**基于密度、统计等方法的异常检测及其优化**

**江南大学人工智能与计算机学院**



**姓 名： 金家耀**

**专 业： 人工智能**

**学 号： 1193210320**

**摘 要**

异常检测在医学、网络安全、工业瑕疵检测等多个领域具有重要意义。在传统异常检测领域，学者们早已进行了基于密度、统计、距离、聚类、以及集成学习等方向的深入研究，取得了显著成果。然而，随着近现代应用产生的大规模数据，维度诅咒的问题使得传统异常检测方法，特别是基于距离的方法，在高维度数据上难以取得令人满意的效果。近年来，学者们开始致力于研究大规模数据集的异常检测方法，包括挖掘正常点与异常点差异较大的子空间、使用多模型集成算法并行计算等方面的探索。由于TensorFlow、PyTorch等深度学习框架在矩阵运算上的广泛应用，如何利用GPU加速异常检测领域也成为研究热点。

本文深入研究了基于不同方法的结构化数据异常检测，对各算法的理论推理和流程进行详细论述。同时，通过Pyod在生成数据集和不同实际数据集上的实验，对比了不同算法的AUC指标、P@N分数。同时对比了使用GPU和CPU的算法时间效率。

**关键词：**异常检测；各类传统方法；并行化；GPU加速；集成算法

**ABSTRACT**

Anomaly detection holds significant importance in various domains such as medicine, cybersecurity, and industrial defect detection. In the traditional anomaly detection field, scholars have conducted in-depth research in directions such as density-based, statistical, distance-based, clustering, and ensemble learning methods, achieving notable success. However, with the emergence of large-scale data in modern applications, the curse of dimensionality hinders the effectiveness of traditional anomaly detection methods, especially distance-based approaches, in high-dimensional data. In recent years, researchers have focused on exploring anomaly detection methods for large-scale datasets, including subspaces that capture significant differences between normal and anomalous points, as well as parallel computation using multi-model ensemble algorithms. The utilization of GPU for accelerating anomaly detection in the era of large-scale data has become a research hotspot, with frameworks like TensorFlow and PyTorch being widely applied for matrix operations.

This paper conducts an in-depth study of structured data anomaly detection based on different methods, and discusses in detail the theoretical reasoning and processes of each algorithm. At the same time, through Pyod's experiments on generated data sets and different actual data sets, the AUC indicators and P@N scores of different algorithms were compared. At the same time, the time efficiency of the algorithm using GPU and CPU was compared.

**Keyword**：Anomaly detection; Various traditional methods; Parallelization; GPU acceleration; Ensemble algorithms

# 目 录

[第1章 引言 1](#_Toc15360)

[1.1 异常检测技术的应用 1](#_Toc6316)

[1.2 异常检测技术国内外现状（传统方法） 1](#_Toc15198)

[第2章 异常检测的定义及相关问题 3](#_Toc25044)

[2.1 异常检测的任务定义与难点 3](#_Toc22717)

[2.2 异常的分类 3](#_Toc13140)

[第3章 异常检测方法分类 5](#_Toc11015)

[3.1 基于密度的方法 5](#_Toc7479)

[3.1.1 Local Outlier Factor (LOF) 5](#_Toc11199)

[3.2 基于统计的方法 6](#_Toc15883)

[3.2.1 高斯混合模型 (GMM) 6](#_Toc17903)

[3.2.2 基于GMM的异常检测 9](#_Toc18592)

[3.3 基于距离的方法 9](#_Toc11617)

[3.3.1 基于索引的方法（Indexed-based Method） 9](#_Toc17616)

[3.3.2 基于嵌套循环的方法（Nested-loop Method） 10](#_Toc18656)

[3.3.3 基于单元格的方法（Cell-based Method） 10](#_Toc19265)

[3.3.4 Local Distance-based Outlier Factor (LDOF) 11](#_Toc3932)

[3.4 基于聚类的方法 12](#_Toc15492)

[3.4.1 基于DBSCAN的异常检测方法 12](#_Toc30481)

[3.5 基于集成学习的方法 13](#_Toc13833)

[3.5.1 基于Feature Bagging的方法 13](#_Toc20275)

[3.5.2 基于孤立森林的方法 14](#_Toc23476)

[第4章 实验部分 17](#_Toc24486)

[4.1 数据集 17](#_Toc795)

[4.1.1 MNIST 17](#_Toc18382)

[4.1.2 Arrhythmia 17](#_Toc30132)

[4.1.3 Breastw 17](#_Toc7287)

[4.1.4 Cardiotocogrpahy 17](#_Toc27006)

[4.1.5 HTTP 17](#_Toc1636)

[4.2 实验指标 17](#_Toc3053)

[4.2.1 AUC指标 17](#_Toc28178)

[4.2.2 P@N指标 18](#_Toc6422)

[4.3 PyOD 18](#_Toc26271)

[4.4 实验对比 19](#_Toc22787)

[4.4.1 生成数据集下的实验 19](#_Toc29395)

[4.4.2 真实数据集下的实验 20](#_Toc14843)

[4.5 SUOD 20](#_Toc23787)

[4.6 Pytod 21](#_Toc10323)

[第5章 结论与展望 22](#_Toc4392)

[参考文献 23](#_Toc32368)

[致 谢 25](#_Toc25714)

# 引言

异常检测是数据分析中的一项关键任务[1]，其目标是识别数据集中与其余数据明显不同的、异常或离群的数据点。这些异常点通常表示一些不寻常或罕见的事件，与正常模式相比存在显著的差异。

* 1. 异常检测技术的应用

异常检测在各个领域都有广泛的应用，从工业制造到金融领域，都可以通过识别异常来提高系统的可靠性、安全性和效率。以下是一些异常检测的主要应用场景：

(1) 工业制造：监测生产线中的设备故障、材料缺陷或生产异常，提高产品质量和生产效率；

(2) 金融领域[2][3]：欺诈检测、检测市场异常、交易异常或投资组合中的异常波动；

(3) 社交媒体分析：监测社交媒体上的异常活动，如虚假账号、恶意评论或舆情异常；

(4) 网络安全[4]：检测网络中的异常流量、异常登录或其他潜在的网络攻击；

(5) 健康检测[5]：在医疗领域中，监测患者的生理参数，识别异常病症或患者状况的突变。

这些应用场景表明异常检测在不同行业中都能发挥关键作用，帮助提高效率、降低风险，并改善决策过程。

* 1. 异常检测技术国内外现状（传统方法）

异常检测技术想要努力解决的问题是在如何在我们设想的正常模式或场景下挖掘那些远离正常模式的数据点，但这个场景往往是比较复杂的[1]，例如不同场景下的异常点不同的定义、异常点与正常点模糊的边界、如何区分真实的异常点与人为造成的异常点、如何在需求量大的场景下高效使用技术等。

虽然异常检测领域存在一些相关问题，但目前国内外研究者们已经提出了一些异常检测的方法，如基于密度[6][7]的方法、基于统计[8]的方法、基于距离[9][10]的方法、基于聚类[11]的方法、基于集成学习[12]的方法等。

在处理实际应用中产生的高维特征数据时，异常值检测面临一系列挑战，与简单的、低维度的特征数据相比存在明显差异。当前应用软件生成的数据往往具有复杂的高维结构，而采用传统的异常值检测方法可能导致一些不利影响。这种情况下，存在两个主要问题需要解决。首先，高维特征数据的处理通常需要更多的计算资源。由于高维数据空间的复杂性，传统的异常值检测方法可能面临计算复杂度的挑战。处理高维数据需要更多的计算资源，这可能对实时性能和计算效率产生不利影响。其次，维度诅咒[13]的影响使得即使在低维空间中能够有效检测异常点，但在高维空间中，由于欧氏距离计算导致数据点之间的距离很接近。这种情况下，传统方法很难准确检测到异常点，因为它们在高维空间中表现得相对均匀，使异常点的特征难以显著突出。

异常检测的意义在不同领域和应用场景中呈现差异，这差异性主要源于数据的本质和领域特定的背景知识。以医学影像和相机拍摄的照片为例，异常的定义和解释在这两个场景中存在显著的不同。

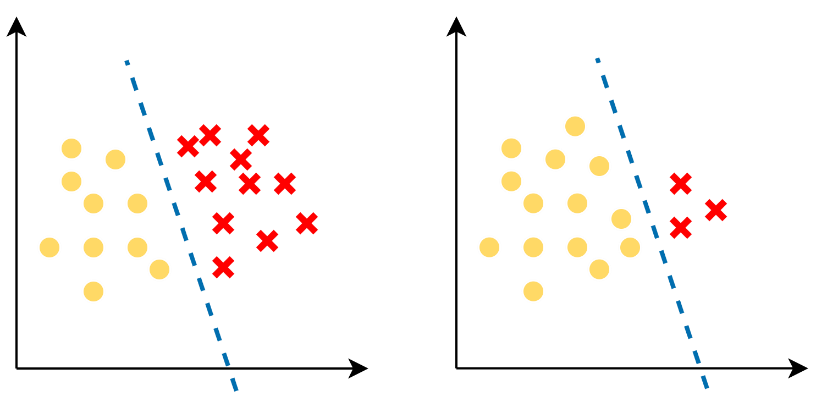
在医学影像领域，X光片中的瑕疵或噪声可能被视为异常。这是因为医学背景知识指导了对人体健康的高度敏感性，对于图像中的任何不寻常现象都可能与患者的症状或疾病相关。相比之下，在相机拍摄的照片中，部分瑕疵和噪声通常被认为是正常现象。相机技术的不确定性和环境变化可能导致图像中的一些不规则性，但这并不一定表示存在问题或异常。

