数据挖掘与应用

异常检测

授课教师：周晟

浙江大学　软件学院

2021.10



教学内容

1. 背景介绍
2. 基于统计的方法
3. 有监督/半监督异常检测算法
4. 基于距离的异常检测方法
5. 基于密度的方法
6. 基于聚类的方法
7. 基于概率分布的异常检测模型



数据中的异常

异常的定义

An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.

数据集中与大部分样本不一样的样本称为异常样本（anomaly sample, abnormalities, deviants, outliers）。

异常检测的定义

离群点检测（又称为异常检测）是找出行为不同于预期的异常的过程。



异常与噪声

噪声是被观测变量的随机误差或方差。
噪声在数据分析（包括异常检测）中不是令人感兴趣的。
离群点是有趣的，因此怀疑产生它们的机制不同于产生其他数据的机制。

1. 所有样本 = 普通样本 + 离群样本（Outlier）
2. 离群样本 = 噪声（Noise）+ 异常样本（Anomaly）
3. 噪声 = 用户没有兴趣的离群点
4. 异常 = 用户感兴趣的离群点



异常的类型

异常的常见类型

1. 点异常（Point Anomaly）
2. 条件异常（Conditional Anomaly）
   1. 时间条件异常
   2. 空间条件异常
3. 集合异常（Collective Anomaly）



点异常

- 单个数据实例是异常的

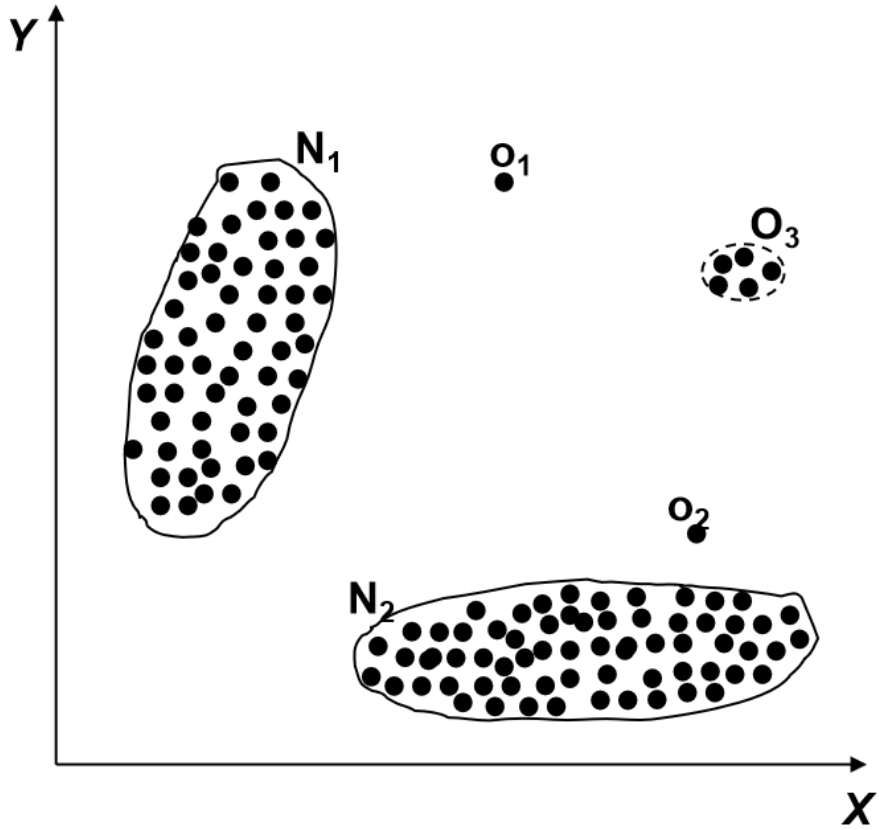


图: 点异常

条件异常

- 单个数据实例在某个条件下是异常的
- 需要有“条件”的前提
- 也称为“上下文异常（Contextual Anomalies)”

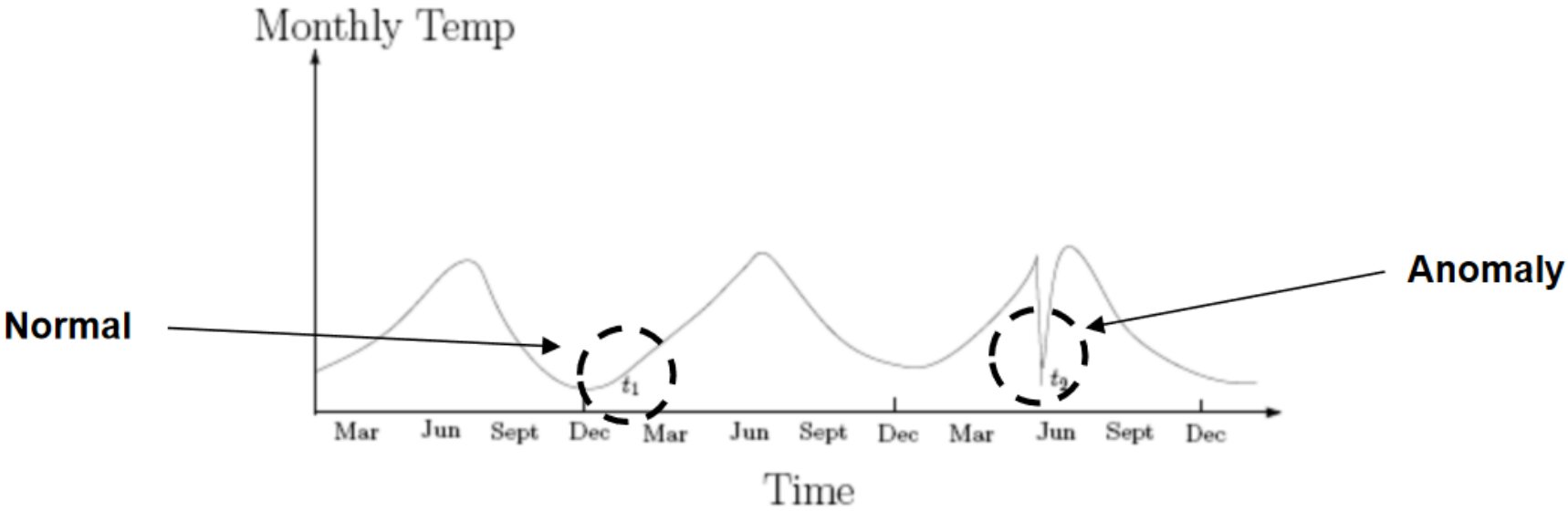


图: 条件异常

集合异常

- 一个由相关数据实例构成的集合是异常的
- 数据实例间存在某种关系
  - 连续数据
  - 空间数据
  - 图数据
- 集合中的单个数据实例本身并不是异常

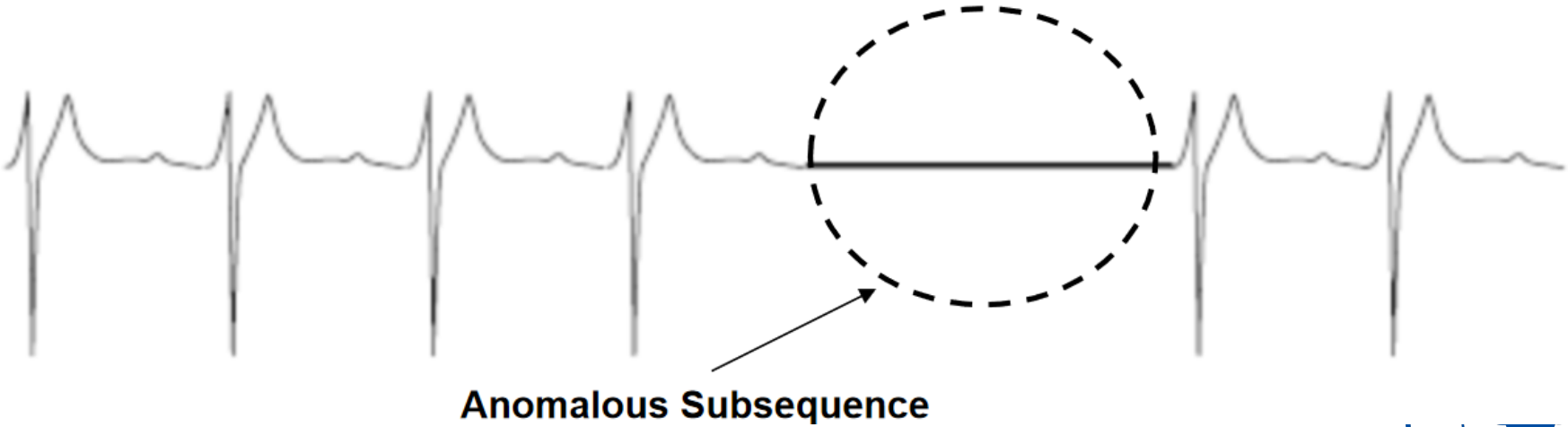


图: 集合异常

异常类型

不同异常类型的案例：

1. 超大金额的单比交易（点异常）
2. 发生在国外的小额交易（条件异常）
3. 发生在凌晨的小额交易（条件异常）
4. 连续十天购买同一件商品（集合异常）



异常检测的应用

1. 侵入检测（Intrusion Detection）
2. 欺诈检测（Fraud Detection）
   1. 保险
   2. 医疗
   3. 金融
   4. 电信
3. 恶意软件检测
4. 多媒体异常检测
5. 社交网络异常检测
6. 时序异常检测
   1. 单变量时序异常检测
   2. 多变量时序异常检测



侵入检测

- 侵入检测
  - 监视计算机系统或网络中发生的事件并对其进行侵入分析的过程
  - 侵入的定义是试图绕过计算机或网络安全机制的行为
- 面对的挑战
  - 传统基于签名（signature）的侵入检测方法是只能检测已知签名的侵入攻击，无法检测到新的侵入攻击
  - 跨计算机系统部署新创建的签名时存在大量延迟
- 异常检测一定程度上能够解决这些问题



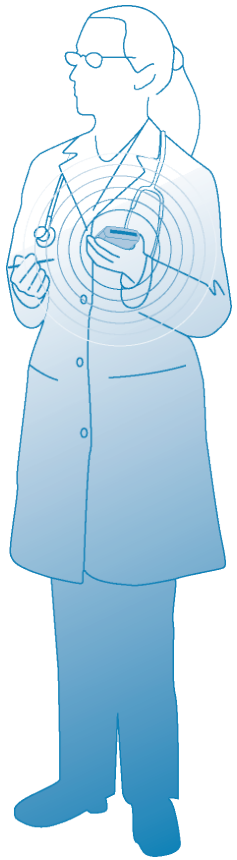
欺诈检测

- 欺诈检测
  - 侦查商业组织中发生的犯罪活动
  - 其中的恶意用户可能是组织的实际客户，也可能是伪装成的客户（也称为身份盗窃）
- 欺诈的类型
  - 信用卡诈骗
  - 保险索赔欺诈
  - 手机诈骗
  - 内幕交易
- 面对的挑战
  - 快速、准确的实时检测
  - 误分类成本非常高



医疗信息

- 医疗信息异常检测
  - 检测异常的患者记录
  - 发现疾病的爆发、仪表错误等
- 面对的挑战
  - 只有正常的标签可用
  - 误分类成本非常高
  - 数据可能非常复杂：时空数据



