电 子 科 技 大 学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

硕士学位论文

**MASTER THESIS**



论文题目  **基于在线学习的异常行为检测**

学科专业 **计算机技术**

学 号 **201422060519**

作者姓名 **罗 钰**

指导教师 **傅彦 教授**

分类号 密级

UDC注1

学 位 论 文

**基于在线学习的异常行为检测**

（题名和副题名）

**罗钰**

（作者姓名）

指导教师 **傅 彦 教 授**

**电子科技大学 成 都**

（姓名、职称、单位名称）

申请学位级别  **硕士** 学科专业 **计算机技术**

提交论文日期 **2016.03** 论文答辩日期

学位授予单位和日期 **电子科技大学** **2016年 月 日**

答辩委员会主席

评阅人

注1：注明《国际十进分类法UDC》的类号

|  |
| --- |
| **RESEARCH AND APPLICATION OF PERSON INFORMATION MINING BASED ON TEXT ANALYSIS** |

**A Master Dissertation Submitted to**

**University of Electronic Science and Technology of China**

Major: **Computer Technology**

Author: **Luo Yu**

Advisor: **Prof. Fu Yan**

School: **School of Computer Science & Engineering**

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

作者签名： 日期： 年 月 日

论文使用授权

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

作者签名： 导师签名：

日期： 年 月 日

摘 要

**关键词：**

**ABSTRACT**

（待翻译）

Keywords:

目 录

[第一章 绪论 1](#_Toc476989961)

[1.1 研究工作的背景与意义 1](#_Toc476989962)

[1.2 国内外发展现状 2](#_Toc476989963)

[1.2.1 异常检测发展现状 3](#_Toc476989964)

[1.2.2 频繁模式发展现状 4](#_Toc476989965)

[1.2.3 在线学习发展现状 5](#_Toc476989966)

[1.3 本文的主要研究内容 6](#_Toc476989967)

[1.3.1 基于频繁模式的异常点检测 6](#_Toc476989968)

[1.3.2 在线学习的频繁模式异常检测 6](#_Toc476989969)

[1.3.3 难点 6](#_Toc476989970)

[1.4 论文的组织结构 7](#_Toc476989971)

[第二章 技术概述 8](#_Toc476989972)

[2.1 学习方式 8](#_Toc476989973)

[2.1.1 监督学习 8](#_Toc476989974)

[2.1.2 非监督学习 9](#_Toc476989975)

[2.1.3 半监督学习 9](#_Toc476989976)

[2.2 关联分析 9](#_Toc476989977)

[2.2.1 频繁模式、关联规则 10](#_Toc476989978)

[2.2.2 频繁模式挖掘算法 10](#_Toc476989979)

[2.3 分类 13](#_Toc476989980)

[2.4 聚类 13](#_Toc476989981)

[2.5 异常点检测 13](#_Toc476989982)

[2.5.1 异常点分类 14](#_Toc476989983)

[2.5.2 常用异常检测方法 17](#_Toc476989984)

[2.5.3 基于频繁模式的检测方法 17](#_Toc476989985)

[2.5.4 数据流的频繁模式异常检测 20](#_Toc476989986)

[2.6 分类中数据不平衡问题 22](#_Toc476989987)

[第三章 基于频繁模式的异常检测 24](#_Toc476989988)

[3.1 引言 24](#_Toc476989989)

[3.2 相关定义 24](#_Toc476989990)

[3.3 算法思想 25](#_Toc476989991)

[3.3.1 FPOF 25](#_Toc476989992)

[3.3.2 WCFPOF 26](#_Toc476989993)

[3.3.3 IWF-CFPOF 26](#_Toc476989994)

[3.4 算法描述 27](#_Toc476989995)

[3.5 算法流程 29](#_Toc476989996)

[3.6 实验与讨论 32](#_Toc476989997)

[3.7 小结 46](#_Toc476989998)

[第四章 在线学习的异常检测 47](#_Toc476989999)

[4.1 在线频繁模式挖掘 47](#_Toc476990000)

[4.1.1 基于抽样 47](#_Toc476990001)

[4.1.2 基于计数 47](#_Toc476990002)

[4.1.3 基于散列 48](#_Toc476990003)

[4.2 频繁模式近似计算 48](#_Toc476990004)

[4.3 时间衰减 51](#_Toc476990005)

[4.4 数据流FP-outlier计算 52](#_Toc476990006)

[4.5 实验和讨论 53](#_Toc476990007)

[第五章 结论与展望 57](#_Toc476990008)

[致 谢 58](#_Toc476990009)

[参考文献 59](#_Toc476990010)

[攻读硕士学位期间取得的成果 63](#_Toc476990011)

1. 绪论

# 研究工作的背景与意义

随着互联网的不断发展，科技的不断进步，信息的不断流通，人们产生的联系越来越紧密，我们从“信息时代”进入“数据时代”。又随着数据的加速产生和运输，我们进入了“云时代”。近几年大数据（big data）一词被不断地提起，它代表着互联网公司在日常运营中生成的、积累的用户网络行为数据，数据规模庞大到不能用G或T来衡量。来自Josh James的Data Never Sleeps 4.0报告显示，截止2016年底， Google每分钟处理约695万个单词，Twitter每分钟新产生大约10万条信息，Youtube每分钟产生用户新分享超过400小时的视频， Instagram每分钟产生大约240万条新信息，每分钟超过大约356万条文本信息。美国互联网数据中心指出，互联网上的数据每年将增长50%。移动互联网的火速发展无疑更加快了这一进程。与此同时，全球工业设备、汽车、数码传感器、检测仪、可穿戴设备也不断地在产生海量的信息。如何加工处理这些信息，从中提取有用信息，对我们无疑是一个巨大的挑战。

数据的膨胀必定会带来工业、科技、生活方式的革新。“大数据”渗透的领域越来越大，从电子商务到O2O，从个性化推荐到物流配送，从航空航天到医学，各行各业都在利用数据，提高其效率，推进其发展。与此同时， “大数据”也正在协助企业不断地发展新业务，创新运营模式。《纽约时报》2012年2月在一篇专栏中指出“大数据”时代已经降临，在商业、经济及其他领域中，决策将日益基于数据和分析而作出，而并非基于经验和直觉”。