这种领域特定性使得异常检测技术难以通用化。在不同领域，异常的定义和上下文都发生变化，因此很难找到一个适用于所有场景的通用异常检测方法。这进一步强调了异常检测的相对主观性和依赖领域背景知识的特征。

# 异常检测的定义及相关问题

1. 1. 异常检测的任务定义与难点

假设有一个包含正常样本和异常样本的数据集 ，其中 表示该数据集大小，异常检测任务就是要把存在异常的数据点挖掘出来。对于存在标签 的数据集来说，0表示正常点，1表示异常点，我们使用监督学习方法；而对于不存在标签的数据集来说，我们可以使用如聚类等无监督学习的方法。



（a）二分类问题 （b）异常检测问题

图2-1 二分类问题与异常检测问题对比图

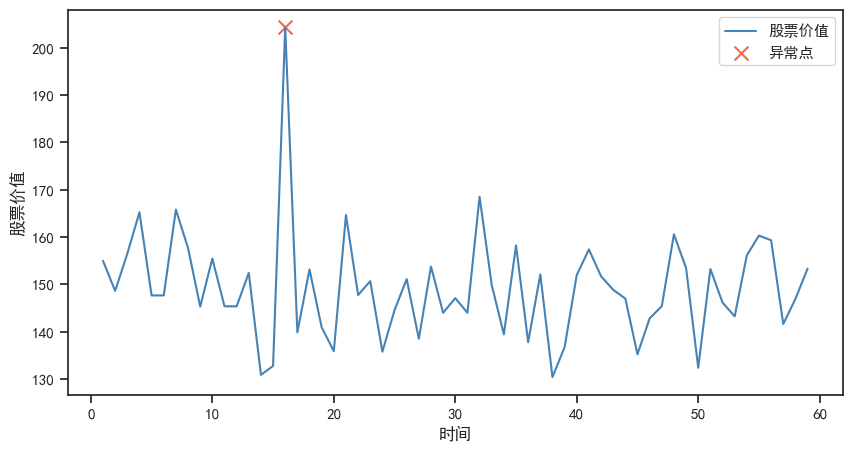
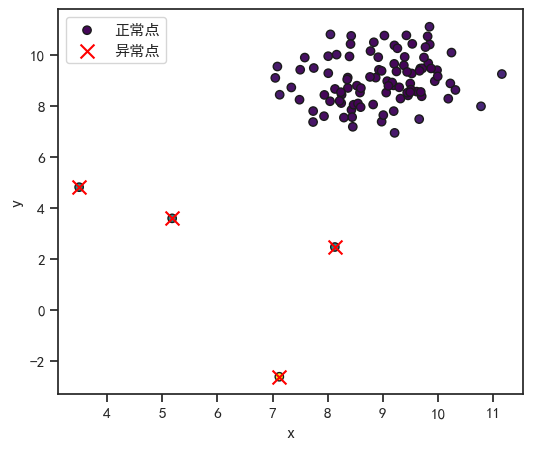
如上述定义，异常检测类似于一个二分类问题，只需要把数据集中的正常点与异常点区分开即可。但二分类问题远远没有异常检测任务复杂，异常检测可能会遇到如下两个问题：

**(1) 类别不均衡：**如图2-1（b）所示，黄色的圆表示正常点，红色的叉表示异常点。在很多情况下，我们无法收集足够量的负样本来训练一个模型，换句话说，数据集中的负样本数量可能会远小于正样本数量。使用这样不均衡的数据集进行二分类模型训练，该二分类模型的性能往往不佳，模型可能会存在过拟合的情况；

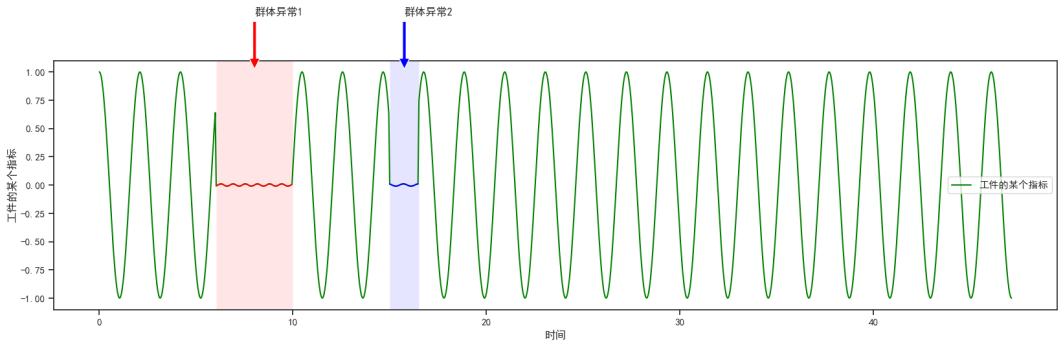
**(2) 未知性与不规则性：**对于异常检测任务而言，模型需要判断一个数据点是否符合我们对训练集中的正常点“正常”的定义。但由于事物的多样性，异常点的情况比正常点复杂得多，比如在医学健康领域，假设我们定义人的正常情况是正常的，那么异常点就是感冒、发烧、咳嗽、呕吐等等，这样的异常点我们可以列举出很多。但事实是，我们不可能在数据集中穷举出所有情况的异常点。当模型遇到没有出现过的异常点时，模型在检测异常时会显得不是那么“得心应手”。

* 1. 异常的分类

异常根据数据集属性分类，可以分为统计型数据（static data）、序列型数据（sequential data）和空间型数据（spatial data）。



（a）点异常示例图 （b）条件异常示例图



（c）群体异常示例图

图2-2 异常检测按类别分类的三种示例图

如图2-2所示，根据异常点的类别分类，可以分为点异常、条件异常和群体异常：

**(1) 点异常（Point Anomalies）:**指在数据集中存在个别的个体实例，其行为或特征与大多数个体实例明显不同，在网络安全领域，如果大多数用户的登录模式是典型的，而有一个用户在半夜频繁登录，这个用户的登录行为可能被视为点异常。

**(2) 条件异常（Conditional Anomalies）:**又称上下文异常:发生在特定的情境或上下文中，表明个体实例在这种情况下被认为是异常，而在其他情况下可能是正常的。在工业生产中，在特定的温度和湿度条件下某个设备的性能突然下降，被视为条件异常。

**(3) 群体异常（Group Anomalies）:**指的是在整体集合中，某些个体实例的组合或集合表现出与整体集合明显不同的行为或特征。在社交网络中，虚假账号可能在单独考察时看起来正常，但如果检测到一组虚假账号形成的集合，这个集合可能被视为群体异常。

# 异常检测方法分类

1. 1. 基于密度的方法

使用基于密度的方法来进行异常检测有着很长一段历史，这种方法认为异常点一般会出现在低密度的区域，而正常点一般会出现在高密度的区域。如图2-2（a）所示，黑色正常点的邻居很多，而且各自之间的距离都比较相近，而黄色异常点周围邻居很少，有些甚至没有邻居。这些方法通过使用某个数据点的局部与这个数据点局部邻居的局部密度进行比较，从而判断其是否为异常点。

* + 1. Local Outlier Factor (LOF)

Local Outlier Factor（LOF）[6]是一种基于密度的异常检测算法，通过计算数据点相对于其邻居的局部密度来确定异常点。LOF考虑了每个数据点与其邻居之间的距离，并通过比较局部密度来识别相对孤立的点。

LOF算法认为，异常点不是非黑即白的，每个点都存在一个异常值，把是否为异常点量化，使得模型能够更好的挖掘异常点。

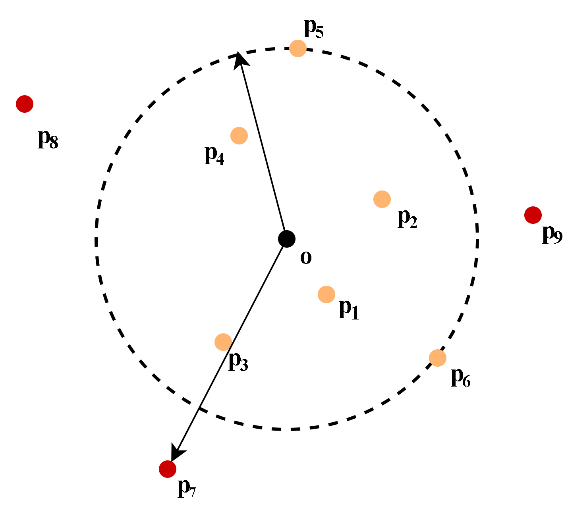


图3-1 点P相对于点O的可达距离（k=5）

**（1）点P的K近邻距离（）**

距离点P第K远的点的距离，但不包括距离点P相同距离的点；

**（2）点P的K近邻邻居（）**

点P的K近邻邻居包括距离点P的距离小于等于点P的K近邻距离的点，点P的K近邻邻居个数为：

在下面的介绍中，使用 来代表 ；

**（3）相对于点O的可达距离**

设为自然数，那么点P相对于点O的可达距离定义如下：

如图3-1所示，为当k=5时点P相对于点O的可达距离，其中 的K近邻邻居（k=5）有 。例如，对于 而言，由于它是 的K近邻邻居，故 ；而对于 而言，由于它不是K近邻邻居，故

**（4）点P的本地可达密度（local reachability density）**

由于上述定义的公式中的k为任意自然数，但为了定义异常，引入了在基于密度聚类中的参数 。该参数的意义是，若某个数据点周围的数据点个数大于等于这个阈值，就把这个点设置为核心点（Core Point），继续迭代聚类。