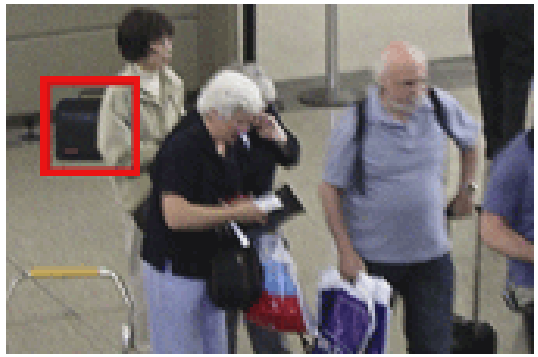
工业损伤检测

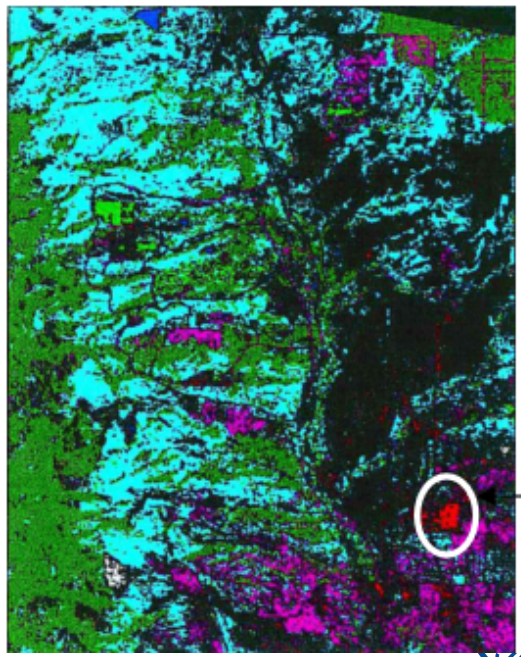
- 工业损伤检测
  - 对复杂工业系统中的故障、结构损害、电子安全系统中的入侵、异常能耗等进行检测。
- 以飞行安全为例
  - 异常的飞机使用
  - 发动机燃烧数据异常
  - 飞机总体运行状况和使用管理
- 面对的挑战
  - 数据极为庞大，杂乱且无标签
  - 许多数据具有时序性
  - 检测到异常事件需立即处理



图像处理

- 图像异常检测
  - 检测随时间监控图像中的异常值
  - 检测图像中的异常区域
- 使用场景
  - x 光检查图像分析
  - 视频监控
  - 卫星图像分析
- 面对的挑战
  - 集合异常检测
  - 数据集非常大





异常检测的挑战

1. 正常对象和异常对象的有效建模
2. 针对应用的异常检测
3. 在异常检测中处理噪声
4. 异常检测结果可解释性
5. 缺少标签数据
6. 面临人为对抗



异常检测相关知识

- 输入数据的性质
- 监督性
  - supervised
  - semi-supervised
  - unsupervised
- 异常的类型
- 异常检测的输出
- 异常检测的评价指标



输入数据

异常检测处理的最常见的数据形式是记录型数据（Record Data）

Tid	SrcIP	Start time	Dest IP	Dest Port	Number of bytes	Attack
1	206.135.38.95	11:07:20	160.94.179.223	139	192	No
2	206.163.37.95	11:13:56	160.94.179.219	139	195	No
3	206.163.37.95	11:14:29	160.94.179.217	139	180	No
4	206.163.37.95	11:14:30	160.94.179.255	139	199	No
5	206.163.37.95	11:14:32	160.94.179.254	139	19	Yes
6	206.163.37.95	11:14:35	160.94.179.253	139	177	No
7	206.163.37.95	11:14:36	160.94.179.252	139	172	No
8	206.163.37.95	11:14:38	160.94.179.251	139	285	Yes
9	206.163.37.95	11:14:41	160.94.179.250	139	195	No
10	206.163.37.95	11:14:44	160.94.179.249	139	163	Yes

图：记录型数据



输入数据——属性类型

属性的类型包括

- 二元数据
- 分类数据
- 连续数据
- 混合数据

	分类数据	连续数据	分类数据	连续数据	二元数据
Tid	SrcIP	Duration	Dest IP	Number of bytes	Internal
1	206.163.37.81	0.10	160.94.179.208	150	No
2	206.163.37.99	0.27	160.94.179.235	208	No
3	160.94.123.45	1.23	160.94.179.221	195	Yes
4	206.163.37.37	112.03	160.94.179.253	199	No
5	206.163.37.41	0.32	160.94.179.244	181	No

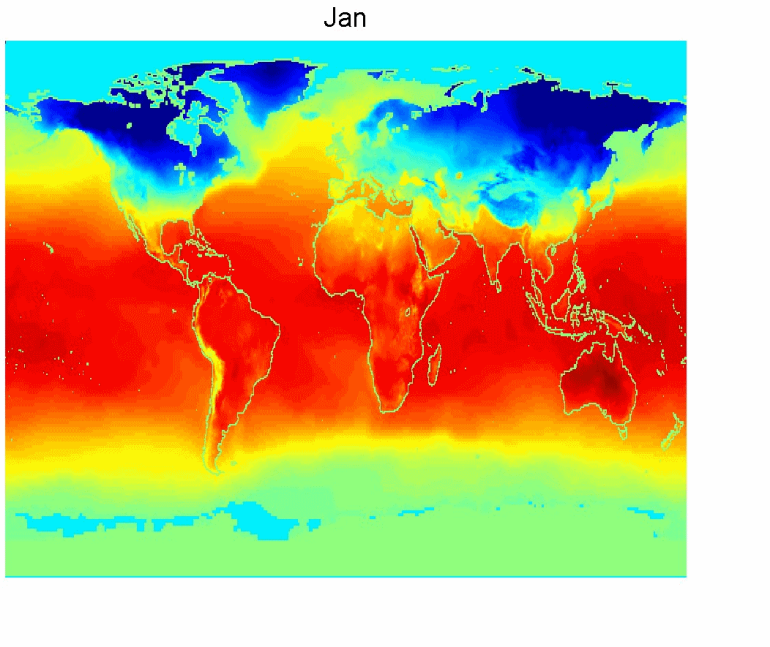
图: 属性类型

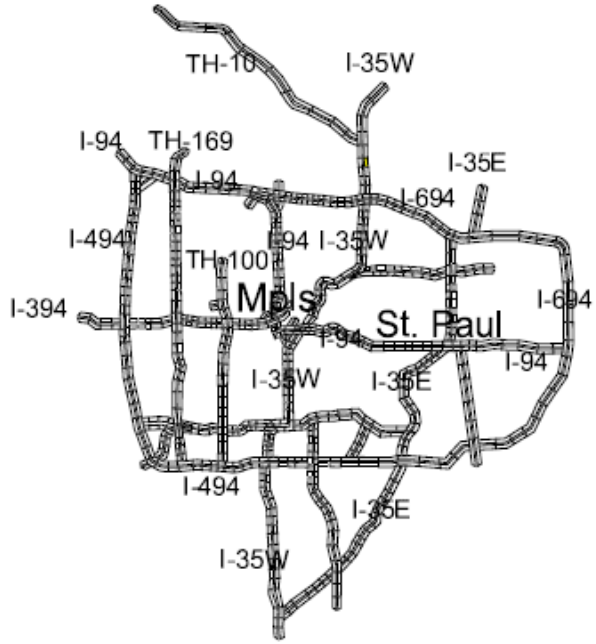
输入数据——复杂数据类型

数据实例之间存在不同的关系：

- 顺序（如时间）
- 空间
- 时空
- 图（graph）

```
GGTTCCGCCTTCAGCCCCGCGCC
CGCAGGGCCCGCCCCGCGCCGTC
GAGAAGGGCCCGCCTGGCGGGCG
GGGGGAGGCGGGGCCGCCCGAGC
CCAACCGAGTCCGACCAGGTGCC
CCCTCTGCTCGGCCTAGACCTGA
GCTCATTAGGCGGCAGCGGACAG
GCCAAGTAGAACACGCGAAGCGC
TGGGCTGCCTGCTGCGACCAGGG
```





异常检测的输出

- 标签
  - 每一个测试样本都被分配一个“正常“”或者“异常“”的标签
  - 特别是对于 supervised 的方法
- 分数
  - 每一个测试样本都被分配一个“异常”分数
  - 因此还需要定义一个阈值来进一步分类



异常检测的评价指标

准确率 ACC 不能作为一个合适的评价指标：

- 比如一个网络流量数据集中有 99.9% 的正常数据和 0.01% 的异常数据，此时一个将所有样本都判断为正常的分类器可以达到 99.9% 的准确率！
- 异常检测的类大小通常极不平衡！



异常检测的评价指标——F 度量

- F 度量的关注点在于召回率和精度

$$Recall(R) = \frac{TP}{TP + FN} \quad , \quad Precision(P) = \frac{TP}{TP + FP}$$