在这个信息技术迅速发展的大数据时代，更加全面、系统和完整的数据也影响着人们的思维方式，给传统企业带来挑战。例如在传统零售行业，公司收集更多社交信息，深入理解商品的营销模式，区分高消费群体和高消费者，同时针对不同的客户使得业务服务更具有目标性。零售企业也可以监控客户的活动区域，决定货架的摆放，合理分配库存。利用“大数据”分析的经典案例-“啤酒和尿布”就是根据顾客大量的购买记录得出客户的购物习惯并作出相应决策。在电商业务中，公司通过大数据分析可以向客户提供差异化服务，根据顾客的浏览数据推荐顾客更需要或者更想买的产品。同时根据顾客的浏览习惯、年龄、地区等多种信息预测客户的需求，并提供相关服务。2009年，谷歌公司工程师在《自然》杂志上发表了一篇关于如何预测冬季流感的文章，根据用户大量的检测数据预测流感的趋势，只有很简单的模型就成功地预测了复杂的流感规模问题。同时，疾病控制和预防中心根据网民搜索分析全球范围内流感等病疫的传播状况[1](The Modeling of Global Epidemics)。在2014年世界杯上，谷歌公司使用英国体育数据提供商Opata Sports的数据和BigQuery（谷歌大数据引擎）工程师乔丹·提加尼开发的实力排行榜系统，评估了全球每个职业足球联盟过去多个赛季的情况，同时考虑了关注热情，用以预测世界杯16强的比赛胜负。其结果则是2014年世界杯16强全部成功预测。大量的事实显示，尽管大数据现在还在发展阶段，但是商业价值已经显现出来，也就是说基于数据做出的决策有助于公司产生更好的效益，做更好的决策，诞生许多新的商业模式。

数据带来机遇，同时也带来了更多挑战，例如对安全的威胁，数据集中出现的异常现象等等。这类在数据集中与众不同的数据，被Hawkins(1980)定义为异常。Hawkings认为这些异常点与其他数据差异太大，以至于使人怀疑这些数据并非随机误差，而是产生于完全不同的机制。例如在寒冷下雪的冬天，某天的温度数据却高于25摄氏度，显然这个温度数据就是一个异常。再例如，某信用卡卡主突然出现比平时购买了大得多的交易，且交易地点不在卡主居住地，那么该购买就是可疑的。异常检测和分析是数据挖掘中一个重要的方面，也是一个比较难的课题。在信息领域中，我们可以利用异常检测尽快发现网络中出现的异常，防止各类恶意攻击，保证信息的安全。在工业中，我们可以利用异常检测发现机器故障和产品损毁，以减少进一步的损失。不仅如此，在其他领域，例如电信和信用卡欺诈检测，贷款审批，药物研究，医疗分析，图像处理，气象预报，网络入侵检测，传感器、视频网络监控等，异常检测都具有广泛的应用。

高效的工具和分析方法对于处理各个领域中，快速增长的数据尤为重要。之前的数据挖掘通常是采用线下采集数据的方式，用整个输入-输入对组成的的历史数据来建模。用这样的方式训练出来的数据便可以用来预测新的输入数据。但是更多的时候，数据是以数据流的方式组成的。现在机器的内存要处理非常庞大的数据流是很困难的。同时由于这些数据并不完全具有价值，要保存全部的数据也是没有意义的。因此这种情况下只有在线处理比较合适。通常，模型的训练有两种方法，增长地持续不断更新或者用最近的一批数据来训练。

在异常检测的实际应用中，很多数据都是一组实时，快速且按序持续到达的，尤其在科学计算、网络安全、工业产生的数据。而在这大量的数据流中，异常点相对于正常点的占比却非常小。那么建立准确的模型对这些异常进行实时在线的检测、预警、追踪，在科学研究和行业应用中都有非常重要的意义。

# 国内外发展现状

许多人认为数据挖掘即是知识发现（KDD:Knowledge Discovery in Database），也就是从数据集中识别出有效的、新颖的、潜在有用的并可以被理解的模式的过程。由Han[2]提出的KDD过程如图所示，通常包括三个步骤：

1. 预处理。在应用常用数据挖掘算法之前对数据的处理工作，包括数据清洗，数据整合，数据选择和变换。
2. 数据挖掘。是最主要的步骤，通过建模提取数据模式。
3. 后处理。根据某种兴趣度度量结果好坏，识别出真正的有趣模式。并且通过可视化的知识表示技术，向用户提供知识并运用知识。



图1-1数据挖掘流程

## 异常检测发展现状

关于异常检测国内外已经有很多研究。Victoria J.Hodge和Jim Austin[5]在2004年发表了《A Survey of Outlier Detection Methodologies》，从计算机和统计学的角度，介绍了当代主流异常检测技术，并对比了其优缺点。大多数异常检测方法使用距离来度量点之间的相似性，然而在高维数据中，尤其是稀疏的高维数据，仅用点之间的距离去刻画相似性是没有意义的，因此高维数据的异常点检测更加复杂。Charu C.Aggarwal和Philip S.Yu[6]在2001年发表了《Outlier detection for high dimensional data》，讨论在高维度的数据集下常用的异常检测方法。以及在文本分析，欺诈检测，网络入侵检测等领域的应用。Manish Gupta和Jing Gao[7]在2014年发表《Outlier Detection for Temporal Data:A Survey》。该文总结了当下主流的异常检测方法，并着重关注异常检测在时间数据中的研究方法，例如数据流，时空数据，分布式数据流，时序网络和时间序列数据等不同的应用产生的数据。文章不仅指出了时间序列数据异常检测的重要性，更对比了当下和以前，静态数据和时间序列数据的研究方法。