则点P的本地可达密度定义如下：

值得注意的是， 表示的是点O到点P的距离。言外之意，该公式表达的是P点相对于P的 近邻邻域内的邻居的可达距离的均值的倒数，衡量的是P对于P的邻居之间的疏远程度。

**（5）异常因子（local outlier factor）**

根据该方法的思想，异常点往往会出现密度相对较低的区域。以上定义了本地可达密度，我们只需要比较某个点的本地可达密度与该点 近邻的邻居的本地可达密度。点P的异常因子定义如下：

观察上述公式，当 数值越大，表明点P的 值越大，点P越有可能是异常点。

* 1. 基于统计的方法
     1. 高斯混合模型 (GMM)

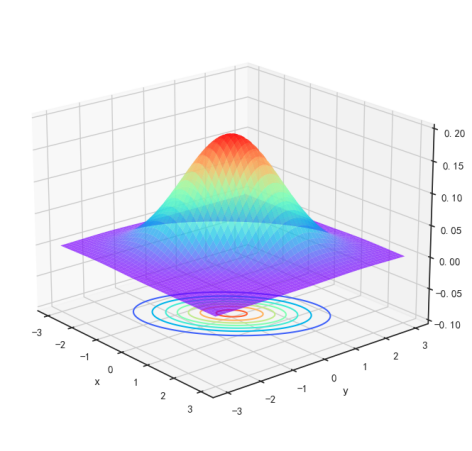
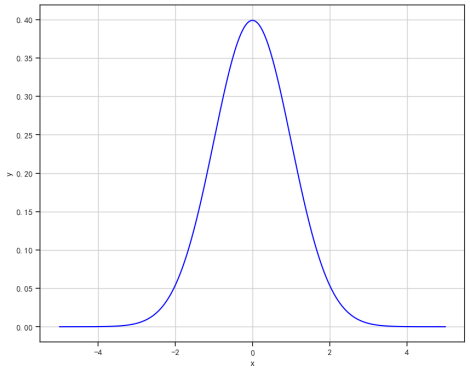
**（1）高斯分布**

如图3-2所示，是一维以及二维高斯分布概率密度函数图像。则一维高斯分布概率密度函数为：

其中 表示均值（期望）， 表示标准差。

而当数据特征具有d维时，如图3-2（b）中的2维特征，则d维数据的高斯分布概率密度函数维：

其中 表示均值（期望）， 表示标数据的协方差， 表示数据维度。



（a）一维高斯分布概率密度函数 （b）二维高斯分布概率密度函数

图3-2 高斯分布示例图

混合模型（Mixture Model）表示总体分布中包含多个子分布的概率模型，该总体分布并不是单一的某种概率分布，而是以一定的概率随机从多个子分布抽取若干个样本点，所有样本点符合一个总体分布。

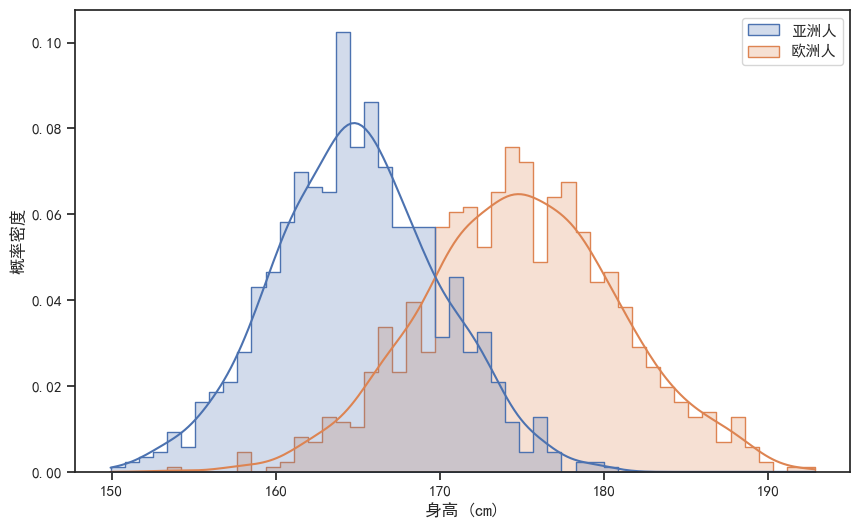


图3-3 亚洲人和欧洲人身高混合数据概率密度模拟图

**（2）高斯混合模型**

众所周知，真实世界中的大部分连续数据都符合高斯分布，例如，身高符合高斯分布，越靠近均值，数量越大。但我们还知道欧洲人的身高普遍比亚洲人高，这个“普遍”的意思时欧洲人的身高均值比亚洲人大。如图3-3所示，假设我们得到一个包含亚洲人和欧洲人身高的数据集，我们需要估计亚洲人和欧洲人各自身高的均值、方差与各自在数据中的占比。

上述问题其实就是混合高斯模型求解问题，下面我将介绍普遍的混合高斯模型求解方法。

假设存在数据集 ， 表示数据集的大小，我们将用 个子高斯模型来估计这个数据集。那么混合高斯模型的概率分布为：

其中， 是观测点属于第 个子高斯分布的概率，且满足 , 。 是第 个子高斯分布密度函数， 为第 个高斯分布的均值和方差或协方差，故 的大小是 。

对于单高斯模型来说，我们可以使用极大似然估计[14]来求解模型参数：

公式（3 - 8）可以直接使用求导来求得最优值。

而对于高斯混合模型而言，最大似然函数为：

但由于累乘会使数值变得很小，不利于优化，所以采用对数极大似然法（Log-Likelihood）：

**（3）Expectation Maximization算法（EM）**

观察公式（3 - 9），之所以能够直接求导求出最优值，是因为该函数是一个凸优化问题。但公式（3 - 10）并不是凸优化问题，我们不能同时求得函数的所有参数。故只能固定一些参数而更新令模型收敛，以下是具体步骤：

* ，表示第 个数据点属于第个高斯模型的概率， 表示优化步数；
* ，计算第 个高斯模型的均值；
* ，计算第 个高斯模型的协方差，使用的是更新之后的均值 ；
* ，计算当前 个模型的概率；
* 重复迭代，直至 ， 为一个很小的正数。
  + 1. 基于GMM的异常检测

而基于GMM的异常检测[8]中的高斯混合模型是一种特殊的模型。在GMM中，子高斯分布的个数 ，是根据类别而定。但对于异常检测来说，不能简单的将 设置成2，即异常分布和正常分布。这样显然是不合理的，正常分布不一定是一个严格的单高斯分布，异常分布也是同理。

而基于GMM的异常检测，是把 直接设置成样本大小 。每一个数据点都是一个高斯分布。则公式（3 - 11）就变成了如下形式：

其中，， 表示第 个高斯子模型的概率，即混合比例。基于GMM的异常检测方法步骤如下所示：

计算每个数据点属于每一个高斯子模型的概率：

更新混合比例：

其中，， 表示 和 之间的距离。

定义 ，该变量可以看作是 和 之间的关联强度。最后将公式（3 - 14）简化为：

最终，异常因子定义为：

观察公式（3 - 15），从某一个数据点 看， 表示其他数据点对其的影响程度，当影响程度越低， 越小， 越大，该数据点越可能是异常点。

* 1. 基于距离的方法
     1. 基于索引的方法（Indexed-based Method）

定义数据点O的D近邻邻居（）为 ，其中 为计算点O和点Q之间的距离的函数。对于每一个数据点，使用索引遍历除该点之外的所有点，若遍历到的点与该点的距离小于D，该点设置为O点的D近邻邻居。若D近邻邻居个数大于 个，则认为该点为正常点，反之，该点为异常点。