$$\text{F-score} = \frac{2 \times R \times P}{R + P}$$

Confusion matrix		预测的类	
		异常	正常
实际的类	异常	TP	FN
	正常	FP	TN



异常检测的评价指标——ROC & AUC

- ROC 曲线是伪阳性率（False Positive Rate, FPR）和真阳性率（True Positive Rate, TPR）之间的权衡。

$$TPR = \frac{TP}{TP + FN} = \frac{TP}{P}$$

$$FPR = \frac{FP}{FP + TN} = 1 - \frac{TN}{N}$$

- AUC 曲线即 ROC 曲线下方的面积，取值范围在 [0,1]。值越大，说明模型的性能越好。



1 背景介绍

2 基于统计的方法

3 有监督/半监督异常检测算法

4 基于距离的异常检测方法

5 基于密度的方法

6 基于聚类的方法

7 基于概率分布的异常检测模型

8 基于树的方法



Box-Plot 中的异常检测

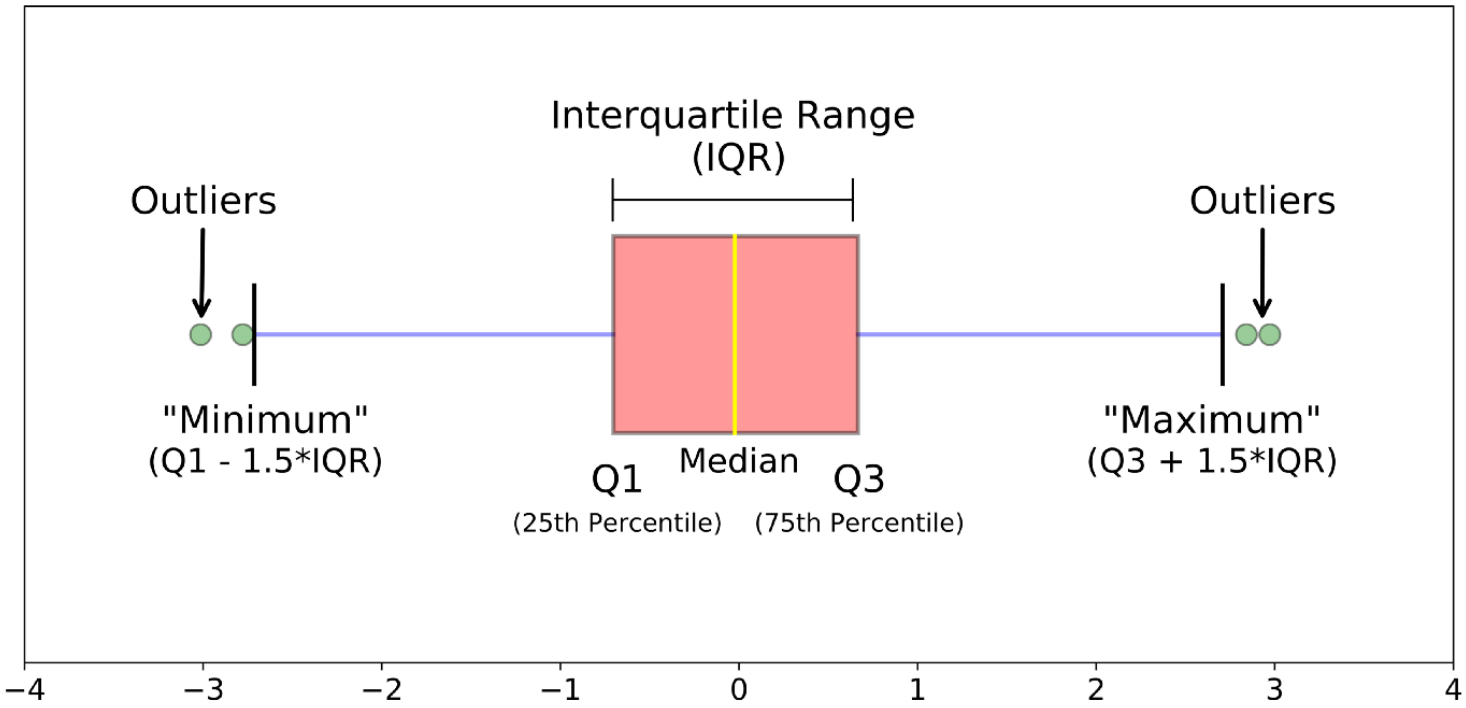


图: Boxplot 中的异常值区分



Box-Plot 中的异常检测

1. 第一四分位数（Q1）：也称下四分位数（Lower Quartile），等于该样本中所有数值由小到大排列后第 25% 的数字。
2. 第二四分位数（Q2）：也称中位数（Middle Quartile or Median），等于该样本中所有数值由小到大排列后第 50% 的数字。
3. 第三四分位数（Q3）：也称上四分位数（Upper Quartile），等于该样本中所有数值由小到大排列后第 75% 的数字。

$$\text{Lower Whisker} = Q_1 - 1.5\Delta Q$$
$$\text{Upper Whisker} = Q_3 + 1.5\Delta Q$$



Box-plot 中的异常检测

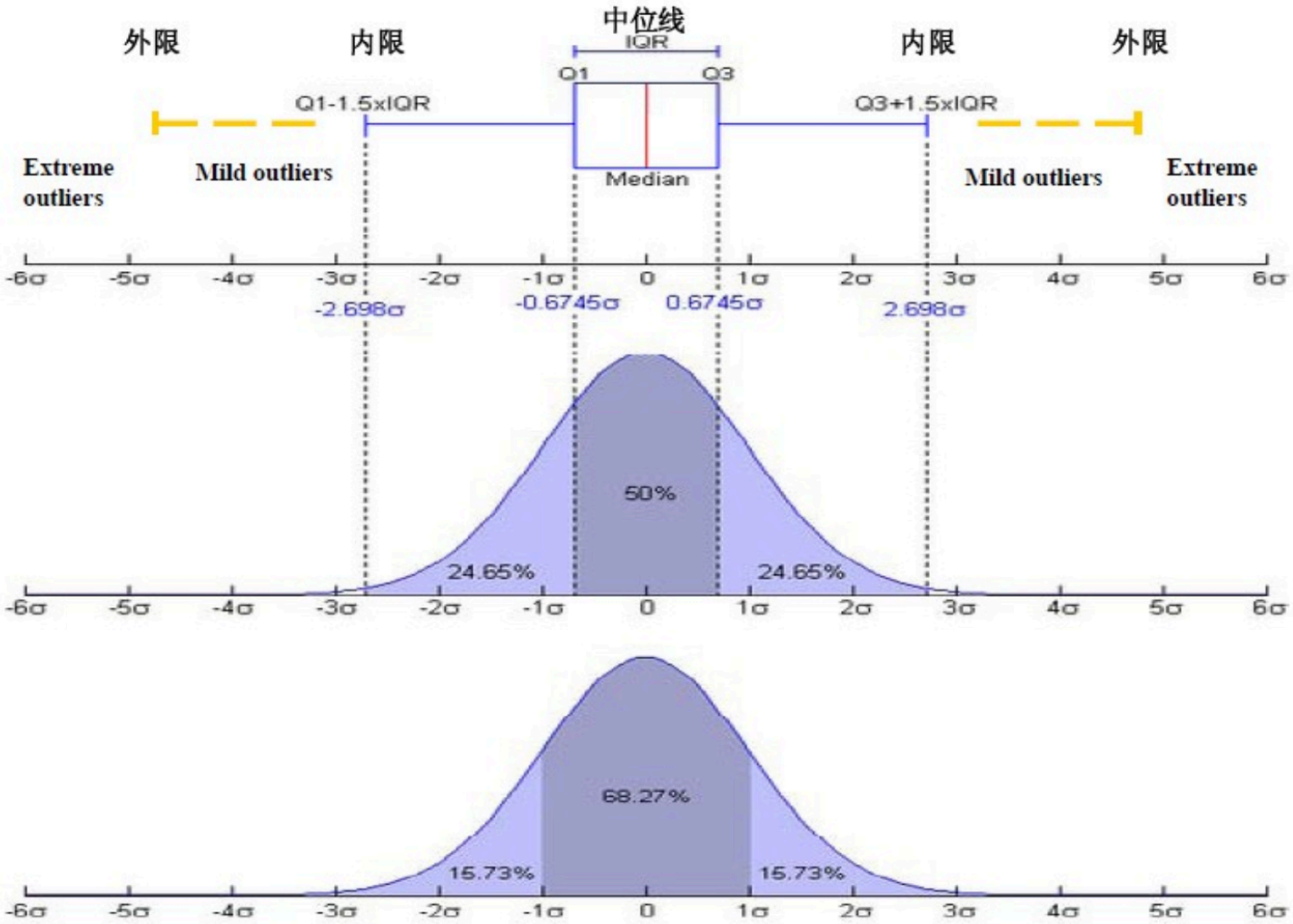


图: BoxPlot 中的异常检测

1 背景介绍

2 基于统计的方法

3 有监督/半监督异常检测算法

4 基于距离的异常检测方法

5 基于密度的方法

6 基于聚类的方法

7 基于概率分布的异常检测模型

8 基于树的方法



有监督/半监督异常检测算法

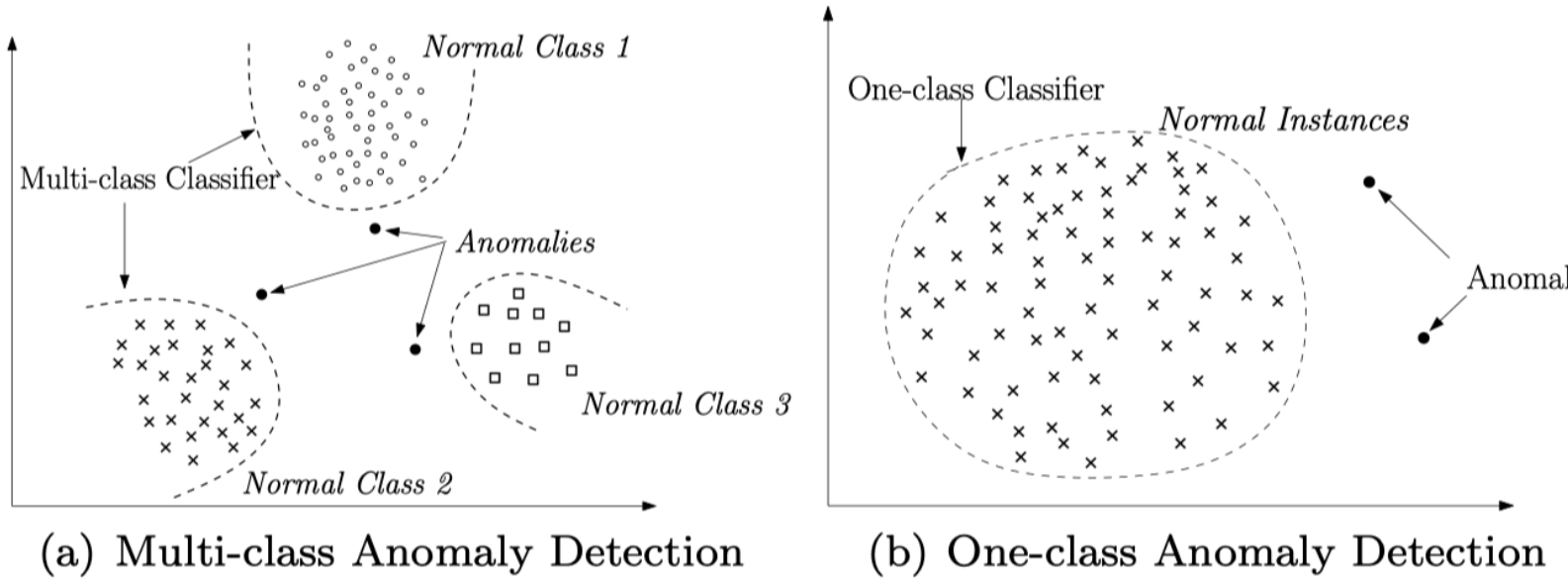


图: 基于分类的异常检测算法.

通过训练一个二分类的分类器，来对没有标签的数据进行分类。

有监督/半监督异常检测

优点

1. 快速高效
2. 能够快速发现已知类型的异常

缺点

1. 标签获取成本过高
2. 类别不平衡
3. 可解释性差
4. 只能发现已经定义好的异常类型
5. 大规模数据难以应用
6. 没有考虑全局信息

1 背景介绍

2 基于统计的方法

3 有监督/半监督异常检测算法

4 基于距离的异常检测方法

5 基于密度的方法

6 基于聚类的方法

7 基于概率分布的异常检测模型

8 基于树的方法



基于距离的方法

基本假设

异常数据通常距离正常的数据较远。
异常的分值可以用样本到它的邻居的距离来定义。

优点

1. 简单直接
2. 影响了许多后续工作
3. 无监督，数据驱动

缺点

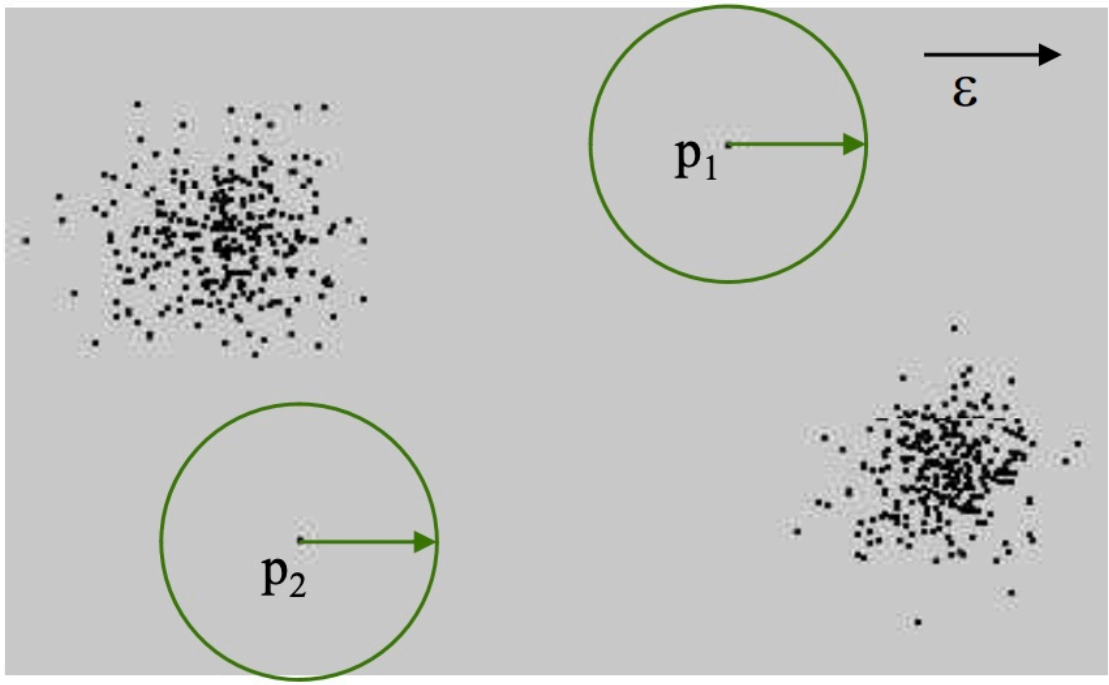
1. 依赖特征
2. 难以处理高维数据
3. 结果难以解释
4. 对数据的密度敏感