异常检测在不同的领域也在不断地发展。入侵检测指检测网络上威胁计算机系统安全的恶意破坏行为，主要包括基于主机的异常检测系统和网络入侵检测。Forrest等提供了更好的入侵检测算法[8][9]。金融欺诈行为是国内外都广泛关注的一个问题，主要用以发现在银行、保险、电话，股票市场发生的犯罪活动。该类异常检测要求及时准确。这类犯罪活动通常会模仿正常行为的模式，伪装成正常点，因此加大了异常检测的难度。Donoho[3]提出了在内幕交易中的异常检测方法。异常检测在图像方面也有广泛的应用，常见的用以发现图像随时间而产生的改变或者在静态图像中出现的异常，包括卫星图像检测[4][10][11]和视频检测[12][13]。在文本中，异常检测常用于主题追踪、新话题的发现[14][15]。

## 频繁模式发展现状

关联规则挖掘首先由Agrawal，Imielinski和Swami[39]等提出，用以发现大量数据集中频繁出现的模式，或相关联和有相关关系的项集。一个流行的应用领域就是购物篮分析，通过搜索经常一起购买的商品集合，研究客户的购买习惯。关联规则首先要找出频繁项集，然后进一步产生形如的强关联规则。频繁项集挖掘已经有很多有效的算法。Agrawal，Srikant等人提出了Apriori算法[40]，该算法使用频繁项集性质的先验知识-非频繁项的超集也是非频繁项集对候选集进行剪枝，减少了计算复杂度。Han，Pie和Yin等提出了一种挖掘频繁模式而不产生候选集的方法-FP-Growth[41]。该方法通过构造FP树，使得不用再重复扫描数据集。它递归构造条件模式基，避免了高代价的候选集产生，从而高效地挖掘频繁模式项集。由于关联规则处理的数据量成指数倍增长，利用多处理器系统进行并行计算可以提高挖掘效率。Agrawal等人[11]提出了CD，DD，CaD三种并行算法.

在过去若干年中，频繁模式挖掘在不同的系统、技术、模型中不断地研究发展，应用在各个领域。特定的数据流管理系统和技术在网络交通管理[43][44]，科学控制过程[46]，传感器数据整合[47]，金融市场分析[48]，在线商务应用[49][50][51]，社团检测[52]，车辆监控[53]，在线声音识别系统[54]等等很多领域都有发展和应用。

随着频繁模式不断地发展，基于频繁模式的异常检测也被提出。Zengyou He[33]基于频繁模式的发现，提出了一种新的异常检测算法。该算法基于这样一个假设：通过关联规则发现的频繁模式反应了存在于数据集中的“通用模式”，那么包含极少这样的“通用模式”的数据对象就更有可能是异常点。也就是说，如果一个数据对象包含更多的频繁模式，也就意味着这个数据对象更有可能是正常点，因为该点符合大多数点的“共同模式”。Zengyou He定义了一种新的“频繁模式异常因子”-FPOF(Frequent Pattern Outlier Factor)，用来度量数据对象的异常程度，并通过FindFPOF算法来计算每个数据对象的FPOF值，发现异常点。同时论文对数据对象中不包含的频繁模式也有相应的描述，该描述从一定角度度量了该数据对象与该频繁模式的抵触程度。FindFPOF算法是采用Apriori算法来发现频繁模式，会产生大量的频繁项集。特别是，当最小支持度阈值设置得比较低或者数据集中存在较长频繁模式时尤其如此。例如一个长度为100的频繁模式，就会产生个频繁模式。调整最小支持度阈值可以减少这样的长频繁模式。而当最小支持度阈值设置得很高时，又无法区别正常点和异常点。闭频繁模项集（即：X是数据集S中的闭频繁项集，如果不存在X的真超项集Y，使得Y在S中与X具有相同的支持度计数，并且X满足最小支持度）就可以显著地减少频繁模式挖掘所产生的模式数量，而且能保持关于频繁项集的集合的完整信息。JiadongRen[34]就提出来了在闭频繁模式下的算法。论文定义了带权重的闭频繁模式异常因子WCFPOF(Weighted Closed Frequent pattern outlier factor)，并给出了计算每个事务的WCFPOF的方法。该算法认为，数据对象包含更多的闭频繁模式，就更有可能是正常点。Weiwei Zhang[35]则提出基于极大频繁模式（如果X是频繁的，并且不存在超项集Y使得并且Y在D中是频繁的）的异常检测算法。论文定义了极大频繁模式异常因子LFPOF(Longer Frequent pattern outlier factor)。并认为包含较多极大频繁模式的更有可能是正常点。

## 在线学习发展现状

在机器学习和数据挖掘研究领域中，在线学习是一个重要的组成部分。在线学习主要针对持续，大量按序到达的数据。由于数据不是一次性完全采集，数据集也很大，因此通常是将数据建模为一个随时间延续而无限增长的动态数据集合。数据在刚开始只能收集一部分，难以估计数据分布的动态进化特征和数据实际分布。同时，由于异常点本身的正负样本不均衡，使得基于机器学习和数据挖掘的算法需要改进才能应用到异常检测。

在过去数十年的研究中，对数据流的研究不断地发展，在线学习算法不断地被提出来并改进[16][19][20][21][22]，以适用于不同的应用平台。与传统静态数据模型不同，数据流是大容量的数据实时持续达到，这样大量数据是不可能完全存储下来的，同时大部分数据没有有用的价值，也就没有必要将所有的数据存下来。因此数据流的处理通常是在线扫描，实时处理。

在不断动态变化的环境中，数据分布是随着时间不断改变的，由此产生了概念漂移[17][18]。概念漂移描述了，给定输入的情况下，输出作为条件概率分布的变化，即使输入的分布并没有改变。一个典型的例子就是一个用户对网络新闻数据兴趣的变化过程。尽管网络新闻的内容保持稳定，但是用户感兴趣的文章分布却是不断变化的。在线学习就是不断对概念漂移进行反馈，在线更新预测模型。

在过去数十年的研究中，对概念漂移的研究不断地发展，在线学习算法有而不断地被提出来并改进。尽管如此，这个研究领域的复杂度和广度使得许多文章只是针对某一特定领域提出相关算法。而本文根据异常检测和数据流的共同特点，提出一类更通用且适用于数据流的算法。