为了提高算法的效率，搜索D近邻邻居时，会使用R树、kd树和X树，这些算法的时间复杂度下限为 ，但这些算法的最差情况依然是 。

* + 1. 基于嵌套循环的方法（Nested-loop Method）

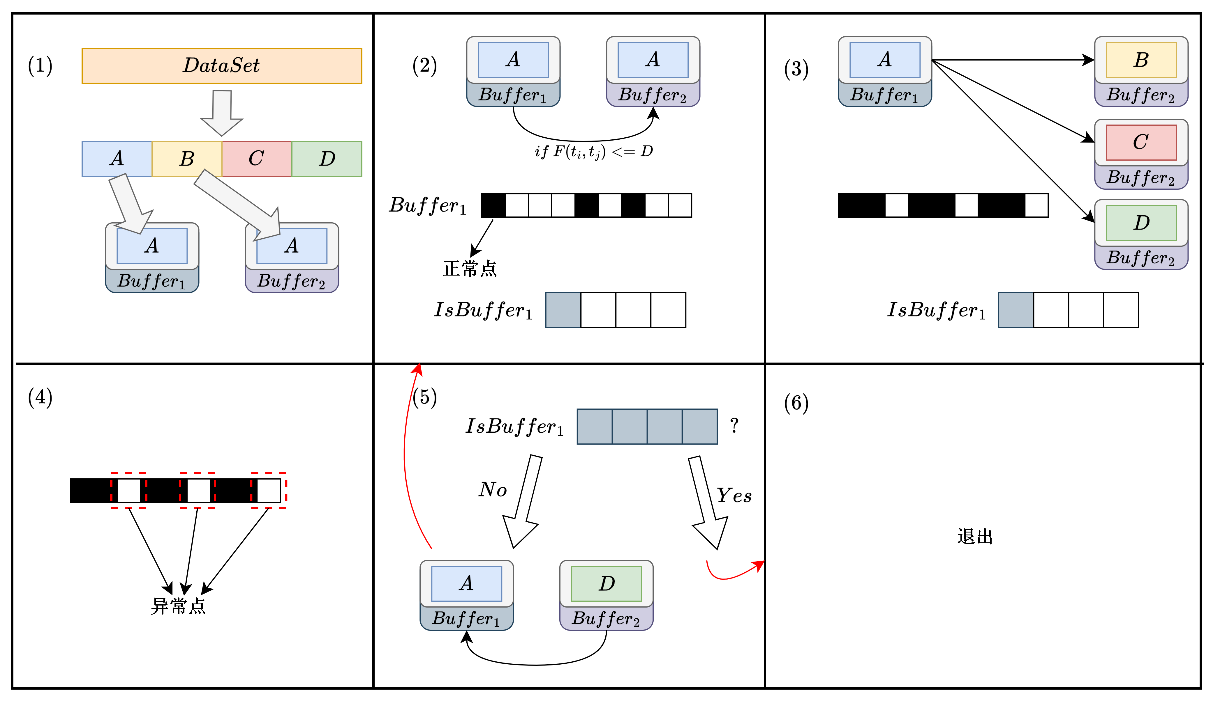


图3-4 基于嵌套循环的距离异常检测算法流程图

为了减少上述基于索引的方法需要巨大的计算量与时间消耗，该方法如下：

(1) 将数据集分为4个子数据集使用2个缓存器来储存2个子数据集。

(2) 对于第一个缓存器中的每一个数据点 ，循环第二个缓存器中数据点 ，并且维护一个D近邻邻居数组。若某个数据点D近邻邻居个数大于 ，则标记该数据点为正常值。

(3) 循环完毕后，若第一个缓存器中仍存在需要比较的数据点，则使用其他子数据集填充，并且将数据点是否已经作为过第一个缓存器中的数据点记录下来。对于第一个缓存器中未被标记的数据点 ，循环第二个缓存器中的数据点 ，继续按照步骤 (3) 中的方法维护一个D近邻邻居数组，并作正常点标记；

(4) 若第二个缓存器中的数据点在之前任何时候都作为过第一个缓存器中的数据，则停止。否则，交换两个缓存器中的数据点，并继续步骤 (2)。

* + 1. 基于单元格的方法（Cell-based Method）

假设数据点的维度为 ，用以判定异常点的阈值为 。数据空间的每一维度都被分为宽度为 的单元格。用图3-4所示的放个图举例，对于每一个单元格A，若该单元格以及 单元格中包含了超过 个数据点，那么这些数据点都被认为是正常点；若该单元格、 单元格以及 单元格内包含少于 个数据点，则单元格A中的所有点都是异常点。

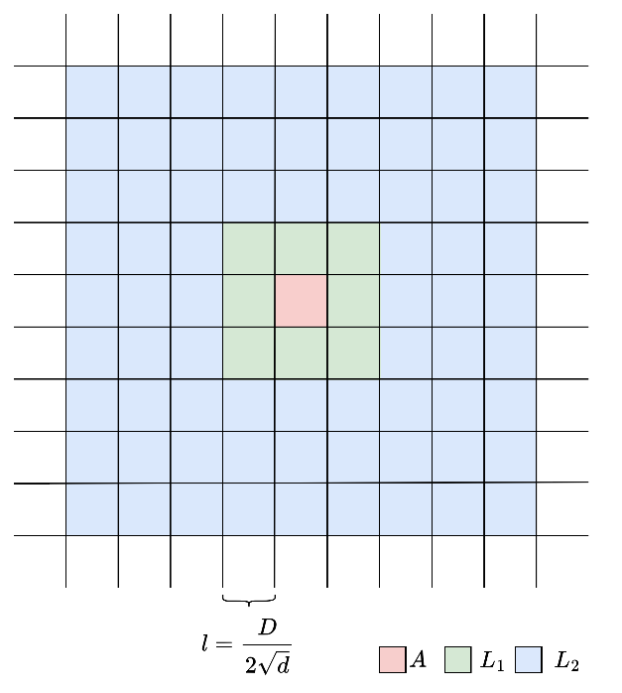


图3-5 基于单元格的距离异常检测二维情况下的方格图

对比上述基于索引的方法，该方法基于分区能够更加快速地寻找异常点，但由于维度诅咒，该方法不适合较高维度的数据集。

* + 1. Local Distance-based Outlier Factor (LDOF)

LDOF解决的是传统方法把一些较小的簇误判定为异常点，尤其是选取的 比较大的时候。这种方法可以考虑到体积较小的正常簇，公式如下：

(1) 计算 点的k近邻邻居的平均距离：

(2) 计算k近邻邻居之间的平均距离：

(3) 计算 的局部距离异常因子（）：

当 时，该数据点被数据点云所包围；当 时，该数据点远离数据点云。

* 1. 基于聚类的方法

聚类的目的是把数据集中一些相似的数据点聚集在一块，这些点构成的集合被称为“簇”。而基于聚类方法的异常检测比较特殊，我们并不是通过聚类方法来寻找这些异常点，而是根据这些方法来找到没有被聚类的数据点。

基于聚类方法的异常检测可以很多类，以下只举出最常见的三类：

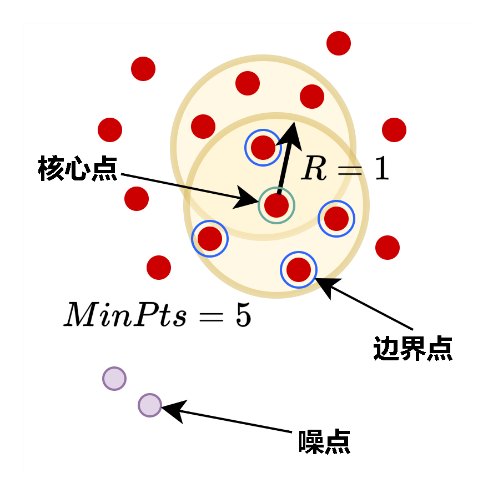
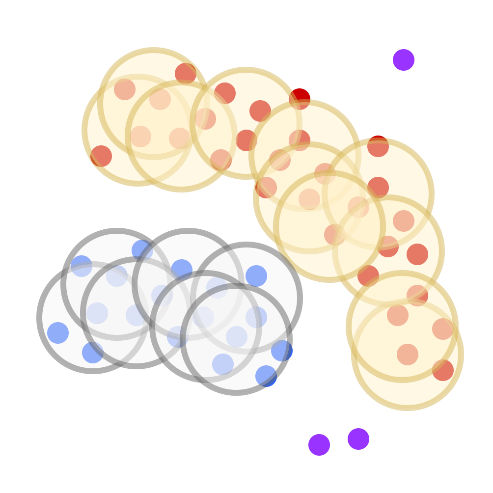
(1) 基于分群的聚类算法：PAM[15]、CLARANS[16]、K-Means[17]等；

(2) 基于密度的聚类算法：DBSCAN[10]、DENCLUE[18]等；

(3) 层次聚类算法：MST[19]、CURE[20]、CHAMELEON[21]等。

* + 1. 基于DBSCAN的异常检测方法

DBSCAN（Density-Based Spatial Clustering of Applications with Noise）是一种密度聚类算法，它能够识别空间中的高密度区域，从而形成簇，并且能够识别低密度区域，将其视为噪声。与传统的聚类算法（如K均值）相比，DBSCAN不需要预先指定簇的数量，能够处理各种形状的簇，并且对异常点具有较强的鲁棒性。

（a）DBSCAN关键概念 （b）DBSCAN聚类过程

图3-5 DBSCAN相关概念介绍

DBSCAN的基本思想是通过定义邻域半径和最小样本数来划定数据点的邻域，然后通过连接具有足够密度的邻域来形成簇。具体的定义如下：

* 邻域半径（）： 指定一个样本的邻域的最大半径，样本之间的距离小于 的话，它们被认为是邻居关系。
* 最小样本数（）： 指定一个核心点的邻域中至少应包含的样本数。如果一个邻域内的样本数大于等于，则该核心点及其邻域内的样本形成一个簇。
* 核心点（Core Point）： 在其邻域内包含至少 个样本的数据点。
* 边界点（Border Point）： 在其邻域内包含少于 个样本，但它是一个核心点的邻居的点。
* 噪声点（Noise Point）： 既不是核心点也不是边界点的点。

如图3-4（a）所示，为一个 ， 的示例图，由绿色圆圈圈出来的为随机选择的 邻域内至少存在4个数据点的核心点。将这些邻域内的数据点与之合并成一个簇，在簇中递归设置为核心点，当设置的核心点邻域内不满足至少存在4个点时，这个点被设为边界点。继续迭代算法，当整一个簇被边界点包围时，该簇的聚类结束。算法继续随机初始化一个核心点，继续DBSCAN算法。

如图3-4（b）所示，在DBSCAN中，一个点被认为是核心点，如果在其邻域内至少存在MinPts个数据点。然后，将所有在核心点邻域内的数据点合并成一个簇，并将这些点标记为已访问。对于每个核心点，DBSCAN会递归地扩展其邻域，将邻域内的点合并到同一个簇中。这个过程会继续，直到簇中的点不再满足成为核心点的条件。

对于异常点检测，DBSCAN算法在聚类过程中会将未被合并的点视为噪声点。在聚类结束后，未被聚类的点即为噪声点，也就是异常点。

* 1. 基于集成学习的方法

集成学习是一种通过结合多个学习器来构建更强大和鲁棒的模型的机器学习方法。这些学习器可以是同一类型的模型，也可以是不同类型的模型。集成算法的目标是通过整合多个模型的预测结果，从而提高整体性能，降低过拟合风险，并在处理复杂问题时取得更好的泛化能力。

集成算法的主要思想是通过对多个模型的预测进行组合，从而弥补单个模型的不足，提高整体性能。对于异常检测来说，集成算法包括Bagging[22]、Boosting[23]、Isolation Forest[12]、Extreme Gradient Boosting Outlier Detection (XGBOD)[24]等。