基于距离的异常检测方法:$DB(\varepsilon, \pi)$-Outliers

定义

数据集中与样本的距离小于 ε 的样本比例不低于 π

$$\text{OutlierSet}\ (\varepsilon, \pi) = \left\{ p \mid \frac{\text{Card}(\{\{q \in DB \mid \text{dist}(p, q) < \varepsilon\})}{\text{Card}(DB)} \leq \pi \right\}$$



基于距离的异常检测算法：KNN-Outlier

定义

使用 KNN 得到样本的 K 近邻之后，用 K 近邻的平均距离作为异常分值。也可以融合样本的 1NN,2NN,...,KNN 平均距离作为异常的分值。

变种

在给定数据特征的基础上，构造有向 KNN 图。如果 a 样本是 b 样本的 K NN，则存在一条有向的边从 b 指向 a。通过统计样本的入度作为异常的指标。

缺陷

1. 复杂度过高。$O(N^2)$。
2. 依赖数据在特征空间中的表示。

解决方案？

基于 KNN 的异常检测算法

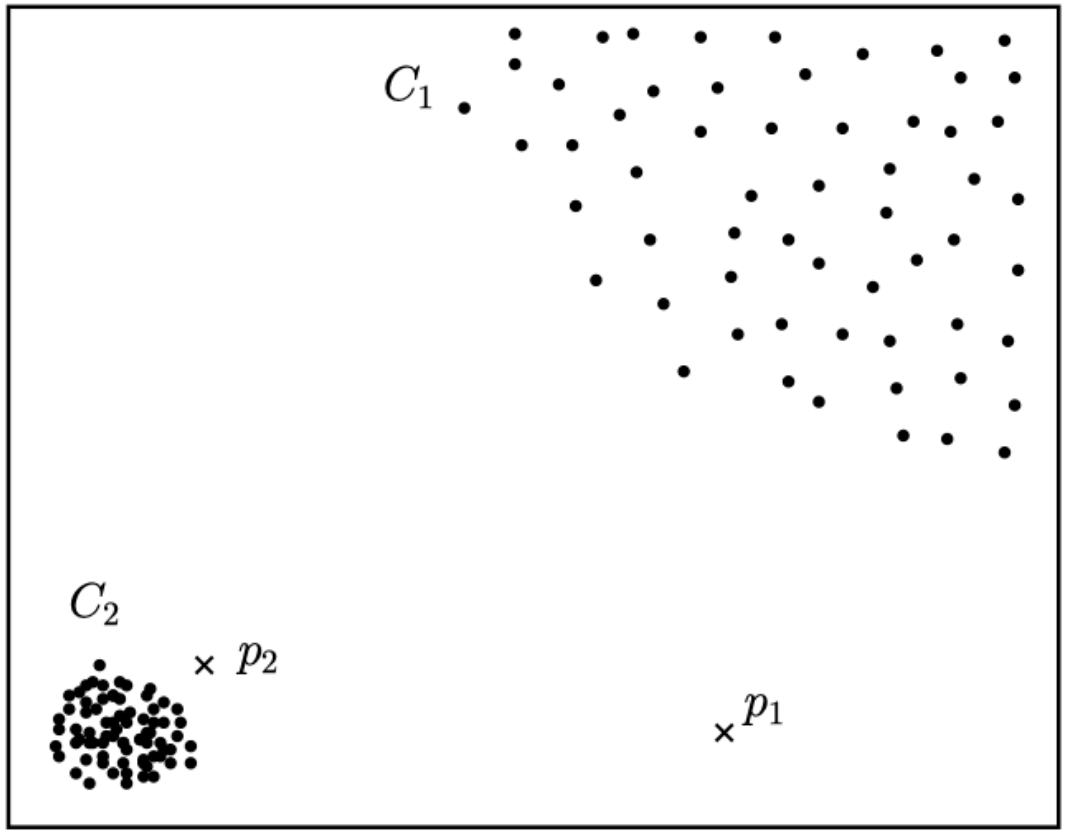


图: 如果数据整体的密度并不均匀，则 KNN 的方法可能失效

1 背景介绍

2 基于统计的方法

3 有监督/半监督异常检测算法

4 基于距离的异常检测方法

5 基于密度的方法

6 基于聚类的方法

7 基于概率分布的异常检测模型

8 基于树的方法



基于密度的方法：Local Outlier Factor(LOF)

显而易见地，我们会有这样一个想法：附近密度较低的数据实例可能是一个异常点。

然而，这个想法对于那些具有不同密度的簇的情况是不适用的。

局部离群值

数据实例 k 个最近邻的平均局部密度与实例本身的局部密度之比

LOF 考虑的是数据实例周围的密度。



Local Outlier Factor (LOF)

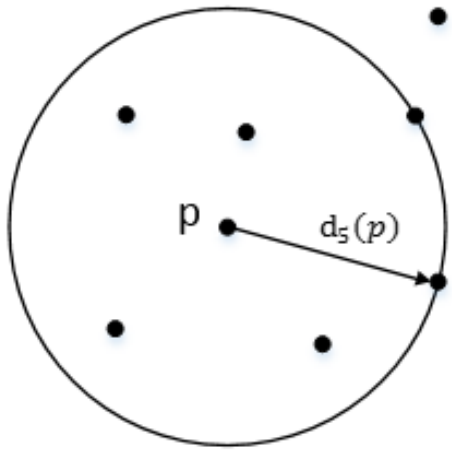
LOF 的相关定义：

- k-distance：第 k 近距离

  在数据点 p 附近的点中，第 k 近的点和点 p 之间的距离，记为 k-distance(p)，或 $d_k(p)$。

- k-distance neighborhood：第 k 距离邻域

  点 p 的第 k 距离邻域 $N_k(p)$ 包含与 p 距离小于等于 k-distance(p) 的所有点。因此有 $|N_k(p)| \geq K$。



$k = 5$ 时的情况

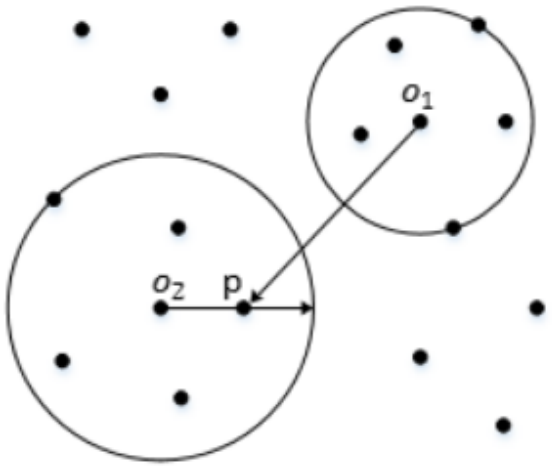
Local Outlier Factor (LOF)

- reach-distance：可达距离

  点 o 到点 p 的第 k 可达距离定义为

$$\text{reach-distance}_k(p,o) = \max\{\text{k-distance}(o), d(p,o)\}$$

即点 o 到点 p 的第 k 可达距离，至少是 o 的 k-distance，或者为 o、p 间的真实距离。这也意味着，离点 o 最近的 k 个点，o 到它们的第 k 可达距离是相等，且都等于 $d_k(o)$。



$$\text{reach-distance}_k(p,o_1) = d(p,o_1)$$
$$\text{reach-distance}_k(p,o_2) = d_5(o_2)$$

Local Outlier Factor (LOF)

- local reachability density (lrd)：局部可达密度

  点 p 的第 k 局部可达密度定义为

$$lrd_k(p) = 1/\left(\frac{\sum_{o\in N_k(p)} \text{reach-distance}_k(p,o)}{|N_k(p)|}\right)$$

  即点 p 的第 k 邻域内的点到点 p 的平均可达距离的倒数。

  一定程度上可以表示点 p 附件的密度：平均可达距离越小，局部可达密度越大。



Local Outlier Factor (LOF)

- local outlier factor(LOF)：局部离群因子

  点 p 的局部离群因子定义为

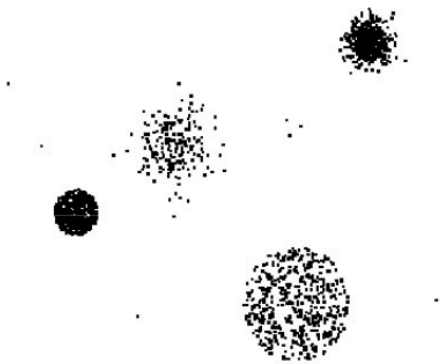
$$LOF_k(p) = \frac{1}{|N_k(p)|} \sum_{o \in N_k(p)} \frac{lrd_k(o)}{lrd_k(p)} = \frac{\sum_{o \in N_k(p)} lrd_k(o)}{|N_k(p)| \cdot lrd_k(p)}$$

  即点 p 第 k 邻域 $N_k(p)$ 中点的局部可达密度与点 p 的局部可达密度之比的平均数。

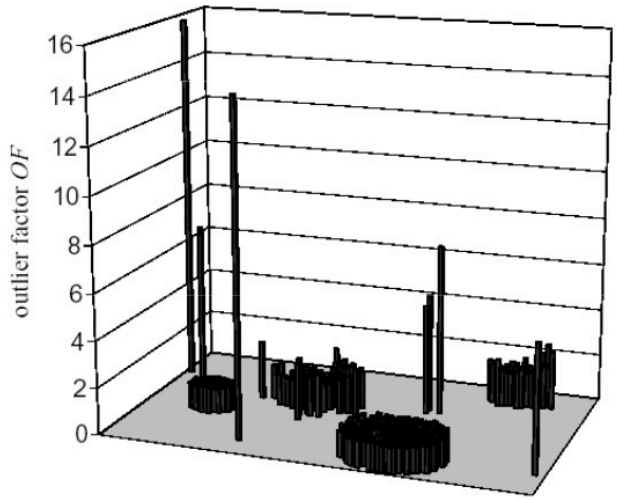


Local Outlier Factor (LOF)

- $LOF \approx 1$: 点 p 与其邻域点密度相似，p 可能和其邻域同属一簇。
- $LOF \ll 1$: 点 p 的密度大于其邻域点密度，p 为密集点。
- $LOF \gg 1$: 点 p 的密度小于其邻域点密度，p 可能是异常点。