Zengyou He将FindFPOF方法应用到数据流环境中检测异常点。Zhou xiao-yun[37]提出了一种与在静态数据集中完全不同的异常检测方法，该文也强调了维度在异常检测中对准确度的影响并提出了一个新的异常度量方法。该方法基于的假设同FindFPOF相同：数据对象包含的频繁项集越多，这个点就越可能是正常点，否则该点更有可能是异常点。Xianghong Tang[38]研究在数据流中采用的滑动窗口技术，并提出了基于频繁模式的数据流异常检测。该算法基于的假设是：事务在一定的滑动窗口内包含越多的与该事务有越少冲突的频繁模式，则该事务更有可能是正常点。反之，若该事务在一定的滑动窗口内包含越少的与该事务有越多的冲突的频繁模式，则该事务更有可能是异常点。

# 本文的主要研究内容

## 基于频繁模式的异常点检测

在过去几十年，许多异常点检测技术不断更新。近些年，基于频繁模式的异常检测越来越多的被人们关注。本文则研究基于频繁模式的异常检测。该方法基于这样一个假设：通过关联规则发现的频繁模式反应了存在于数据库中的“通用模式”，那么包含极少这样的“通用模式”的数据对象就极有可能是异常点。也就是说，如果一个数据对象包含更多的频繁模式，也就意味着这个数据对象更不可能是异常点，因为该点符合大多数点的“共同模式”。同样，不被包含在数据对象中的频繁模式可以用来描述该频繁模式与数据对象之间的冲突情况。文中提出了一种基于词频的带权闭频繁模式异常检测因子，考虑了在正负样本情况下，各类频繁模式对数据对象的贡献，通过该特征转换，更好的区分开正负样本点。

## 在线学习的频繁模式异常检测

目前国内外学者针对数据流的异常检测已经提出了一些新的思路和方法，主要包括改进静态方法使其适应数据流和增量学习、在线集成学习。在数据流上进行频繁模式挖掘也被陆续提出。尽管基于数据流的算法形式各异，但是都需要解决无限的数据和有限的存储空间之间的冲突。随着数据流的不断的到达，频繁项集有也发生了改变，原来不频繁的项集可能变得频繁，而原来频繁的项集可以变得不频繁。本文提出了根据时间动态计算频繁模式的方法，并在一定的滑动窗口内，根据当前的频繁模式计算每个数据对象的异常因子。

## 难点

总体来说，异常点是那些与定义的正常模式不符合的点。挖掘异常点的方式通常根据不同的应用场景差别很大。一种最直观的方法就是定义一个正常点的区域，那些不被包含在这个区域里面的点，就是发现的异常点。常见的方法有聚类，分类，密度估算等。然而由于异常点所具有的特点，这类方法面临诸多挑战。

1）定义一个正常点区域包含所有正常模式是非常困难的。更何况，正常点和异常的边界通常并不是很清楚，即靠近异常区域的点也可能是正常点。

2）部分异常点来自于恶意攻击，这些恶意攻击为了避免暴露，通常会伪装成普通模式，同时正常数据也可能由多个机制/统计过程产生，使得正常模式更难以定义。

3）行为的不断变化，正常模式随着时间以不可预见的方式变化，即概念漂移(concept drift)。现在定义的正常模式未必适用于将来的模型。例如一个人的消费模式在学生时代和白领阶段是不相同的。在白领阶段的消费模式与学生时代很不相同，但这些却不能定义为异常点。

4）应用场景不同，异常点的确定性定义也不同，采用的数据集和算法均不相同。例如在医学上，很小的偏差—体温或血压，就是异常的，但是在其他领域，例如信用欺诈监测，一点波动是正常的。因此异常检测算法通常针对于某一特定领域，无法直接套用。事实上大多数现有的异常检测方法都只针对一类特殊问题，这与数据的其他因素直接相关，例如是否是标签数据集，数据的格式，待检测数据类型等。

5）异常模式的直接定义通常是困难的，想要获得所有的异常模式数据是不可能的，同时异常点的定义隐含了异常点是极少数点，因此我们很难获得均衡的数据集。

6）数据集中噪音的干扰使得异常点更难以发现。而噪音又不可避免地存在于许多应用收集的数据集中。低质量的数据和噪音的干扰增加了异常检测的难度。噪音干扰了数据，模糊了正常数据和异常点之间的差距，甚至掩盖异常点，使得异常点的发现更难。

7）当数据是以流的形式组成时，由于数据量的巨大和处理器的限制，必须及时在线处理。随着时间的变化，模型发生概念漂移，传统预测模型需要重新学习。

8）维度诅咒。随着数据维度的升高，计算复杂度和模型的拟合能力都受到限制。而现实数据的维度又是在不断上升的。

# 论文的组织结构

第一章绪论部分阐述了本文的研究背景和意义。然后从国异常检测、频繁模式研究、在线学习研究等三个方面分别介绍了国内外研究现状，并简单概括了当前频繁模式在异常检测方面的研究和应用。

第二部分主要介绍了论文需要用到的理论基础和技术。首先对机器学习方式做了简单的介绍。其次介绍了本文最主要的基础技术-频繁模式挖掘，并详细介绍了常用频繁模式挖掘算法。接着介绍了常用分类和聚类算法，用于频繁模式的特征转换的下一步-模型选择。然后详细介绍了常用的异常点检测方法。最后分析了在数据不平衡问题中数据分类的困难。

第三部分提出了基于逆向词频的闭频繁模式异常因子IWF-CFPOF，并给出算法描述和流程。然后通过与其他算法的对比试验，进一步分析该方法的优缺点。

第四部分是基于频繁模式挖掘方法在在线学习领域的改进。我们通过时间窗口仅保留当前一段时间内的数据，根据以前的频繁模式和现在的数据，共同决定当前的频繁模式项集。然后介绍了时间衰减对频繁模式的影响，进一步调整以前频繁模式对当前时间数据对象的影响。接着给出了数据流FP-outlier计算方法，并分析在公开数据集下实验效果。