* + 1. 基于Feature Bagging的方法

1. Bagging（Bootstrap aggregating）

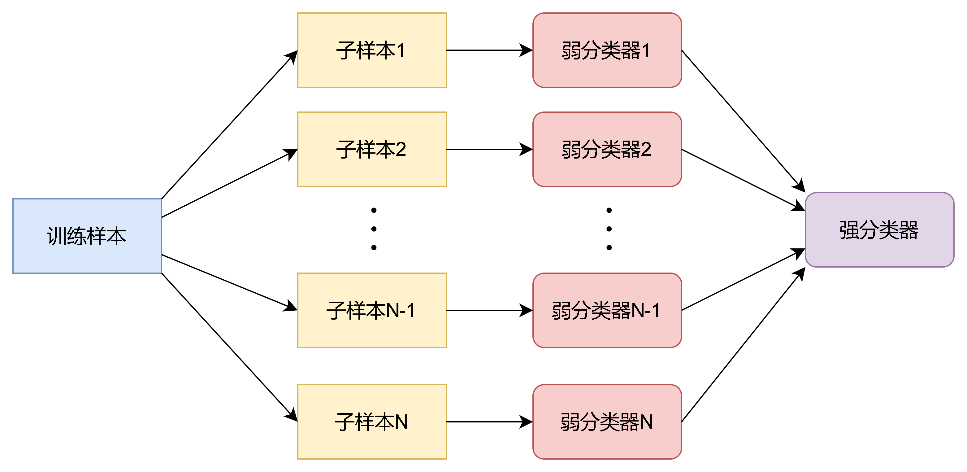


图3-6 Bagging示例图

Bagging（Bootstrap Aggregating）是一种集成学习方法，旨在提高模型的稳定性和准确性。该方法由Leo Breiman于1996年提出，如图3-6所示，通过对训练数据集进行有放回的自助采样（Bootstrap Sampling），产生多个不同的子样本，然后使用每个子样本独立训练一个基本模型。最后，通过对所有基本模型的预测进行组合来得到最终的集成模型。

Bagging的基本步骤如下：

* Bootstrap采样： 从原始的训练数据集中使用有放回的采样方式，随机抽取若干个样本，构成一个新的子样本集。这个过程重复多次，生成多个不同的子样本集。
* 独立训练： 对于每个子样本集，使用一个基本模型进行训练。这些基本模型可以是任意的机器学习算法，如决策树、支持向量机等。
* 集成预测： 对于新的输入样本，通过将所有基本模型的预测进行组合，通常使用投票（对于分类问题）或平均（对于回归问题）的方式得到最终的集成模型的预测结果。

1. Feature Bagging

Feature Bagging，是对于Feature的Bagging。给定集合 ，，标签为 ，其中0表示正常点，1表示异常点， 表示特征维度。在训练分类器之前，需要先进行归一化。假设分类器个数为 ，则对于每一个分类器，Feature Bagging算法步骤如下：

* 随机选取符合均匀分布的特征，数量为 ， 表示分类器索引；
* 使用 个特征构建原始样本的子空间 ；
* 在子空间 使用异常检测算法；
* 异常检测算法输出每个样本的异常值为 ；

计算最终每个样本的异常值 ：

* + 1. 基于孤立森林的方法

孤立森林（Isolation Forest）是一种基于集成学习的异常检测算法，通过构建一组孤立树来识别异常点。它是一种高效且可扩展的方法，特别适用于大规模数据集。

孤立森林的核心思想是假设异常点在特征空间中被孤立得较快，而正常点需要更多的分割才能被孤立。孤立森林通过构建随机分割的二叉树来实现这一目标，树的构建过程基于以下原理：

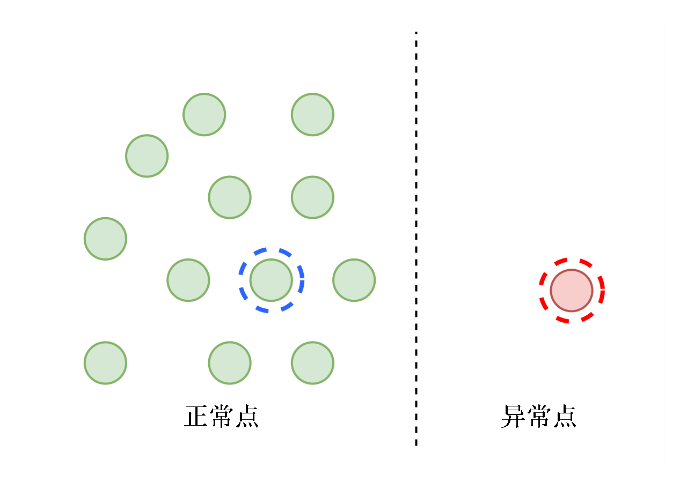


图3-7 孤立森林的假设可视化图

基于异常点的定义，异常点的数量总是要比正常点小很多。而且基于上述的基于密度的异常检测的方法，我们得知异常点往往会出现在数据点密度比较小的区域。如图3-7所示，蓝色虚线框选的是一个正常点，而红色虚线框选的是一个异常点。由于上述假设，若我们使用若干条线条来分割整个平面来挑选出这两个点，我们观察到正常点的挑选难度一定是高于异常点的。换言之，经过若干次的分割之后，异常点会先被挑选出来，而正常点由于其所在的区域密度较大，更难以被挑选出来。以下是孤立森林的具体算法：

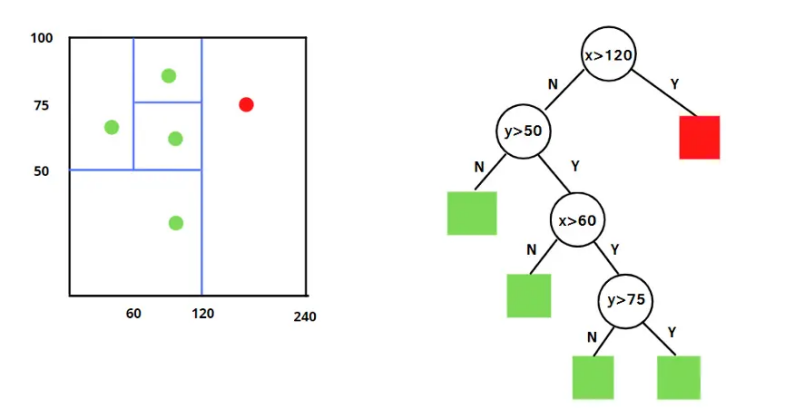
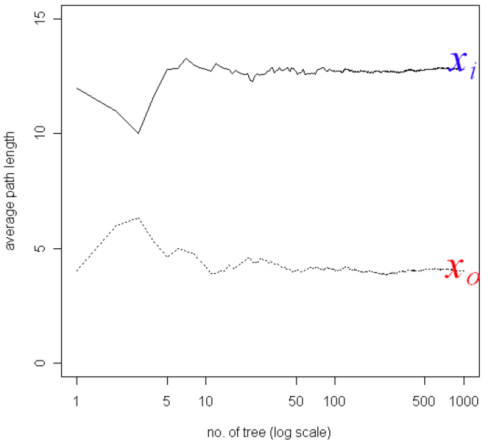
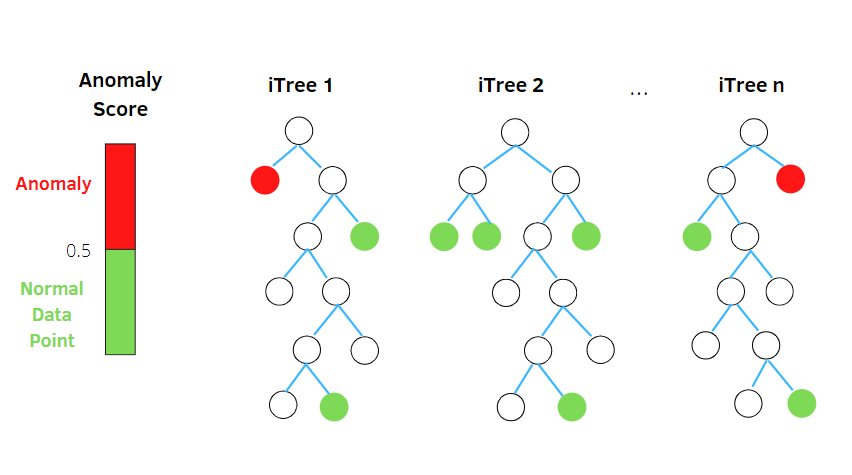


图3-8 单棵孤立树示例图

对于数据集 ，其中，。对于每一个数据集，随机选取 维度中的一维，随机在该维度中的最小值与最大值之间选取一个分割点 ，使得 被分为两部分，分别为该树的左右子树，若分割完的子空间只包含一个数据点或数的深度 已经到达了设定的最大值 ，则停止递归；否则，继续构造子孤立树。如图3-8所示，为构建一个单棵孤立树的示例。其中，可以观察到异常点在较小的划分次数就停止，正常点在较大的划分次数才停止。



（a）孤立森林示例图 （b）模型稳定性与孤立森林大小之间的关系

图3-9 孤立森林介绍

但由于构建单棵数存在随机性，包括随机选取维度、随机选取分割点。所以单凭一棵孤立树无法保证准确率。故需要构建 棵孤立树来构建孤立森林，最后计算每个数据点的平均深度，平均深度越小，该数据点越异常，反之，为正常点。

关于如何选取 的数值，如图3-9（b）所示，原论文中给出结果。当 时，数据点的平均深度比较稳定。

# 实验部分

1. 1. 数据集
2. * 1. MNIST

原始的手写数字MNIST数据集有6万个样本的训练集和1万个样本的测试集。它是NIST提供的一个更大集合的子集。这些数字的大小已被标准化，并在固定大小的图像中居中。该数据集被转换为异常检测检测，因为数字0类被认为是异常点，而从数字6类中采样700张图像作为异常点。另外，从784个特征中随机抽取100个特征。