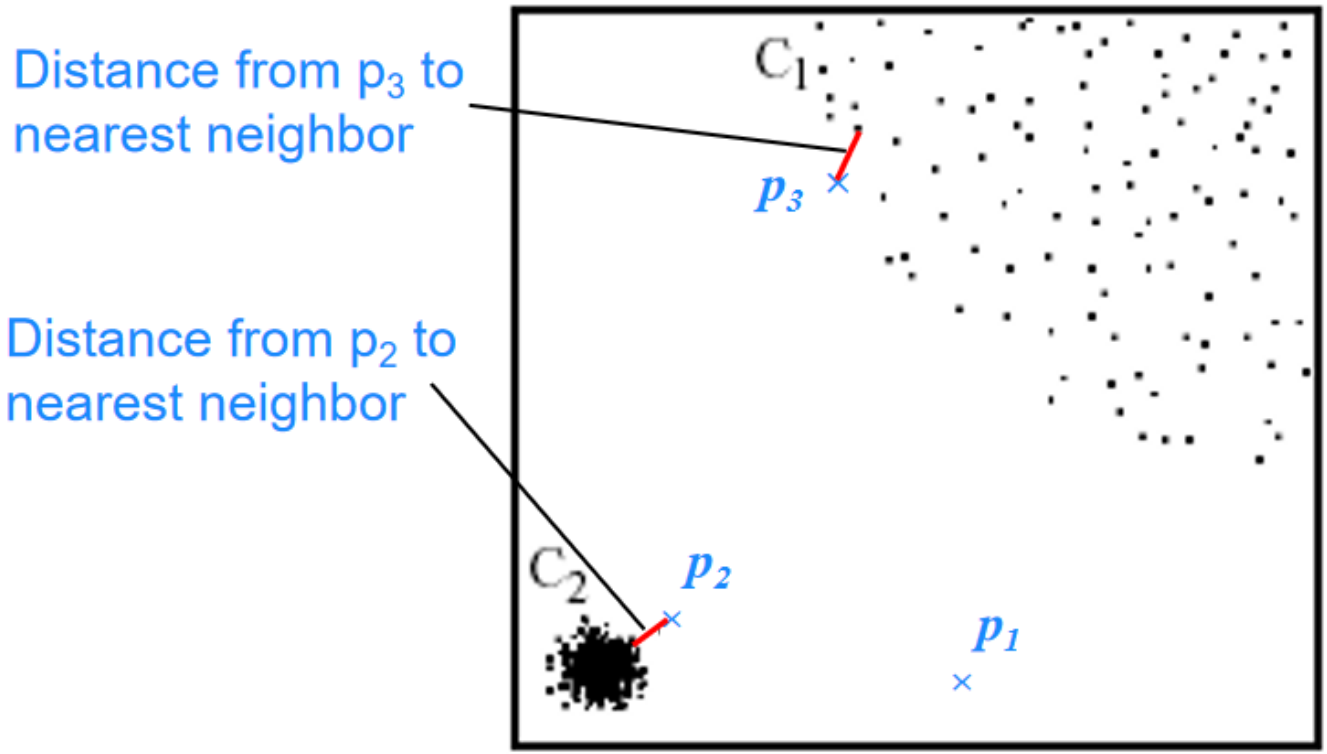
Data set



LOFs (*MinPts* = 40)

基于密度的方法的优点

- 在基于距离的方法中，p_2 不会被作为异常点，而 LOF 则会将 p_1 和 p_2 都判为异常点。
- 基于距离的方法可能会将 p_3 当做异常点，而 LOF 不会。



LOF 的各种变体

1. 局部密度的不同估计方式

   Connectivity-based Outlier Factor(COF): 迭代地添加与现有实例最接近的实例

2. 处理不同类型的数据

   学习其表征，找到更好的距离/相似性度量方式

3. 提高效率

   数据划分，哈希以及聚类



1 背景介绍

2 基于统计的方法

3 有监督/半监督异常检测算法

4 基于距离的异常检测方法

5 基于密度的方法

6 基于聚类的方法

7 基于概率分布的异常检测模型

8 基于树的方法



基于聚类的方法

研究动机

聚类和异常检测是最具代表性的两个无监督任务，两者假设类似且彼此依赖。

基于聚类的异常检测算法的假设

1. 正常的数据会呈现一定的聚类分布，而异常数据不会属于任何类。
2. 正常数据和他们最近的聚类中心比较接近，异常数据与聚类中心距离较远。
3. 正常数据属于大且稠密的类，异常数据不属于任何类或者属于小的稀疏的类。



基于聚类的异常检测算法

1. 对数据进行聚类，并把不在类中的数据作为异常数据。
   1. DBSCAN
   2. ROCK
   3. SNN clustering

依赖于聚类的质量，效果难以保证

2. 对于每个数据，计算它到最近的聚类中心的距离，并作为异常分值

Two-step method

3. Cluster-Based Local Outlier Factor (CBLOF)

聚类大小 + 到聚类中心的距离



Cluster-Based Local Outlier Factor (CBLOF)

CBLOF 使用聚类来确定数据中较密集的区域，并依靠聚类中心来衡量每一个数据点的局部异常因子。

- 首先，选择某种聚类方法对数据进行聚类（常用 K-means），获得若干个簇。
- CBLOF 采用启发式的方式将这些簇分为大簇和小簇。具体过程如下
  - 将所有簇按照从大到小的顺序排好

$$|C_1| \geq |C_2| \geq |C_3| \geq ... \geq |C_k|$$



Cluster-Based Local Outlier Factor (CBLOF)

- 定义两个参数 α 和 β，簇的划分需要满足两个条件：

$$(|C_1| + |C_2| + ... + |C_b|) \geq |D| \cdot \alpha$$
$$|C_b|/|C_b + 1| \geq \beta$$

分别称为绝对多数原则和突降原则。$b/b+1$ 即为大小簇划分点。

$$LargeClusters(LC) = \{C_1, C_2, ..., C_b\}$$
$$SmallClusters(SC) = \{C_{b+1}, C_{b+2}, ..., C_k\}$$

- 最优情况是能找到同时满足两个条件的分割，次优为满足绝对多数原则，最差是只找到了满足突降原则的分割。



Cluster-Based Local Outlier Factor (CBLOF)

- 数据点 p 的离群因子定义为

$$CBLOF(p) = \begin{cases} |C_i| \cdot \min(distance(p, C_j)) & \text{where } p \in C_i, C_i \in SC \text{ and} \\ & C_j \in LC \text{ for } j = 1 \text{ to } b \\ |C_i| \cdot distance(p, C_i) & \text{where } p \in C_i \text{ and } C_i \in LC \end{cases}$$

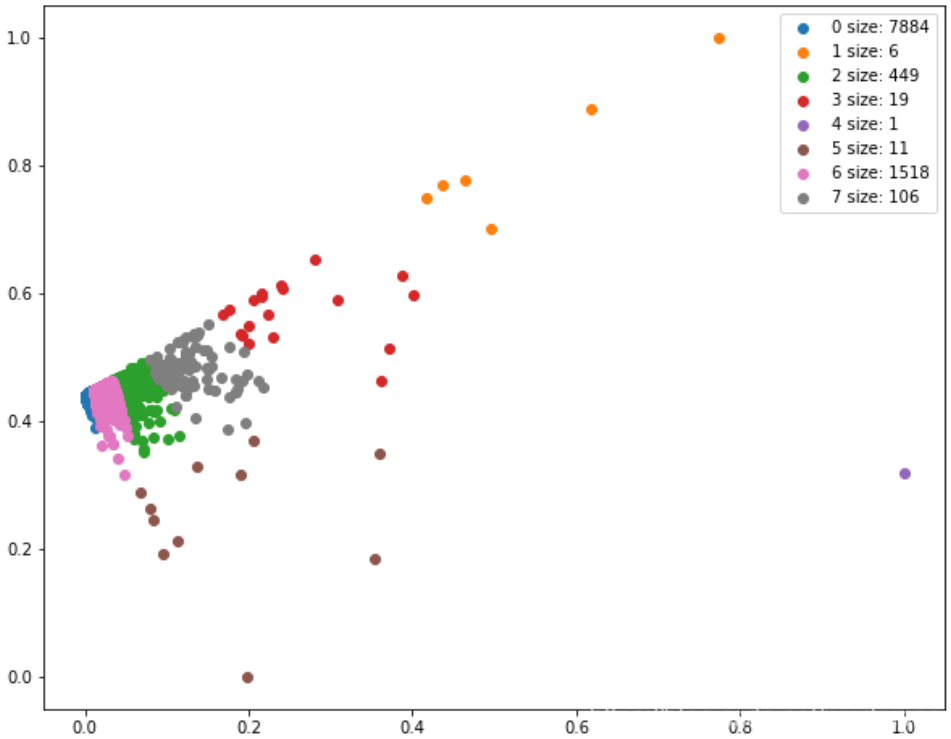
若点 p 为大簇中的点，则直接计算当前簇大小与 p 到当前簇中心距离的乘积。

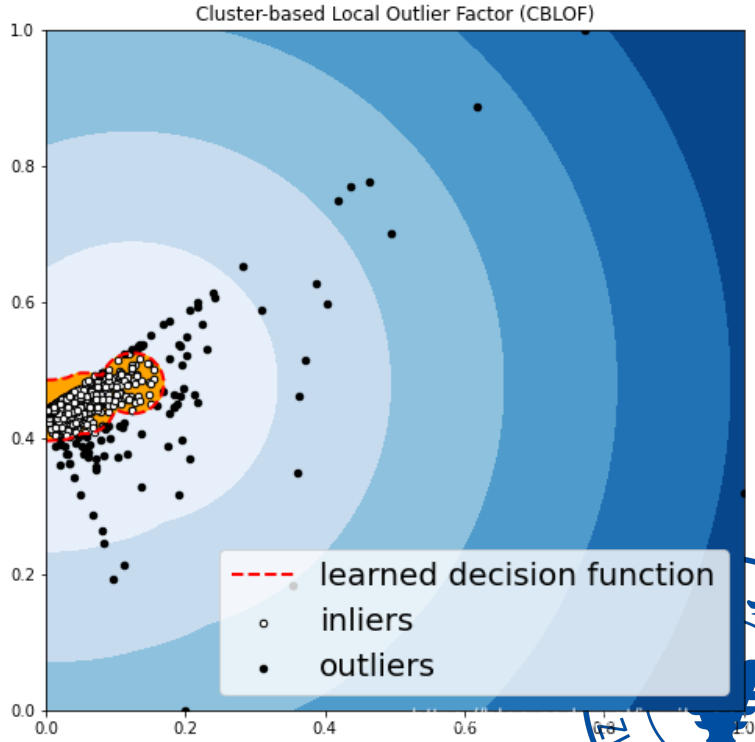
否则，计算当前簇大小与 p 到最近的大簇中心距离的乘积。



Cluster-Based Local Outlier Factor (CBLOF)

