30	0.29	0.25	0.31	0.35	0.22	**0.44**
Ionosphere	0.74	0.75	0.64	0.89	0.75	0.63	**0.92**
Leaf	0.11	0.14	0.03	0.14	0.05	0.09	**0.20**
MUSK	0.68	0.73	0.68	0.70	0.66	0.58	**0.75**
SHS	0.53	0.52	0.51	**0.58**	0.53	0.45	0.57
TAE	0.35	0.25	0.37	0.32	0.31	0.31	**0.39**

实验过程中通过使用 AUC 准则衡量算法的性能, 根据 AUC 的评价度量, 再次证明了 SOPD 算法的优越性. 图 4 描绘了不同异常检测算法的 AUC, 图 4 一共有 10 个柱形图, 分别表示 SOPD 算法和 6 种异常检测算法在 10 个数据集上的 AUC 值的比较结果, 其中每个柱形图中的不同颜色的长条代表不同的异常检测算法, SOPD 算法由最后一个长条所表示, 通过图 4 可以清晰地看出同一个数据集上不同异常检测算法的性能差异. SOPD 算法的 AUC 只在 SHS 数据集上较低于 SOS 算法和 SOD 算法, 即使在 TAE 数据集上, SOPD 算法的 AUC 也与 6 种异常检测算法中的最高值一样. 除此之外, 在其他数据集上, SOPD 算法的性能明显比另外 6 种异常检测算法较优越, 特别是在 ForestTypes 数据集、Ionosphere 数据集、Leaf 数据集上, SOPD 算法的 AUC 分别为 0.91、0.93 和 0.91, 表现出了良好的性能. 因此, 由图 4 可以看出, SOPD 算法的准确性总体上较好.

表 3 异常检测算法的 maxF_1

数据集	SOS	PCA	LMDD	SOD	OCSVM	LOCI	SOPD
WPBC	0.39	0.41	0.38	0.40	0.40	0.39	**0.43**
Ovarian	**0.78**	**0.78**	**0.78**	**0.78**	**0.78**	**0.78**	**0.78**
Wholesale	0.49	0.49	0.59	0.54	0.49	0.49	**0.67**
ForestTypes	0.28	0.67	0.71	0.59	0.25	0.51	**0.72**
Haberman	0.43	0.42	0.42	0.42	0.43	0.42	**0.50**
Ionosphere	0.65	0.65	0.62	0.82	0.66	0.58	**0.85**
Leaf	0.19	0.26	0.06	0.27	0.14	0.17	**0.33**
MUSK	0.72	0.75	0.72	0.74	0.74	0.72	**0.76**
SHS	0.63	0.63	0.63	0.63	**0.64**	0.63	**0.64**
TAE	**0.50**	0.49	0.49	**0.50**	0.49	0.49	**0.50**



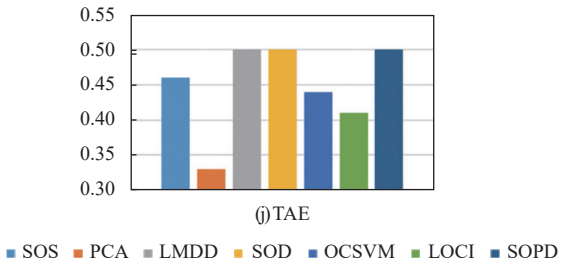


图 4 各异常检测算法在 10 个数据集上的 AUC (其中, 横坐标表示各算法, 纵坐标表示 AUC)

4 结束语

本文提出了一种基于二阶近邻的异常检测算法, 利用二阶近邻将数据全局结构考虑进去, 并利用数据点与所属簇中心之间的关系捕获异常情况. 总体来说, 该算法主要包括聚类和异常检测两个阶段. 在聚类过程中利用一阶近似和二阶近邻综合考虑局部性和全局性信息, 对数据的初始相似性矩阵进行重新构造, 提高聚类的性能. 之后, 在聚类结果的基础上, 同时考虑每个簇中的数据点与该簇中心之间的距离和数据点本身的密度, 得到异常值分数, 根据异常值分数判断数据点的异常情况. 我们在 10 个真实数据集上进行了比较实验, 其结果表明, SOPD 算法与经典的异常检测算法相比具有一定的优越性. 下一步的研究方向是减少算法的时间复杂度, 并进一步增强算法对异常值的敏感度.

参考文献

1 Saxena S, Rajpoot DS. Density-based approach for outlier detection and removal. Proceedings of 2018 International Conference on Signal Processing and Communication. Singapore: Springer, 2019. 281–291. [doi: 10.1007/978-981-13-2553-3_27]

2 Liu HW, Li EH, Liu XW, *et al*. Anomaly detection with kernel preserving embedding. ACM Transactions on Knowledge Discovery from Data, 2021, 15(5): 91. [doi: 10.1145/3447684]

3 Liu TF, Gao H, Wu JJ. Review of outlier detection algorithms based on grain storage temperature data. Proceedings of 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA). Dalian: IEEE, 2020. 1045–1048. [doi: 10.1109/ICAICA50127.2020.9182588]

4 Wahid A, Rao ACS. An outlier detection algorithm based on KNN-kernel density estimation. Proceedings of 2020 International Joint Conference on Neural Networks (IJCNN). Glasgow: IEEE, 2020. 1–8. [doi: 10.1109/IJCNN48605.2020.9207033]

5 Siegel AF. Statistics and Data Analysis: An Introduction. New York: John Wiley & Sons, 1988.

6 Laurikkala J, Juhola M, Kentala E. Informal identification of outliers in medical data. Proceedings of 5th International Workshop on Intelligent Data Analysis in Medicine and Pharmacology. Berlin: ECCAI, 2000. 20–24.