* + 1. Arrhythmia

UCI机器学习存储库的原始心律失常数据集是一个多维度为279的多类分类数据集。这里丢弃了5个分类属性，总共274个属性。将最小的类，即3、4、5、7、8、9、14、15组合成异常点，其余的组合成正常点。

* + 1. Breastw

UCI机器学习存储库中的乳腺癌（原始）数据集是一个分类数据集，它记录了乳腺癌病例的测量值。有良性和恶性两类。这个数据集的维数为9。该数据集的恶性类被认为是异常点，而良性类中的点被认为是正常点。

* + 1. Cardiotocogrpahy

UCI机器学习存储库的原始心脏摄影（Cardio）数据集由产科专家分类的心脏图上的胎儿心率(FHR)和子宫收缩(UC)特征的测量数据组成。这是一个分类数据集，其中分类是正常的、可疑的和病理的。对于异常点检测，正常类别形成了正常点，而病理(异常)类别被下采样到176个点。可疑类被丢弃。

* + 1. HTTP

来自UCI机器学习存储库的原始KDD Cup 1999数据集包含41个属性(34个连续属性，7个分类属性)，然而，它们被减少到4个属性(服务，持续时间，src\_bytes, dst\_bytes)，因为这些属性被认为是最基本的属性(参见kddcup.names)，其中只有“服务”是分类属性。使用 service属性，数据被划分为 子集。这里只使用了“http”服务数据。由于连续属性值集中在“0”附近，我们将每个值转换为远离“0”的值，通过 。原始数据集在4,898,431条记录中有3,925,651次攻击(80.1%)。一个较小的集合是由976,157条记录中只有3,377次攻击(0.35%)伪造的，其中属性“logged\_in”为正。从这个伪造的数据集中，567,497个“http”服务数据被用来构造http (KDDCUP99)数据集。

* 1. 实验指标
     1. AUC指标

AUC (Area Under the ROC Curve) 是一种用于评估二分类模型性能的指标。ROC 是 Receiver Operating Characteristic 的缩写，表示接收者操作特征曲线。AUC 表示 ROC 曲线下的面积，即曲线与横轴之间的面积。

以下是关于 AUC 的一些关键概念和解释：

**(1) ROC 曲线：** ROC 曲线是一条以真正例率（True Positive Rate，TPR）为纵轴，假正例率（False Positive Rate，FPR）为横轴的曲线。TPR 是指正确预测为正例的样本数占所有实际正例的比例，而 FPR 是指错误预测为正例的样本数占所有实际负例的比例。

**(2) AUC 的计算：** AUC 表示 ROC 曲线下的面积，范围在 0 到 1 之间。AUC 越接近 1，说明模型性能越好。具体地，AUC 是 ROC 曲线与横轴之间的面积，计算方法可以通过积分或梯形法则进行估算。

**(3) 性能解释：** AUC 提供了一个综合性的性能度量，不受不同分类阈值选择的影响。模型在不同阈值下的表现通过 ROC 曲线反映，AUC 考虑了这个曲线下的整体性能。

* + 1. P@N指标

P@N，或者叫 Precision at N，是一种用于评估排名模型性能的指标。它衡量了在前 N 个预测中有多少是正确的正例。通常，P@N 用于在一个预测列表中衡量模型的预测精度，尤其是在排序任务中。

以下是关于 P@N 的一些关键概念和解释：

**(1) Precision（精度）：** Precision 是指模型在预测为正例的样本中有多少是真正例的比例。在二分类任务中，精度的计算公式为：Precision = TP / (TP + FP)，其中 TP 是真正例，FP 是假正例。

**(2) Precision at N（P@N）：** P@N 是在前 N 个预测中的精度。例如，P@5 表示在模型的前 5 个预测中有多少是正确的正例。

* 1. PyOD

PyOD (Python Outlier Detection) [11]是一个用于异常检测的 Python 库，提供了多种经典和先进的异常检测算法。异常检测是识别数据集中与大多数数据点不同的观察值的任务。PyOD 旨在简化异常检测的使用，并为用户提供一个灵活而强大的工具集。

以下是一些关键的 PyOD 特性和概念：

**(1) 多种算法支持**： PyOD 提供了许多常用的异常检测算法，包括但不限于：

* k-最近邻（k-Nearest Neighbors）
* Isolation Forest
* One-Class SVM
* AutoEncoder
* Mahalanobis Distance
* HBOS（Histogram-Based Outlier Score）

**(2) 集成方法**： 除了单独的算法之外，PyOD 还提供了一些集成方法，如 Average of Maximum (AOM) 和 Maximum of Average (MOA)，这些方法通过组合多个算法的输出来提高鲁棒性。

**(3) 支持多维数据**： PyOD 能够处理多维数据，因此你可以在高维空间中执行异常检测任务。

**(4) 直观的API**： PyOD 的 API 被设计得易于使用，用户可以轻松地拟合模型、进行预测和评估性能。

**(5) 可视化工具**： 提供了可视化工具，使用户能够更好地理解模型的性能和异常检测的结果。

* 1. 实验对比
     1. 生成数据集下的实验

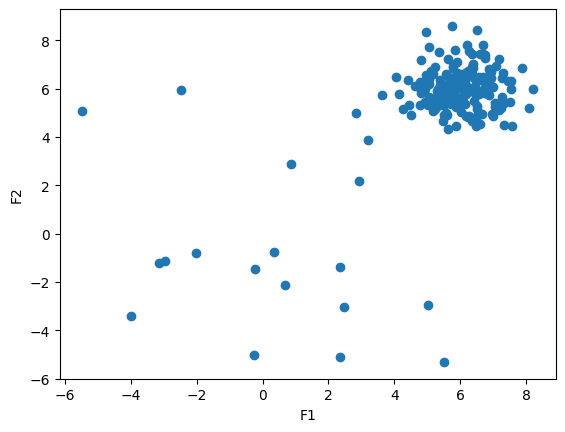


图4-1 生成数据集

如图4-1所示，使用pyod库生成了一个200个样本点，其中异常点的比例10%。下面使用PCA、OCSVM、LOF、HBOS、GMM、KNN、IForest、Feature Bagging来仿真实验。

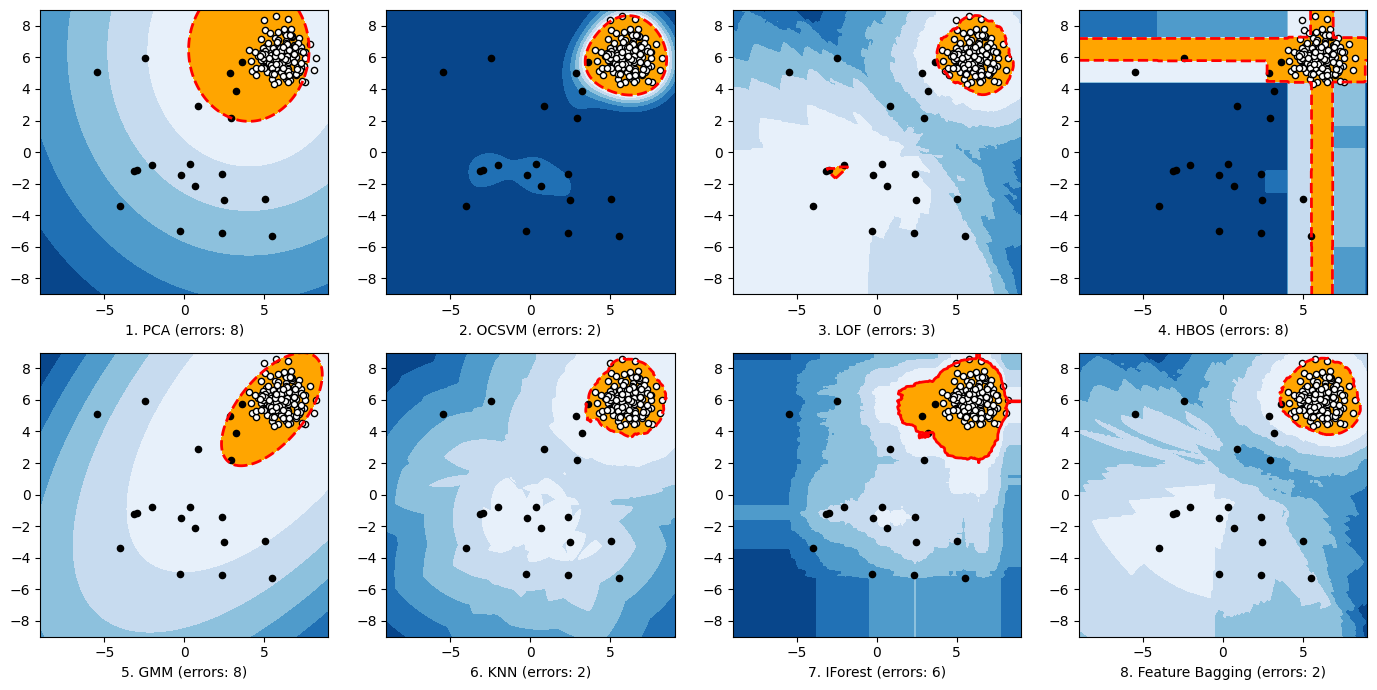


图4-2 各模型的异常检测效果可视化

如图4-2所示，为各个模型在生成数据集中异常检测效果的可视化展示，其中包括异常值的阶梯可视化、正常点判定范围以及错误样本个数。观察图像，可以看到在该生成的数据集中，OCSVM、KNN、Feature Bagging比其他模型略胜一筹，一方面原因是OCSVM、KNN模型在低维数据集中，另一方面，该数据集异常点特征与正常点有明显差异，这也是Feature Bagging取得较好效果的原因。