- 获得所有数据点的离群因子后，定义异常点的比例（一般为 1%），找出离群因子最大的 1% 的点，将其作为异常点。





基于聚类的异常检测算法

优点

1. 有很多现成的聚类算法
2. 得到聚类结果之后可以快速得到异常检测结果

缺点

1. 依赖聚类算法的效果
2. 聚类不是为异常检测单独优化



1 背景介绍

2 基于统计的方法

3 有监督/半监督异常检测算法

4 基于距离的异常检测方法

5 基于密度的方法

6 基于聚类的方法

7 基于概率分布的异常检测模型

8 基于树的方法



基于概率分布的异常检测模型

先验假设

许多真实数据是由概率分布生成的，而异常数据则是与整体数据的概率分布不同。

模型假设

正常样本出现在一个分布的高概率密度区域，而异常样本则出现在低概率密度区域。

通过学习数据集的概率密度参数，估计出样本所处的概率密度空间，并估计置信度。



基于概率分布的异常检测模型

基本流程

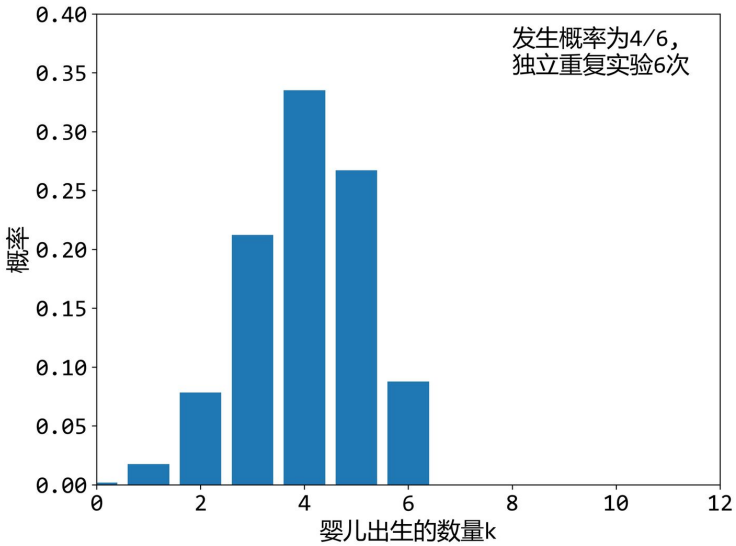
1. 为数据选择一个概率模型
2. 根据概率模型选择一个概率阈值
3. 计算观测到每个样本的概率
4. 将低于阈值的样本作为异常样本

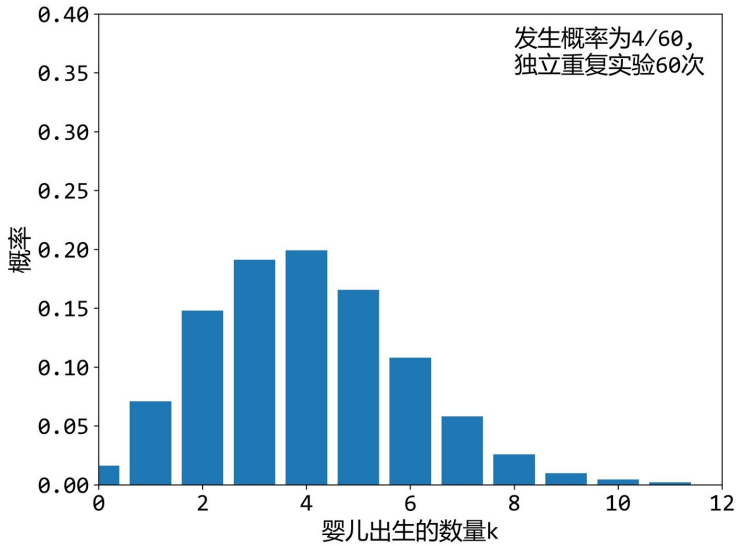
常见的概率模型：高斯分布，泊松分布（Poisson Distribution）

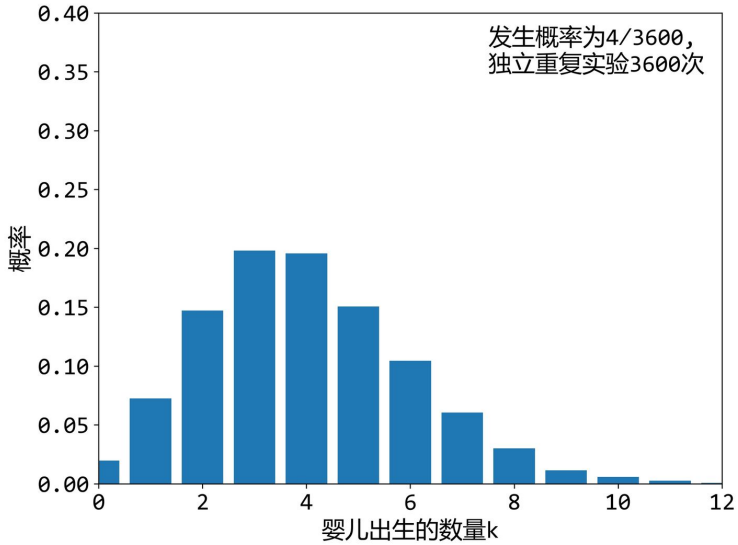
$$P(X = k) = \frac{e^{-\lambda}\lambda^k}{k!}$$



泊松分布







二项分布在时间间隔无限小情况下极限收敛至泊松分布

基于概率分布的异常检测模型

优点

1. 有许多成熟的概率模型可供选择
2. 在数据建模的同时直接得到异常分值
3. 能够对每个数据是否是异常的概率进行估计
4. 可用于不同的数据集

缺点

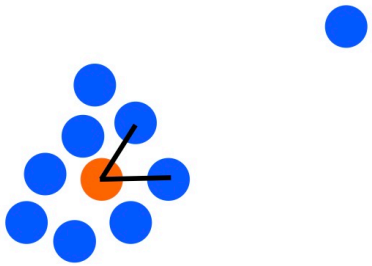
1. 依赖对数据的先验假设
2. 需要大量的数据进行假设估计
3. 高维数据难以用简单的概率模型进行建模

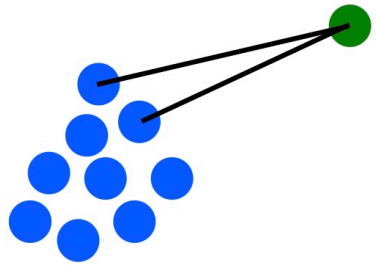


基于角度的异常检测

基本假设

1. 正常样本与其他样本的连线夹角种类比较多
2. 异常样本与其他样本的连线夹角相对单一





基于角度的异常检测

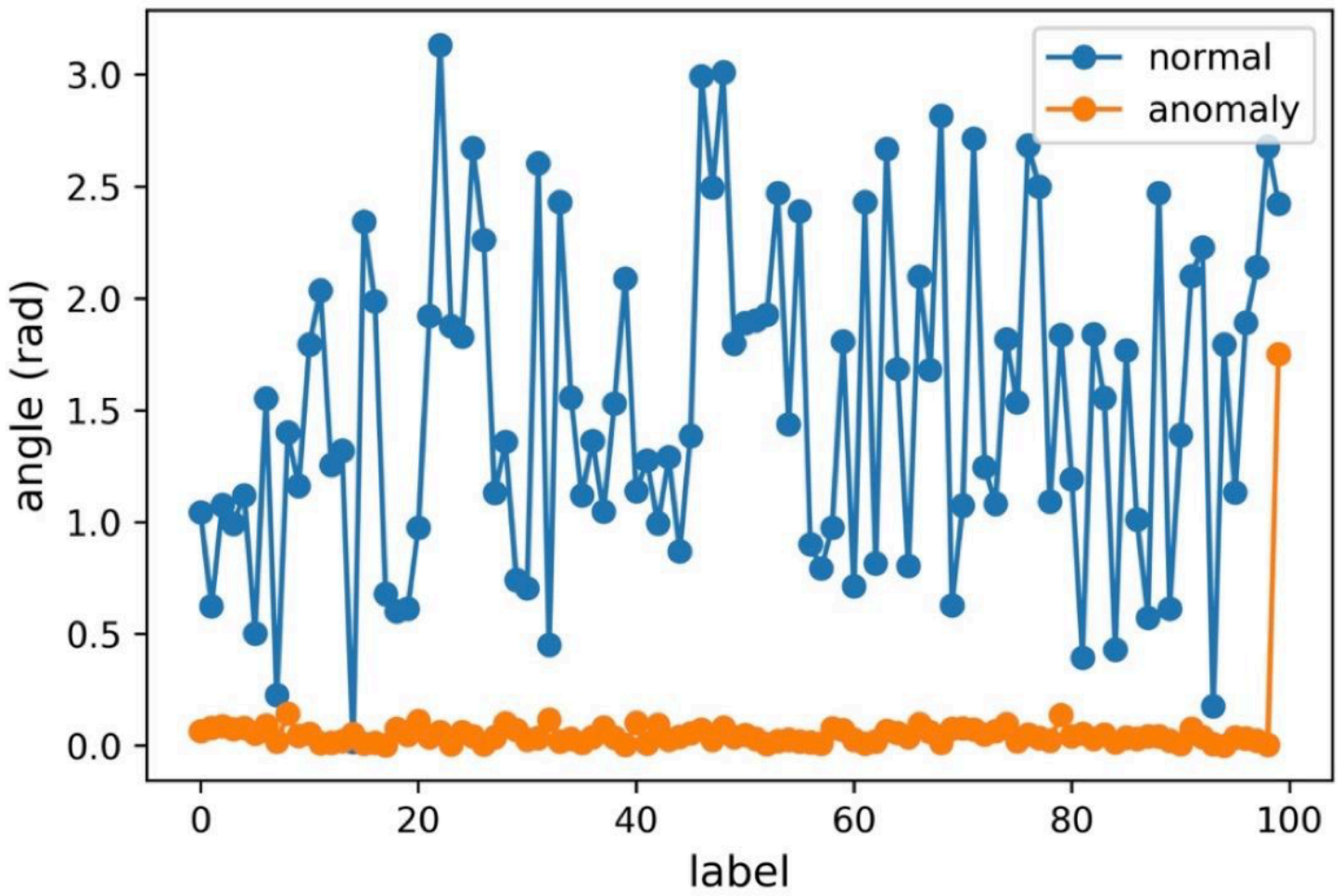


图: 基于角度的异常检测案例

单分类问题

常见的分类方法：

1. 单分类（只能定义正样本不能定义负样本）
2. 二分类（邮件分类）
3. 多分类（图像分类）

单分类与二分类的区别

在二分类问题中，训练集中就由两个类的样本组成，训练出的模型是一个二分类模型；而 OneClassClassification 中的训练样本只有一类，因此训练出的分类器将不属于该类的所有其他样本判别为“不是”即可，而不是由于属于另一类才返回“不是”的结果。



One Class SVM

模型假设

寻找一个超平面将样本中的正例圈出来，预测就是用这个超平面做决策，在圈内的样本就认为是正样本。

- 它和传统的基于监督学习的分类回归支持向量机不同，它是无监督学习的方法，不需要训练集的标签。
- 那么没有类别标签，我们如何寻找划分的超平面以及寻找支持向量呢？
- One Class SVM 这个问题的解决思路有很多。其中一种特别的思想叫做 SVDD(support vector domain description)，中文翻译为：支持向量域描述



One Class SVM

SVDD 的优化目标就是，求一个中心为 a，半径为 R 的最小球面：

$$F\left(R, a, \xi_i\right) = R^2 + C\sum_i \xi_i$$

ξ_i 是球体的松弛变量，使得这个球面满足：

$$\left(x_i - a\right)^{\mathrm{T}}\left(x_i - a\right) \leqslant R^2 + \xi_i \quad \forall_i, \xi_i \geqslant 0$$

满足这个条件就是说要把 training set 中的数据点都包在球面里。



One Class SVM

现在有了要求解的目标，又有了约束，接下来的求解方法和 SVM 几乎一样，用的是拉格朗日乘子法:

$$L\left(R, a, \alpha_i, \xi_i\right) = R^2 + C\sum_i \xi_i$$

$$-\sum_i \alpha_i \left\{R^2 + \xi_i - \left(x_i^2 - 2ax_i + a^2\right)\right\} - \sum_i \gamma_i \xi_i$$

注意 $\alpha_i \geqslant 0 \quad \gamma_i \geqslant 0$, 对参数求导并令导数等于 0 得到:

$$\sum_i \alpha_i = 1, \quad a = \frac{\sum_i \alpha_i x_i}{\sum_i \alpha_i} = \sum_i \alpha_i x_i$$

$$C - \alpha_i - \gamma_i = 0 \quad \forall_i$$

把上面公式带入拉格朗日函数函数，得到:

$$L = \sum_i \alpha_i \left(x_i \cdot x_i\right) - \sum_{i,j} \alpha_i \alpha_j \left(x_i \cdot x_j\right)$$



One Class SVM

sklearn.linear_model.SGDOneClassSVM

class sklearn.linear_model.SGDOneClassSVM(*nu=0.5, fit_intercept=True, max_iter=1000, tol=0.001, shuffle=True, verbose=0, random_state=None, learning_rate='optimal', eta0=0.0, power_t=0.5, warm_start=False, average=False*) [source]

Solves linear One-Class SVM using Stochastic Gradient Descent.