7 Angiulli F, Fassetti F. DOLPHIN: An efficient algorithm for mining distance-based outliers in very large datasets. ACM Transactions on Knowledge Discovery from Data, 2009, 3(1): 4. [doi: 10.1145/1497577.1497581]

8 Zhang K, Hutter M, Jin HD. A new local distance-based outlier detection approach for scattered real-world data. Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining. Bangkok: Springer, 2009. 813–822. [doi: 10.1007/978-3-642-01307-2_84]

9 Angiulli F, Basta S, Lodi S, *et al*. Reducing distance computations for distance-based outliers. Expert Systems with Applications, 2020, 147: 113215. [doi: 10.1016/j.eswa.2020.113215]

10 Breunig MM, Kriegel HP, Ng RT, *et al*. LOF: Identifying density-based local outliers. ACM SIGMOD Record, 2000, 29(2): 93–104. [doi: 10.1145/335191.335388]

11 Jin W, Tung AKH, Han JW, *et al*. Ranking outliers using symmetric neighborhood relationship. Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining. Singapore: Springer, 2006. 577–593. [doi: 10.1007/11731139_68]

12 Riahi-Madvar M, Azirani AA, Nasersharif B, *et al*. A new density-based subspace selection method using mutual information for high dimensional outlier detection. Knowledge-based Systems, 2021, 216: 106733. [doi: 10.1016/j.knosys.2020.106733]

13 Pu G, Wang LJ, Shen J, *et al*. A hybrid unsupervised clustering-based anomaly detection method. Tsinghua Science and Technology, 2021, 26(2): 146–153. [doi: 10.265

99/TST.2019.9010051]

14 Ester M, Kriegel HP, Sander J, *et al*. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining. Portland: AAAI Press, 1996. 226–231.

15 Huang JL, Zhu QS, Yang LJ, *et al*. A novel outlier cluster detection algorithm without top-n parameter. Knowledge-Based Systems, 2017, 121: 32–40. [doi: 10.1016/j.knosys.2017.01.013]

16 Nozad N, Ahmad S, Haeri A, *et al*. SDCOR: Scalable density-based clustering for local outlier detection in massive-scale datasets. Knowledge-based Systems, 2021, 228: 107256. [doi: 10.1016/j.knosys.2021.107256]

17 Gao JH, Ji WX, Zhang LL, *et al*. Cube-based incremental outlier detection for streaming computing. Information Sciences, 2020, 517: 361–376. [doi: 10.1016/j.ins.2019.12.060]

18 Bansal M, Sharma D. A novel multi-view clustering approach via proximity-based factorization targeting structural maintenance and sparsity challenges for text and image categorization. Information Processing & Management, 2021, 58(4): 102546. [doi: 10.1016/j.ipm.2021.102546]

19 Frey BJ, Dueck D. Clustering by passing messages between data points. Science, 2007, 315(5814): 972–976. [doi: 10.1126/science.1136800]

20 Janssens JHM, Huszár F, Postma EO, *et al*. Stochastic outlier selection. Technical Report. Tilburg: Tilburg University, 2012.

21 Shyu ML, Chen SC, Sarinnapakorn K, *et al*. A novel anomaly detection scheme based on principal component classifier. Proceedings of IEEE Foundations and New Directions of Data Mining Workshop, in Conjunction with the 3rd IEEE International Conference on Data Mining (ICDM). IEEE, 2003. 172–179.

22 Arning A, Agrawal R, Raghavan P. A linear method for deviation detection in large databases. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining. Portland: AAAI Press, 1996. 164–169.

23 Kriegel HP, Kröger P, Schubert E, *et al*. Outlier detection in axis-parallel subspaces of high dimensional data. Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining. Bangkok: Springer, 2009. 831–838. [doi: 10.1007/978-3-642-01307-2_86]

24 Schölkopf B, Platt JC, Shawe-Taylor J, *et al*. Estimating the support of a high-dimensional distribution. Neural Computation, 2001, 13(7): 1443–1471. [doi: 10.1162/089976601750264965]

25 Papadimitriou S, Kitagawa H, Gibbons PB, *et al*. LOCI: Fast outlier detection using the local correlation integral. Proceedings of the 19th International Conference on Data Engineering. Bangalore: IEEE, 2003. 315–326. [doi: 10.1109/icde.2003.1260802]

26 Zhao Y, Nasrullah Z, Li Z. PyOD: A Python toolbox for scalable outlier detection. Journal of Machine Learning Research (JMLR), 2019, 20(96): 1–7.

(校对责编: 牛欣悦)