* + 1. 真实数据集下的实验

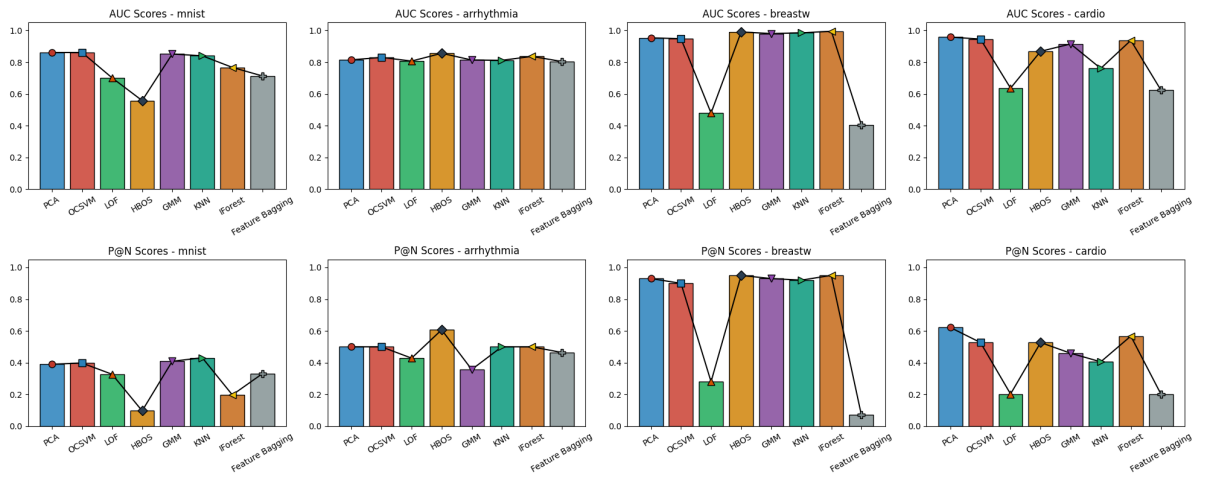
****

图4-3 各模型的4个真实世界数据集上的AUC分数与P@10分数

为了验证以上模型在真实世界的数据集中是否有效果，考虑到http数据集过于庞大，这里只涉及mnist、Arrhythmia、Breastw、Cardiotocogrpahy数据集。如图4-3所示，绘制出了各模型的4个真实世界数据集上的AUC分数与P@10分数，并使用不同标志表示不同的异常检测方法。观察图表，在不同的数据集上，IForest方法相对比较稳定。

* 1. SUOD

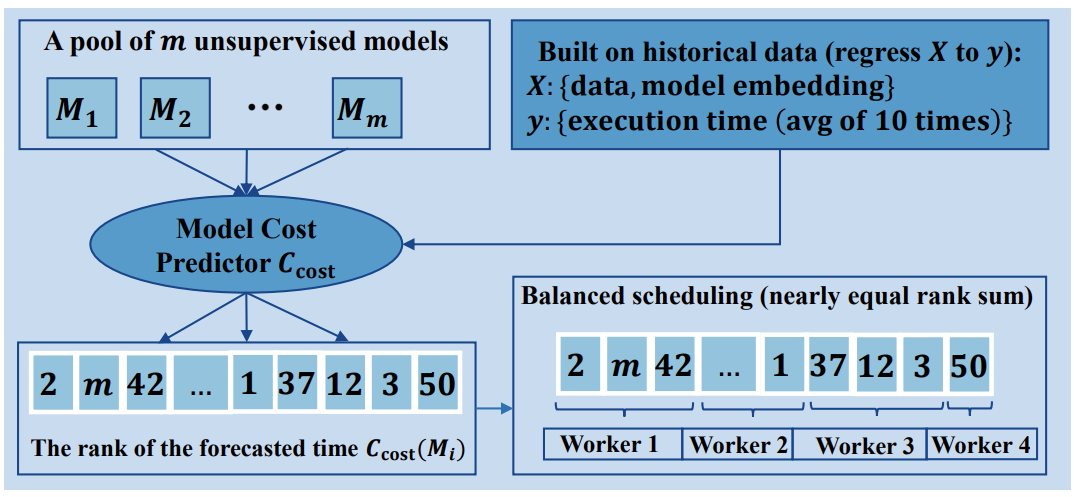


图4-4 SUOD调度多核CPU示例图

SUOD（Scalable Unsupervised Outlier Detection）[25] 是一种专为处理大规模无监督异常检测问题而设计的算法。其主要优势在于对高维数据集的高效处理，使其成为处理具有大量特征的数据异常检测的有力工具。

SUOD的首要特点之一是可伸缩性。通过采用数据分割和并行计算等技术，SUOD能够有效提高处理大规模数据集的效率，适用于处理包含大量记录和特征的数据。

在异常检测领域，不同算法的时间消耗差异较大，例如决策树算法相对于KNN算法具有较低的计算复杂度。为了有效地利用多核CPU资源，降低整体训练时间，研究人员提出了一种策略性的调度方法。其核心思想是通过训练一个回归模型，该模型能够预测每个异常检测算法在给定数据集上的计算时间。通过建模算法执行时间与输入数据之间的关系，SUOD能够在训练阶段估计每个算法的相对计算负担。随后，SUOD通过将算法分组，确保这些分组中的算法在多核CPU上几乎同时完成训练，从而显著提高整体训练效率和并行化水平。

如图4-4所示，该示例图表达了SUOD的多核CPU调度示例。通过对算法的计算时间进行建模和预测，SUOD有效地将算法分组，并使得这些分组中的算法能够在几乎相同的时间内完成训练，最大程度地减少了空闲时间，提高了整体训练效率。这种并行化策略对于处理大规模数据集和提高异常检测系统性能具有重要意义。

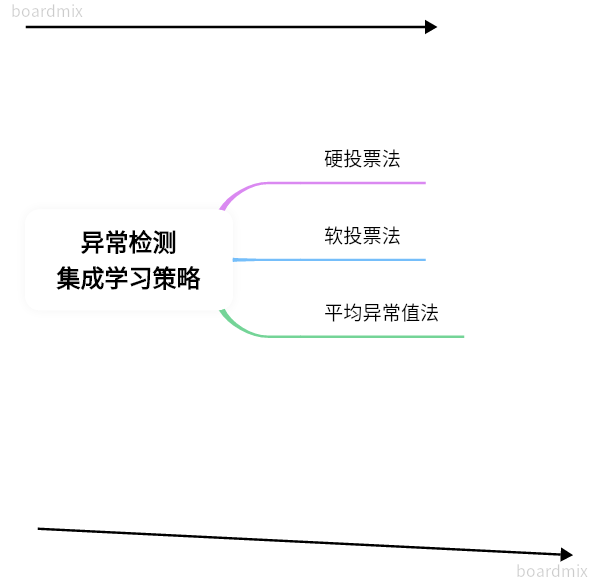
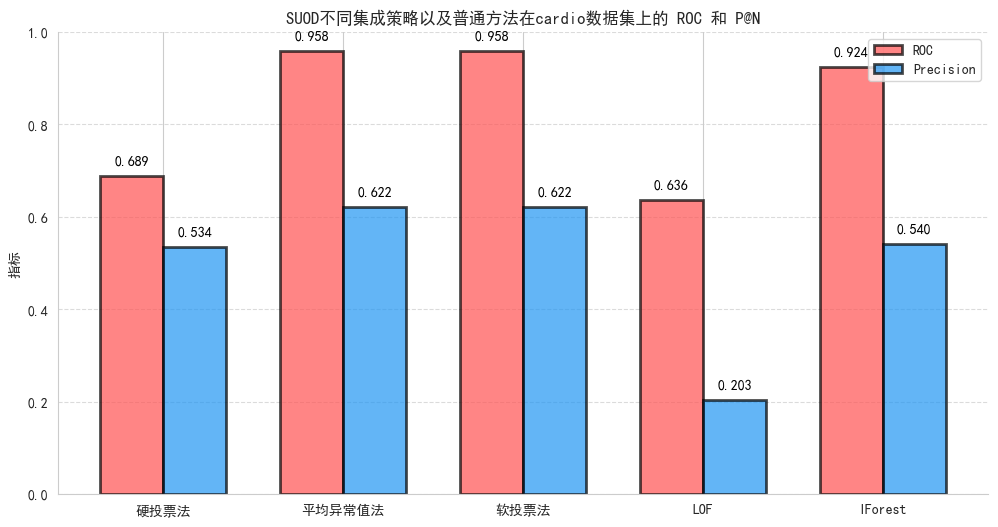
同时SUOD采用了基于集成学习的思想，通过整合多个异常检测模型的结果，以提高整体性能。不同于单一模型，SUOD提供多种集成策略，用户可根据实际问题选择不同的整合方式，如多数投票、平均值或最大概率等。

SUOD支持多种常见的基础模型，包括 PCA、OCSVM、LOF、HBOS、GMM、KNN、IForest等。这使得SUOD更加灵活，能够适应不同数据集的特点。

算法适用于高维数据，并通过设计能够在高维数据中表现较好。其结构允许在处理具有大量特征的数据时，仍能有效地捕获异常模式。

SUOD不仅提供了异常检测的结果，还具备较强的可视化和解释性。用户能够通过可视化工具和解释模型输出，更好地理解异常检测的结果及模型的工作原理。

综上所述，SUOD作为一个专为大规模高维数据设计的异常检测框架，以其可伸缩性、灵活性和解释性等特点，为异常检测任务提供了一种强大的工具。

** **

（a）异常检测集成学习策略 （b）指标图

图4-5 SUOD不同集成策略以及普通方法在cardio数据集上的 ROC 和 P@N图

如图4-5所示，为SUOD不同集成策略以及普通方法在cardio数据集上的 ROC 和 P@N图。观察图像，SUOD使用平均值和最大概率的集成方法时，其集成模型的ROC分数和P@N值最高，超过了传统方法的LOF和同样基于集成学习的IForest，在一定程度上可以说明，SUOD尤其是在大规模数据集上表现优异。