This implementation is meant to be used with a kernel approximation technique (e.g. `sklearn.kernel_approximation.Nystroem`) to obtain results similar to `sklearn.svm.OneClassSVM` which uses a Gaussian kernel by default.

Read more in the User Guide.

New in version 1.0.

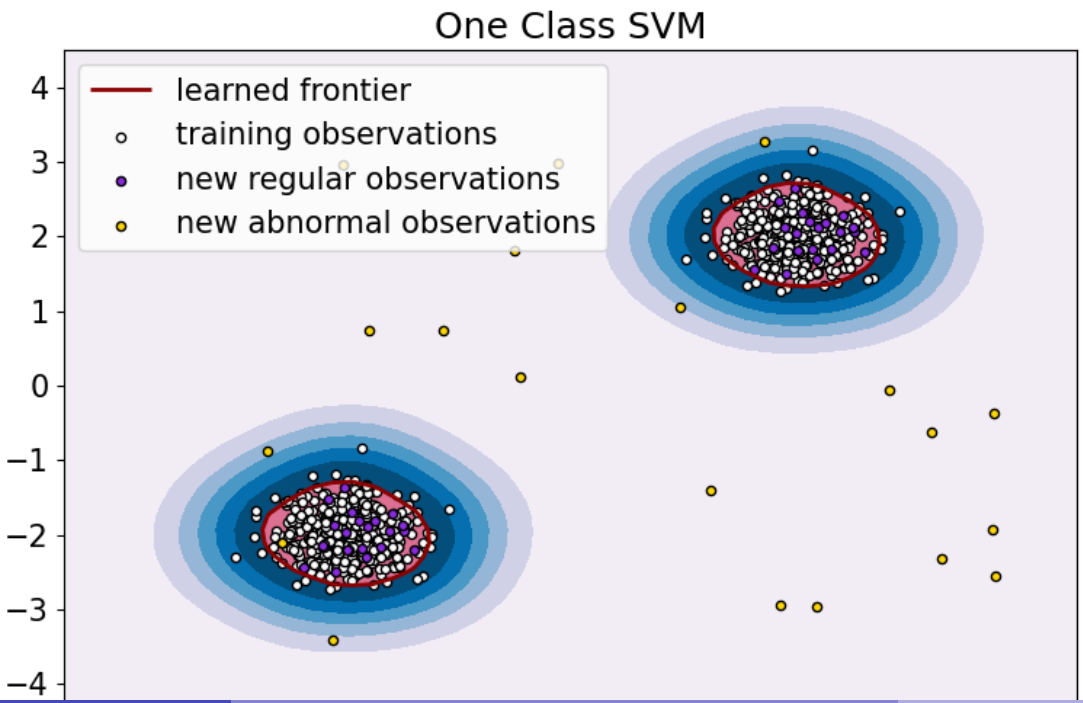


One Class SVM

One-Class SVM versus One-Class SVM using Stochastic Gradient Descent

This example shows how to approximate the solution of `sklearn.svm.OneClassSVM` in the case of an RBF kernel with `sklearn.linear_model.SGDOneClassSVM`, a Stochastic Gradient Descent (SGD) version of the One-Class SVM. A kernel approximation is first used in order to apply `sklearn.linear_model.SGDOneClassSVM` which implements a linear One-Class SVM using SGD.

Note that `sklearn.linear_model.SGDOneClassSVM` scales linearly with the number of samples whereas the complexity of a kernelized `sklearn.svm.OneClassSVM` is at best quadratic with respect to the number of samples. It is not the purpose of this example to illustrate the benefits of such an approximation in terms of computation time but rather to show that we obtain similar results on a toy dataset.



类不均衡对机器学习模型的影响

引申

把异常检测的概念泛化一下，所谓异常和正常的区分，其实是少数样本和多数样本的二分类问题。

以聚类中的 K-Means 为例：两轮迭代后，原本正确的聚类结果被破坏了

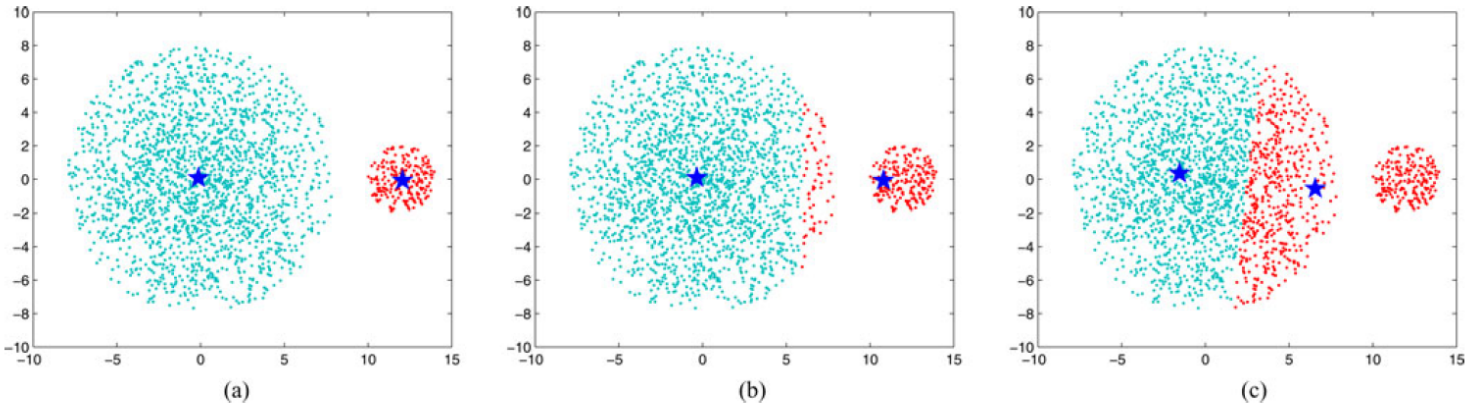


Fig. 1. "Uniform effect" occurs when the hard k-means algorithm is implemented. (a) Imbalanced data distribution before clustering. (b) Clustering result in the first iteration. (c) Clustering result in the last iteration.

类不均衡对机器学习模型的影响

思考

这个现象是偶然发生的呢？还是普遍存在的呢？

哲学原理

- 偶然是必然的表现形式，必然 = 规律
- 认识是阶段性的，实践是检验认识真理性的唯一标准



K-Means 的天生缺陷

$$X = \{x_1, \ldots, x_n\}, \qquad m_l = \sum_{x \in C_l} \frac{x}{n_l}$$

可以写出熟悉的形式化 K-Means 目标函数

$$F_k = \sum_{l=1}^{k} \sum_{x \in C_l} \|x - m_l\|^2$$

这个式子看起来很美丽，也很容易理解，似乎聚类目标函数的终极解决方案就该长这样。现在让我们站在类不均衡的角度揭开 K-Means 的天生缺陷。



K-Means 的天生缺陷

令

$$d(C_p, C_q) = \sum_{x_i \in C_p} \sum_{x_j \in C_q} \|x_i - x_j\|^2$$

基于它写出数据集中所有样本之间的距离之和：

$$D_k = \sum_{i=1}^{n} \sum_{j=1}^{n} \|x_i - x_j\|^2 = \sum_{l=1}^{k} d(C_l, C_l) + 2 \sum_{1 \le i < j \le k} d(C_i, C_j)$$

* 我们在为之后的步骤做铺垫。另外请注意，D_k 对一个数据集来说是常数。



K-Means 的天生缺陷

我们先来看二分类的情况：

$$D_2 = \sum_{i=1}^{n}\sum_{j=1}^{n}\|x_i - x_j\|^2 = d(C_1, C_1) + d(C_2, C_2) + 2d(C_1, C_2)$$

现在用 $m_l = \sum_{x \in C_l} \frac{x}{n_l}$ 代入

$$F_2 = \sum_{l=1}^{2}\sum_{x \in C_l}\|x - m_l\|^2$$

可以得到

$$F_2 = \frac{1}{2n_1}\sum_{x_i, x_j \in C_1}\|x_i - x_j\|^2 + \frac{1}{2n_2}\sum_{x_i, x_j \in C_2}\|x_i - x_j\|^2$$



K-Means 的天生缺陷

令：

$$F_D^{(2)} = -n_1 n_2 \left[\frac{d(C_1, C_1)}{n_1^2} + \frac{d(C_2, C_2)}{n_2^2} - 2\frac{d(C_1, C_2)}{n_1 n_2} \right]$$

加上刚才定义的：

$$D_2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \|x_i - x_j\|^2 = d(C_1, C_1) + d(C_2, C_2) + 2d(C_1, C_2)$$

代入刚才化简得到的：

$$F_2 = \frac{1}{2n_1} \sum_{x_i, x_j \in C_1} \|x_i - x_j\|^2 + \frac{1}{2n_2} \sum_{x_i, x_j \in C_2} \|x_i - x_j\|^2$$



K-Means 的天生缺陷

构造出：

$$F_2 = -\frac{F_D^{(2)}}{2n} + \frac{D_2}{2n}$$

觉得构造得太快了？如果不理解的话就请自己展开推导一遍。
再进一步：

$$\frac{2d\left(C_1, C_2\right)}{n_1 n_2} = \frac{d\left(C_1, C_1\right)}{n_1^2} + \frac{d\left(C_2, C_2\right)}{n_2^2} + 2\left\|m_1 - m_2\right\|^2$$