第五部分是对论文的总结与展望。主要包括论文的主要工作，创新点，不足之处以及对今后工作的展望。

1. 技术概述

# 学习方式

根据数据类型的不同，对一个问题的建模有不同的方式。通常分为监督学习，非监督学习和半监督学习。

## 监督学习

在监督学习过程中需要训练数据集，且每组训练数据有一个明确的标识或结果。例如把银行贷款申请划分为安全或危险，将邮件划分为垃圾邮件或非垃圾邮件，手写体划分等。监督学习利用训练数据集学习一个模型，再用模型对测试样本进行预测。监督学习建立一个学习过程，将预测结果与训练数据的实际结果进行比较，不断调整预测模型，直到模型的预测结果达到一个预期的准确率。常见的监督学习算法有逻辑斯蒂回归(Logistic Regression)，支持向量机(SVM)等。



图2-1监督学习流程

## 非监督学习

在某些应用中，没有标记训练数据集，直接对输入数据进行建模，自动从范例中找出潜在的类别规则，推断出数据的一些内在结构和特征。常见的算法如关联规则Apriori，聚类等。图2-2展示了非监督学习中的聚类。

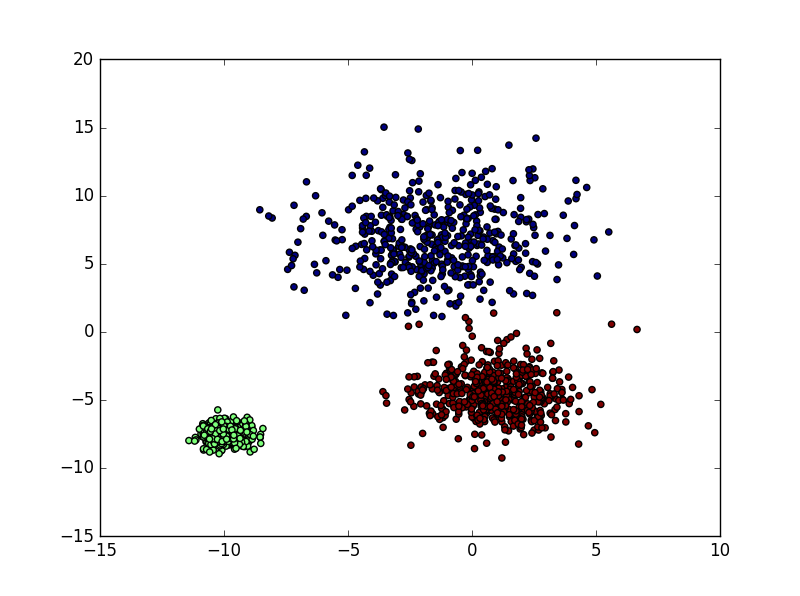


图2-2 非监督学习

## 半监督学习

在许多应用中，我们只能标记部分数据，且这类标记的实例数据通常很少。这类学习模型可以用来预测，但通常需要先学习数据的内在结构以便合理的组织了数据来进行预测。常见的应用如异常检测，只有少量正常和离群点实例被标记，但大多数的点是无标记的，尤其是异常点，要标出所有的异常点几乎是不可能的。少量被标记的离群点不太可能代表所有的离群点。因此仅基于少量被标记的离群点而构建的离群点模型效果并不会很好，为了提高离群点检测的质量，可以采用无监督模型学习。

# 关联分析

在大型事务或关系数据集中频繁出现的模式（项集、子序列或子结构）之间存在着某种有趣的关联或相关性，例如著名的购物篮分析，如果顾客在一次超市购物时买了面包，那么他极有可能会购买牛奶。

分析每一项记录，得到反应数据项频繁关联或同时出现的模式。这类模式可以用关联规则（association rule）的形式表示：

(2-1)

其中，规则的支持度(support)和置信度(confidence)是规则的两种度量方式，分别表示所发现规则的有用性和确定性。 (2.1)中支持度为1%意味着在给定数据集中有1%的事务表明面包和牛奶被同时购买，置信度69%则说明在购买面包的顾客中有69%也购买了面包。满足最小支持度阈值和最小置信度阈值的关联规则被称为强关联规则，挖掘给定数据集的强关联规则是数据挖掘任务中一个重要的部分。由于关联分析体现了数据中内部的联系和规律，因此它对分类、聚类、异常检测和其他数据挖掘任务也有帮助。

关联分析中频繁地出现在数据集中的模式成为频繁模式，它可以是一个频繁项集（频繁出现在数据集中的点的集合）、一个序列模式（时间序列或其他序列数据中频繁出现的子序列）、一个子图、子树或子格。通常，关联规则分析包括两个步骤：1）找出满足最小支持度和最小置信度阈值的所有频繁模式；2）根据频繁项集产生强关联规则。而步骤2）的复杂度远远低于步骤1），因此，挖据频繁模式是导出关联分析至关重要的一步。

## 频繁模式、关联规则

设是项的集合，给定一个事务数据库，其中的每一个事务*t*是中一组项目的集合，即，每一个*t*有一个唯一的标识符tid。对于项集且*，X*包含的项的个数称为*X*的长度，包含*k*个项的项集称为*k-itemset*。当时，称*t()*包含*X*。对于项集且，且*，*则关联规则是形如的蕴含式：

(2-2)

(2-3)

关联规则在数据库*D*中成立的条件是：①具有支持度*s*，其中*s*是*D*中事务包含*A**∪B*（即集合*A*和*B*的并）的百分比，即事务数据库*D*中至少有*s%*的事务包含*X∪Y*。②具有置信度*c*,其中*c*是*D*中包含*A*的事务同时也包含*B*的事务的百分比，即事务数据库*D*中包含*X*的事务至少有*c%*同时也包含*Y*。

## 频繁模式挖掘算法

关联规则的挖掘就是在事务数据库*D*中找出具有用户给定的最小支持度(*min\_sup*)和最小置信度(*min\_conf*)的关联规则。而关联规则的主要任务是找出给定数据集的频繁模式。挖掘频繁模式的主要挑战来自于频繁模式挖掘常常产生大量满足最小支持度阈值的项集，随着数据量的增大和最小支持度阈值的减小情况更加明显。

有很多算法用于频繁模式挖掘[25] [31][32]，大致可以分为两类：①候选集生成和检查，例如基于反单调性的Apriori ②递归拆分数据库为子数据库并挖掘，例如FP-growth。