* 1. Pytod

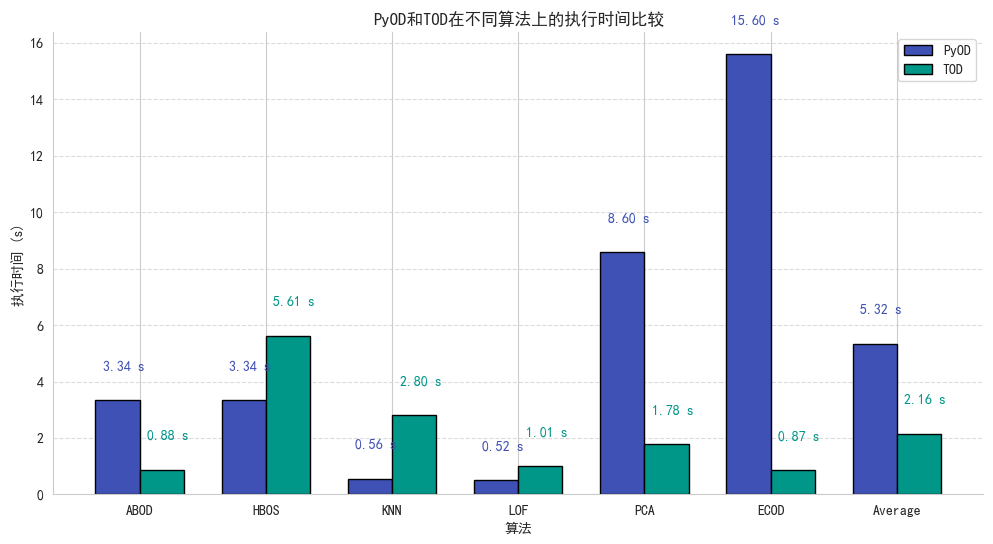


图4-6 PyOD和TOD在不同算法上的执行时间比较

由于PyOD和SUOD皆基于CPU架构，尽管SUOD采用了多进程的实现，但其在大规模数据集上的效率依然较为有限。当前，基于GPU的深度学习框架如TensorFlow和PyTorch在矩阵运算加速方面表现卓越，因此，SUOD的开发者借助这一机遇，推出了基于GPU的异常检测系统Pytod[26]。

如图4-6所示，对比了PyOD和TOD在不同算法上的执行时间。在小规模算法中，由于GPU调度所耗费的时间较多，TOD并未显著提升运算速度，反而在调度过程中增加了计算时间。然而，在处理较大规模或大数据量的数据集时，TOD充分发挥了其基于GPU加速矩阵运算的潜力。值得注意的是，在最后一项实验中，展示了两者在各算法上执行时间的平均值，结果显示TOD的平均效率是PyOD的两倍以上。

这一创新性的引入基于GPU的异常检测系统Pytod为异常检测领域注入了新的活力，为应对大规模数据集和复杂算法的挑战提供了有效的解决方案。

# 结论与展望

本文旨在深入调研异常检测在实际生活中的应用，涵盖了传统方法下基于密度、统计、距离、聚类、集成学习等多种异常检测算法，以及在Pyod、SUOD、PyTOD系统下的实验探究。对于传统方法而言，未来的深入研究可从以下几个方面展开：

**(1) 多样化数据类型的异常检测：**当前研究中所采用的数据集主要为统计数据集，然而实际生活中存在大量半结构化、非结构化以及序列数据集上的异常，如社交网络中的诈骗、工业设备在时间维度上的异常情况等。未来的研究可以拓展到不同数据类型，提高算法的适用性。

**(2) 大型数据集和复杂算法的性能优化：**虽然PyTOD实现了基于GPU的异常检测算法，但库中并不支持一些传统方法，如何在大型数据集或者复杂算法中提高异常检测算法的效率依然是一个值得研究的方向。进一步优化算法的性能，以适应更加复杂和庞大的实际应用场景。

**(3) 领域通用的异常检测算法：**异常检测在不同领域中的定义存在差异，有些异常点与正常点的差异巨大，而有些则相对微小。未来研究可以致力于实现更为通用的异常检测算法，使其能够适应不同领域的特殊需求，从而提高算法的普适性。

综上所述，未来的异常检测研究可以朝着更加广泛的数据类型、更高效的算法实现以及更为通用的应用方向不断拓展和深化。

# 参考文献

1. Wang H, Bah M J, Hammad M. Progress in outlier detection techniques: A survey[J]. Ieee Access, 2019, 7: 107964-108000.
2. Paula E L, Ladeira M, Carvalho R N, et al. Deep learning anomaly detection as support fraud investigation in brazilian exports and anti-money laundering[C]//2016 15th ieee international conference on machine learning and applications (icmla). IEEE, 2016: 954-960.
3. Porwal U, Mukund S. Credit card fraud detection in e-commerce: An outlier detection approach[J]. arXiv preprint arXiv:1811.02196, 2018.
4. Alrawashdeh K, Purdy C. Toward an online anomaly intrusion detection system based on deep learning[C]//2016 15th IEEE international conference on machine learning and applications (ICMLA). IEEE, 2016: 195-200.
5. Hodge V, Austin J. A survey of outlier detection methodologies[J]. Artificial intelligence review, 2004, 22: 85-126.
6. Breunig M M, Kriegel H P, Ng R T, et al. LOF: identifying density-based local outliers[C]//Proceedings of the 2000 ACM SIGMOD international conference on Management of data. 2000: 93-104.
7. Tang J, Chen Z, Fu A W C, et al. Enhancing effectiveness of outlier detections for low density patterns[C]//Advances in Knowledge Discovery and Data Mining: 6th Pacific-Asia Conference, PAKDD 2002 Taipei, Taiwan, May 6–8, 2002 Proceedings 6. Springer Berlin Heidelberg, 2002: 535-548.
8. Yang X, Latecki L J, Pokrajac D. Outlier detection with globally optimal exemplar-based GMM[C]//Proceedings of the 2009 SIAM international conference on data mining. Society for Industrial and Applied Mathematics, 2009: 145-154.
9. Knox E M, Ng R T. Algorithms for mining distancebased outliers in large datasets[C]//Proceedings of the international conference on very large data bases. Citeseer, 1998: 392-403.
10. Ester M, Kriegel H P, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise[C]//kdd. 1996, 96(34): 226-231.
11. Zhao Y, Nasrullah Z, Li Z. Pyod: A python toolbox for scalable outlier detection[J]. arXiv preprint arXiv:1901.01588, 2019.
12. Liu F T, Ting K M, Zhou Z H. Isolation forest[C]//2008 eighth ieee international conference on data mining. IEEE, 2008: 413-422.
13. Köppen M. The curse of dimensionality[C]//5th online world conference on soft computing in industrial applications (WSC5). 2000, 1: 4-8.
14. Cramer J S. Econometric applications of maximum likelihood methods[M]. CUP Archive, 1989.
15. L. Kaufman and P. J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis. Hoboken
16. Ng R T, Han J. CLARANS: A method for clustering objects for spatial data mining[J]. IEEE transactions on knowledge and data engineering, 2002, 14(5): 1003-1016.
17. MacQueen J. Some methods for classification and analysis of multivariate observations[C]//Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. 1967, 1(14): 281-297.
18. Hinneburg A, Keim D A. An efficient approach to clustering in large multimedia databases with noise[C]//Knowledge Discovery and Datamining (KDD'98). 1998: 58-65.
19. Zahn C T. Graph-theoretical methods for detecting and describing gestalt clusters[J]. IEEE Transactions on computers, 1971, 100(1): 68-86.
20. Guha S, Rastogi R, Shim K. CURE: An efficient clustering algorithm for large databases[J]. ACM Sigmod record, 1998, 27(2): 73-84.
21. Karypis G, Han E, Kumar V. A hierarchical clustering algorithm using dynamic modeling[J]. 1999.
22. Lazarevic A, Kumar V. Feature bagging for outlier detection[C]//Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. 2005: 157-166.
23. Rayana S, Akoglu L. Less is more: Building selective anomaly ensembles[J]. Acm transactions on knowledge discovery from data (tkdd), 2016, 10(4): 1-33.
24. Zhao Y, Hryniewicki M K. Xgbod: improving supervised outlier detection with unsupervised representation learning[C]//2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018: 1-8.
25. Zhao Y, Hu X, Cheng C, et al. Suod: Accelerating large-scale unsupervised heterogeneous outlier detection[J]. Proceedings of Machine Learning and Systems, 2021, 3: 463-478.
26. Zhao Y, Chen G H, Jia Z. TOD: GPU-accelerated Outlier Detection via Tensor Operations[J]. arXiv preprint arXiv:2110.14007, 2021.

# 致 谢

首先，感谢学院开设《模式识别》这门课，让我能够在这门课中学到模式识别相关的理论知识，实验上机课加强了自身动手能力。

同时，在《模式识别》这门课程的学习过程中，我深感荣幸能够有葛老师这位卓越的教师作为我们的课程老师。给我们讲述相关的内容，让我受益匪浅。

最后，向在我这几年在学习和生活中曾给予我支持和教导、扶持和帮助的老师和同学表示深深的谢意。