最终得到：

$$F_D^{(2)} = 2n_1 n_2 \left\|m_1 - m_2\right\|^2$$



K-Means 的天生缺陷

于是我们把 K-means 优化目标改造成这样：

$$F_2 = -\frac{F_D^{(2)}}{2n} + \frac{D_2}{2n}$$

其中的非常数项如下：

$$F_D^{(2)} = 2n_1n_2 \left\| m_1 - m_2 \right\|^2$$

*D_2 对于整个数据集而言是常数

- 结论：最小化 F_2 等价于最大化 $F_D^{(2)}$



K-Means 的天生缺陷

$$F_D^{(2)} = 2n_1n_2 \|m_1 - m_2\|^2$$

- 由于 $n_1 + n_2 = n$，如果不看后一项，那么该式的最大值当且仅当 $n_1 = n_2 = \frac{n}{2}$ 时取到，也就是说，K-Means 会强行往类别均衡的方向优化。
- 但事实上前一项和后一项是会相互影响的。大量实验表明，如果数据集中样本没有分得足够开，类之间的距离比较接近，那么类别均衡的优化方向将显著地主导整个式子的优化。但如果数据集中样本分得足够开，类之间的距离比较远，那么类别均衡的优化方向并不显著。
- 到底分得多开才算足够开，这个界限至今没有解析解。上述结论由大量实验得出。



Bonus

- 刚才是二分类的情况，请尝试推导多分类的情况，看看能否得出相似的结论。
- 向助教提交你的推导过程会有加分。不提交不会有扣分。



Bonus

- 所有形式类似的损失函数都存在这样的问题，那么有什么样的方法来应对类不均衡问题呢？

1. 算法层面：对类别不均衡的样本施加权重再计算最终的目标函数
2. 数据层面：对数据集进行预处理，用采样或其他手法降低数据集的不均衡性
3. 后处理：在已经得出分类结果后进行评估，对误分类的指标进行惩罚



1 背景介绍

2 基于统计的方法

3 有监督/半监督异常检测算法

4 基于距离的异常检测方法

5 基于密度的方法

6 基于聚类的方法

7 基于概率分布的异常检测模型

8 基于树的方法



孤立森林 Isolation Forest

Iforest 由南大周志华老师的团队在 2008 年提出，算法效果好，时间效率高，能有效处理高维数据和海量数据，在工业界得到了广泛的应用。

基本假设

在数据空间里面，分布稀疏的区域表示数据发生在此区域的概率很低，因此可以认为落在这些区域里的数据是异常的。异常数据由于跟其他数据点较为疏离，可能需要较少几次切分就可以将它们单独划分出来，而正常数据恰恰相反。



孤立森林 Isolation Forest

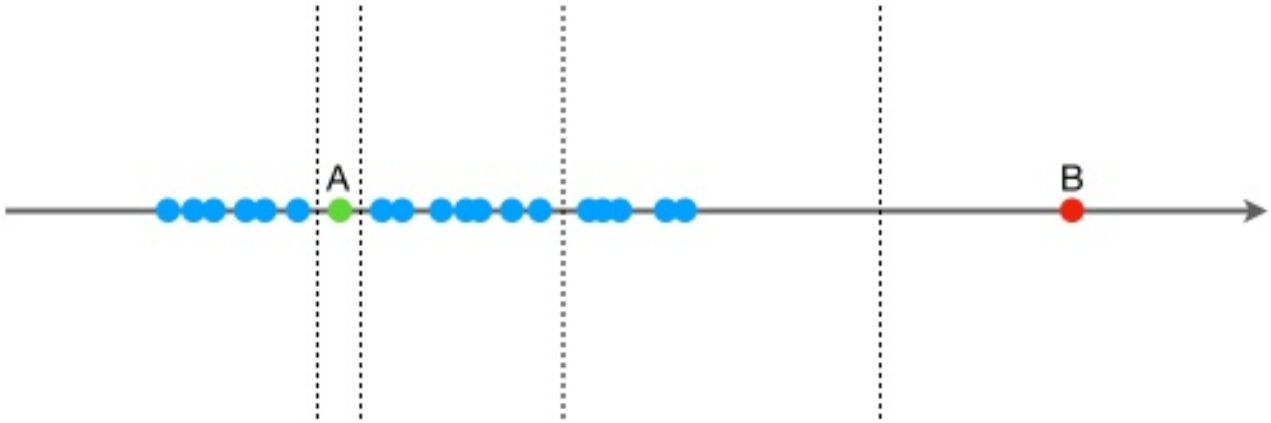
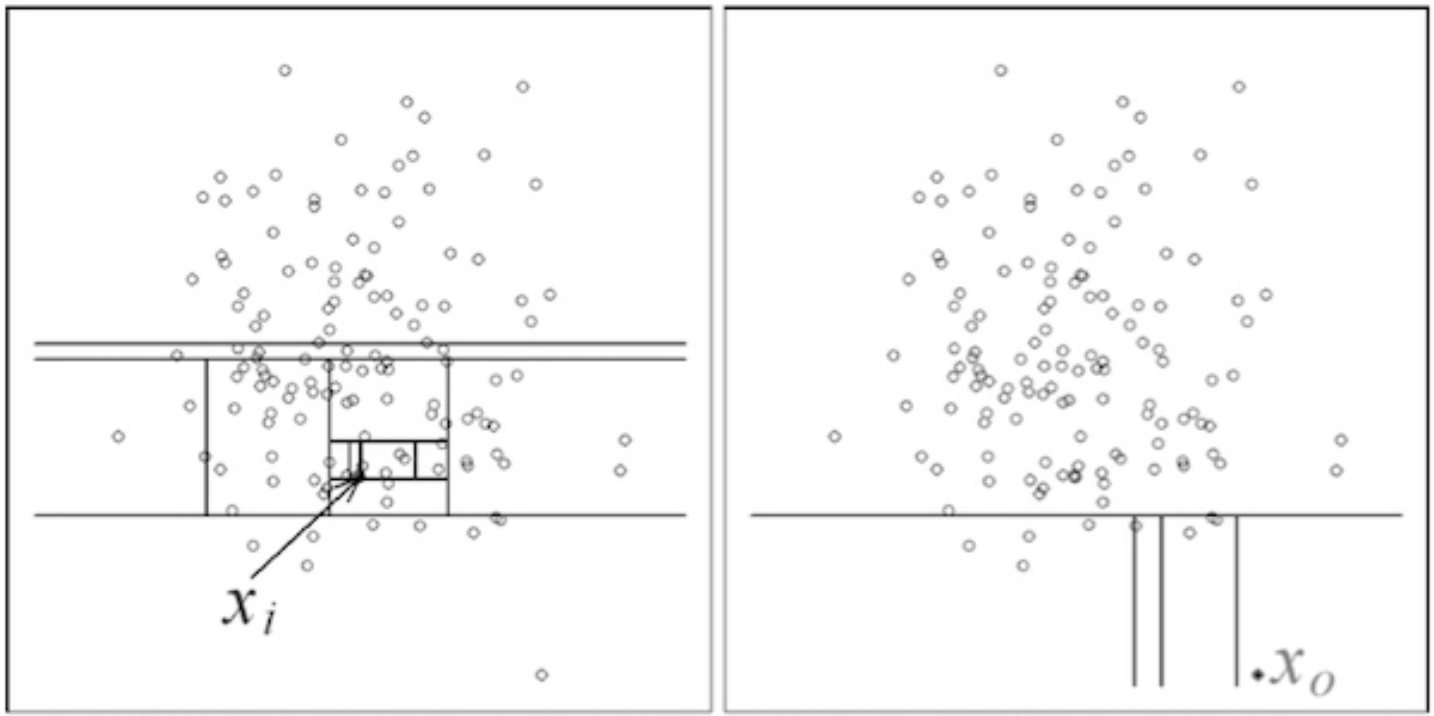


图: IForest 一维示例

1

[1] https://zhuanlan.zhihu.com/p/27777266

孤立森林 Isolation Forest



(a) Isolating x_i (b) Isolating x_o

图: IForest 二维示例

孤立森林 Isolation Forest

孤立森林的定义

iForest 由 T 个 iTree 组成，每个 iTree 是一个二叉树结构。

孤立树的定义

若 T 为孤立树的一个节点，T 存在两种情况：没有子节点的外部节点，有两个子节点 (T_l, T_r) 和一个 test 的内部节点。在 T 的 test 由一个属性 q 和一个分割点 p 组成，$q < p$ 的点属于 T_l，反之属于 T_r。

路径长度的定义

样本点 x 从 iTree 的根节点到叶子节点经过的边的数量



孤立树的生成过程

1. 随机选择一个属性；
2. 随机选择该属性的一个值；
3. 根据属性对每条记录进行分类，把属性小于阈值的记录放在左子节点，把大于等于阈值的记录放在右子节点
4. 递归构造直到满足以下条件：
   1. 传入的数据集只有一条记录或者多条一样的记录；
   2. 树的高度达到了限定高度；

Algorithm 2 : $iTree(X, e, l)$

Inputs: X - input data, e - current tree height, l - height limit

Output: an iTree

if $e \geq l$ or $|X| \leq 1$ **then**
  return $exNode\{Size \leftarrow |X|\}$
else
  let Q be a list of attributes in X
  randomly select an attribute $q \in Q$
  randomly select a split point p from max and min values of attribute q in X
  $X_l \leftarrow filter(X, q < p)$
  $X_r \leftarrow filter(X, q \geq p)$
  return $inNode\{Left \leftarrow iTree(X_l, e+1, l),$
    $Right \leftarrow iTree(X_r, e+1, l),$
    $SplitAtt \leftarrow q,$
    $SplitValue \leftarrow p\}$
end if

图：孤立树的生成过程



路径长度的计算

把叶子节点到根节点的路径 h(x) 长度来判断一条记录 x 是否是异常点（也就是根据 h(x) 判断 x 是否是异常点）

Algorithm 3 : $PathLength(x, T, e)$

Inputs : x - an instance, T - an iTree, e - current path length; to be initialized to zero when first called

Output: path length of x

1: **if** T is an external node **then**
2: return $e + c(T.size)$ $\{c(.)$ is defined in Equation 1$\}$
3: **end if**
4: $a \leftarrow T.splitAtt$
5: **if** $x_a < T.splitValue$ **then**
6: return $PathLength(x, T.left, e + 1)$
7: **else** $\{x_a \geq T.splitValue\}$
8: return $PathLength(x, T.right, e + 1)$
9: **end if**

图: 路径长度的计算



IForest 算法

$$S(x,n) = 2^{-\frac{h(x)}{c(n)}}$$
$$c(n) = 2H(n-1) - (2(n-1)/n)$$
$$H(k) = \ln(k) + \zeta, \zeta = 0.5772156649$$

Algorithm 1 : $iForest(X, t, \psi)$

Inputs: X - input data, t - number of trees, ψ - sub-sampling size

Output: a set of t *iTrees*

1: **Initialize** $Forest$
2: set height limit $l = ceiling(\log_2 \psi)$
3: **for** $i = 1$ to t **do**
4: $\quad X' \leftarrow sample(X, \psi)$
5: $\quad Forest \leftarrow Forest \cup iTree(X', 0, l)$
6: **end for**
7: **return** $Forest$

图: IForest 算法流程图



IForest 算法

优点

1. 线性时间复杂度
2. 每棵树单独建立，支持大规模分布式计算

缺点

1. 高维数据灾难，无法充分利用所有特征
2. 难以处理局部稀疏点



挑战与机会

挑战

1. 大多数方法都依赖人为假设
2. 数据的表征非常重要，但是这些方法并没有学习好的表征。
3. 二阶段算法难以取得最优解

机会

1. 为数据学习更好的表征
2. 捕获数据之间的复杂依赖关系
3. 端到端模型。



课程总结

1. 背景介绍
2. 基于统计的方法
3. 有监督/半监督异常检测算法
4. 基于距离的异常检测方法
5. 基于密度的方法
6. 基于聚类的方法
7. 基于概率分布的异常检测模型
   - One Class SVM
   - 类不均衡
8. 基于树的方法