1. Apriori 算法

Apriori算法是Agrawal和R.Srikant于1994年提出的[25]，基于频繁项集的先验知识，即频繁项集的所有非空子集也一定是频繁的。同时使用一种称为逐层搜索的迭代方法。首先通过扫描数据库，累计求出频繁1-项集，该集合记为，然后通过找出频繁2-项集，记为，即由满足条件的*k*项集产生候选集并扫描数据库，找出满足条件的*k+*1项集。具体做法分为2步：

1. 连接步：通过与其自身连接，产生候选集。中两个元素可以连接合并的要求是这两个元素仅有一个项是不同的，其余（*k*-2）个项相同。例如项集与可以连接并产生，而与则不能连接。
2. 剪枝步：候选集包含了非频繁项集，但是频繁项集都在中。通过扫描数据库*D*，计算中每个候选项出现的次数。累计次数大于等于最小支持度的项集便是频繁模式。对于小于最小支持度的便可以直接剪枝。此时的依据是Apriori先验知识。

Apriori通过限制候选集产生并发现频繁项集。算法使用频繁项集性质满足频繁项集的所有非空子集也一定是频繁的这一先验知识来压缩搜索空间，但是算法效率仍然不高。主要原因包括：

1. 由频繁项集自身连接产生项集，每次增加一个频繁项，判断连接条件却需要时间复杂度，其中*n*为频繁项集的个数。假设有个频繁1-项集，将会产生5000个候选频繁2-项集，如果其中有一个长度为100的频繁项集，则将产生大约个候选项集。
2. 每次生成新的频繁项集都需要扫描数据库。为了得到频繁-*k*项集，需要扫描数据库*k*次。重复扫描数据库的每一个事务来计算候选集的支持度开销很大。

因此基于Apriori算法已经提出许多变形，旨在提高原算法的效率。①基于散列的技术[26]能压缩候选集的数量，散列方法构建了桶地址，桶计数和桶内容三项。桶地址为项集的Hash散列映射值，桶计数为这个地址的项集的总数目，桶内容则为候选项集。按照桶计数，剪掉不符合最小支持度的项集。基于散列的方法能显著的减少候选集的数量，尤其是频繁2-项集。事务压缩，进一步压缩迭代扫描的事务数；②划分，将事务数据库进行划分以便更高效的找出候选项集[27]。③抽样[28]，该算法用于克服数据库太大而无法一次装入内容的I/O局限性[30]。选取给定事务数据集的随机样本S，并在随机样本S上搜索频繁项集。这种方法必然会损失一部分精度，因此可以根据经验选择较低的支持度，并扫描数据库检验频繁项集是否有遗漏。

1. FP-growth算法

由于Apriori算法仍然需要产生大量的候选集，而且需要重复扫描数据库，通过模式匹配检查计算候选集计数，因此算法的开销仍然很大。由Han, Pie和Yin等人提出了一种挖掘频繁模式而不产生候选集的模式增长方法[29]。频繁模式增长（Frequent-Pattern Growth，即FP-growth）方法，首先将代表频繁项集的数据库压缩到一棵频繁模式树（FP-tree）该树仍保留项集的关联信息；然后把这种压缩后的数据库划分为一组条件数据库（即一种特殊类型的投影数据库），每个数据库关联一个频繁项或“模式段”，并分别挖掘每个条件数据库。主要由两步构成。首先构造FP-Tree，然后在FP-Tree上递归构造条件模式基产生频繁模式。

FP-growth分两次扫描数据库就可构造FP-Tree，之后便不需数据库，而仅在FP-Tree上挖掘频繁模式。第一次扫描数据库，导出频繁1-项集的集合，并按每个项的支持度计数递减排序；在创建标记为“*null*”的根节点之后第二次扫描数据库。每个事务都按照排序后的次序处理，并对每个事务创建一个分支，沿共同前缀上的每个结点的计数加1，或创造新的分支。

FP-growth的改进N-lists[31]，Nodesets[32]，基于矩阵的关联规则挖掘等算法。基于矩阵的FP-Tree挖掘方法[33]由Zhenyu Liu等人提出来。该方法首先通过已经建好的FP-Tree对每一个1-频繁项构造局部频繁项集矩阵矩阵，然后通过这个频繁1-项集矩阵挖掘频繁模式。该方法仅需2次扫描数据库。当构建矩阵完成后，也不需要再次递归遍历FP-Tree以及构建（子）频繁模式树。通过对矩阵的位运算大大提升了频繁模式挖掘的效率。

1. 垂直和其他挖掘方法

与Apriori及FP-growth等水平数据格式不同，垂直数据格式使用项-TID集。首先通过扫描一次数据集，把水平格式的数据转换成垂直格式，项集的支持度计算等于项集的TID集长度。然后根据先验知识使用频繁*k*项集的TID的交集来构造并计算候选*k*+1项集。此算法不需要扫描数据库。

其他频繁模式挖掘的其他算法包括图挖掘方法[34]，FIAST[35]，ILLT[36]，Hyper-Structure[37]等。

# 分类

分类是数据挖掘中一种重要的数据分析方法。这种把数据按照某种方式预测其标号的模型称为分类器。例如，我们可以把银行贷款申请数据划分为“安全”和“危险”，医疗上根据病人症状划分为“生病”或“不生病”，工业上商品划分为“合格”和“不合格”。像上述标签为两类的称为二分类。其他如手写体识别，方案A，方案B，方案C等多个标签的称为多分类。

分类一般包含两个阶段，即“学习阶段”和“分类阶段”。学习阶段用的数据称为训练集，包括特征*X*和一个属性标签*Y*，在这个阶段，通过模型分析数据集的特点，找出数据集内部规律，为每个类别构建一个用以描述的模型。在分类阶段，利用学习到的每个类别的模型，对数据进行分类，并验证效果。由于同时使用了特性*X*和标签*Y*，这类方法也成为监督学习。分类是数据挖掘领域中应用极其广泛的主要技术之一，已有很多算法可用于分类。常见的分类算法有：

1. 逻辑斯蒂回归（Logistic regression）
2. 决策树（Decision Trees）
3. 朴素贝叶斯（Naïve Bayes）
4. 支持向量机（Support Vector Machines）

# 聚类

# 异常点检测

异常点是指在数据集中不符合正常模型定义的点。Hawkins在1980年曾给出异常点的定义：一个异常点通常距离其他观察点很远，就好像这类点是由不同的机制产生。如图1-2所示。三角形所示的点远离正常点（圆形）。

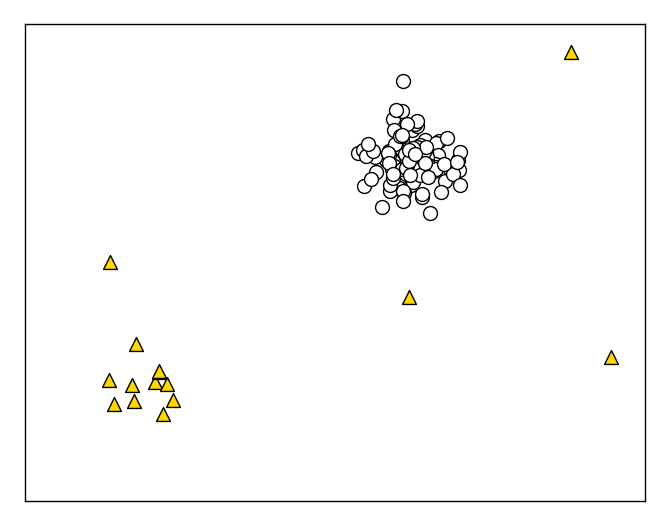


图2-3异常点

异常检测的研究已经有很长一段时间了，通常用于检测或及时发现异常。异常检测通常来源于机器故障，人工失误，欺诈行为等。这些异常可以被尽快的检测出来，以避免带来更多的危害。通过数据异常分析往往能够发觉未知的机会或风险。在金融方面，通过对金融数据进行特征化分析和比较分析建立模型，对异常点分析从而探测出异常资金量的转移或其他行为，进而分析金融欺诈和洗钱等行为。在生物医学方面，大量的研究集中在DNA的数据分析之上，人类约有100000个基因，每个基因通常由成百个核苷酸按一定次序组织而成。核苷酸的不同次序和序列可以形成不同的基因。从中找出异常的基于序列模式，分析可能治病的基因，开发针对某种疾病不同阶段的治疗药物等对人类具有重大意义。因此异常检测在机器学习中是一个研究的主要领域，大量的算法用于异常点的检测。

## 异常点分类

异常点的形式有很多种，例如，在欺诈监测中信用卡被盗用，用户的消费行为会发生变化，通过分析异常消费模式发现信用卡滥用。在公共医疗领域，一种特殊疾病的发现，能通过频率，周期等方面的异常及早发现。在工业上，来自传感器的数据可能因为操作、系统或者机器的故障，出现差错，通过这类异常点的检测能及时发现这类差错。异常点产生有很多原因。恶意行为，例如信用欺诈，网络入侵；机器故障，例如系统崩溃，断电；操作误差，记录误差，操作违规等。异常点与噪音相关，但是不同于噪音。噪音通常来自于传输过程中周围各种干扰对数据产生的影响，这类信息不仅对分析没有帮助，而且还阻碍了数据分析。现在通常在数据分析之前根据需要用分箱，聚类，计算机和人工检查结果，回归等方法去除噪声。

异常点的类型很多，通常分为[24]：

1. 全局离群点

如果一个点显著地偏离数据中的其余对象，则这个点是全局异常点。这是最简单的一类异常点，且大部分的研究和方法都旨在找出全局离群点。图2-3显示了全局离群点。

在许多应用中，全局离群点的检测非常重要。例如，在股票交易审计系统中，不遵守常规的交易与其他正常交易行为差别较大，可能被视为全局离群点，应当暂停成交。再比如，信用卡欺诈检测中，训练数据集是用户信用卡交易记录和每次交易总金额，某一次的交易总额远远超过了该用户普通消费水平，对该用户来说这个点就可能是异常点，可能存在信用卡被盗刷的风险。

1. 情境离群点(Contextual Anomalies)

给定数据集中，一个数据对象关于特定情境显著地偏离其他对象，则属于情境离群点。情境离群点又称为条件离群点，因为它们条件地依赖于选定的情境。因此在这情境异常点检测中，情境需要作为问题的一部分进行说明。情境异常检测的数据对象需要考虑到以下两种属性：

* 1. 情境属性。数据的情境属性定义数据所处的情境。例如，在空间数据中，经度和维度是情境属性。在描述温度的数据中，时间和地点属于情境属性。
  2. 行为属性。数据的行为属性决定了数据的非情境属性，用于评估对象关于它所处的情境是否是离群点。在一个描述股票的数据中，股票的涨跌就是行为数据。

在情境离群点检测中，与全局异常点检测最大的不同就是，情境离群点与数据对象所处的情境有关。某一数据对象的行为属性在某种情境下可能是异常点，但是在另一情境下可能就不是异常点。这类异常点是局部异常点的推广。局部异常点指，在基于密度的检测方法中，如果数据集中一个点显著地偏移局部区域的密度，那么这个点很可能就是局部离群点。如图1-4中的t1点，显著地与周围点不同。但整体来看，t1的取值也是有可能的。因此，t1是情境异常点。

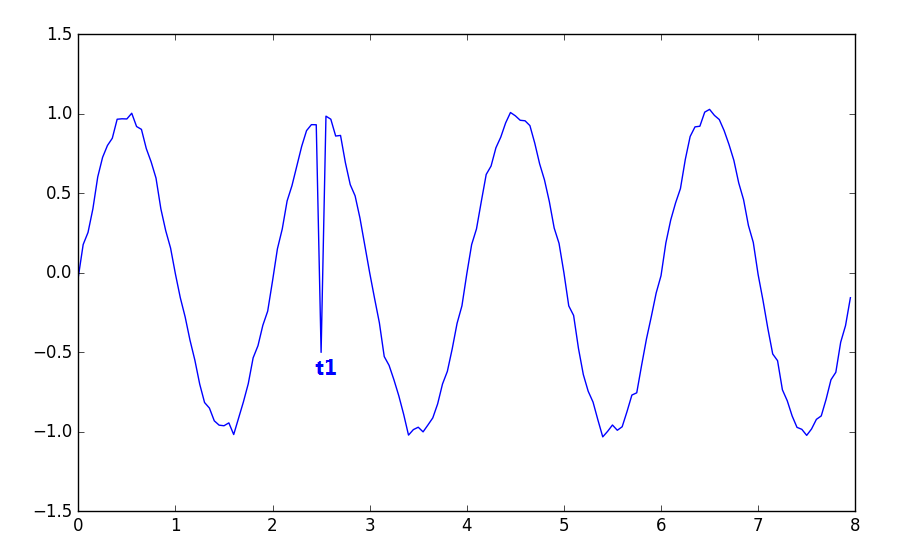


图1-4 情境离群点

1. 集体异常点

如果数据对象的一个子集作为一个整体显著地偏离了整个数据集，那么该数据子集就形成了一个集体异常点。而子集中的每一个独立点对于全局可能并不是异常点，而且每一个独立点对于该子集来说也不是异常点。

集体异常点常出现在序列数据、图和时空数据中。全局异常点可能出现在任何数据集中，但是集体异常点只能发生在相互管理的数据集中。因此，在集体异常点中，不仅必须考虑个体对象的行为，还需要考虑对象族群的行为。为了检测这类异常点，还需要关于对象之间联系的背景知识，如对象之间的距离或相似测量方法等。

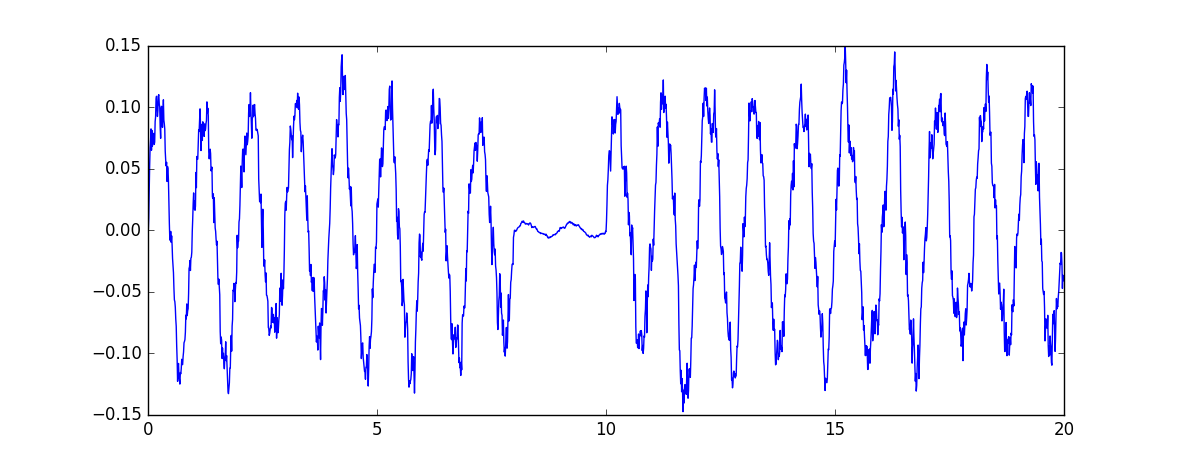


图1-5集体离群点

## 常用异常检测方法

与传统的发现数据内在模式的数据挖掘方式不同，异常检测主要发现与其他大部分数据均不相同的特殊点。当前常见的异常检测方法包括基于密度的方法，基于方差的方法，基于模型的方法，基于距离的方法及其他一些方法。

基于密度的方法比较点与它周围点的密度。该算法基于这样的假设：如果一个点与它周围点的密度相近，则这个点更接近于正常点。如果一个点的密度与周围点很不相同，则该点更类似异常点。该方法常用于局部异常点检测。基于方差的检测方法认为异常点由于与普通点差别很大，因此，异常点的方差比正常点要大。因此，通过比较方差来发现异常点。基于模型的方法包括以统计为基础的各类方法，有聚类、分类等各种算法。基于距离的算法通常认为正常点距离它的邻居很近，而异常点距离邻居很远。在该方法中，如何计算距离是一个难点。

## 基于频繁模式的检测方法

1. FPOF(Frequent pattern outlier factor)

Zengyou He [39]提出了一种基于频繁模式的异常检测算法。因为频繁模式是反应大数据内部的“普通模式”，即“共同特征”。因此包含更多“共同特征”的数据更倾向于属于普通点，而越少的包含“共同特征”就越可能属于异常点。Zengyou提出了一个新的度量标准用以描述异常点：

定义事务数据集，其中的每一个表示一个事务，每一个事务包含若干个来自项集*I*的项。给定一个最小阈值*minisupport*，所有的频繁模式被记为：

(2-4)

对于每一个事务*t*，*t* 的频繁模式因子被定义为：

(2-5)

并且*，*表示集合的数目。

如果一个事务*t*包含的频繁模式越多，那么FPOF(*t*)也就越大，即事务*t*本身更不可能被判为一个异常点。相反，如果事务*t* 包含的频繁模式很少，FPOF(*t*)也就越小，事务*t*就更可能是一个异常点。

对于事务*t*，若一个频繁项集，则认为*t*与*X*是相抵触(contradictive)的，那么*X*相对于*t*的抵触值为:

(2-6)

其中，第一部分表示属于*X*但是不属于*t*的元素个数，这表示包含元素个数多的项集抵触值会更大。第二部分代表了这个项集在在数据库中的频繁程度，意味着项集*X*的支持度越高，*contradict-ness(X,t)*越大，即*X*的支持度在某种程度上也衡量了*X*偏离正常模式的程度。这个公式衡量了每个频繁模式对事物的异常值贡献程度。随着数据量的增大，要留出所有与事物相关的抵触因子也是不可能的，因此更多时候是找出top-k抵触频繁模式。

算法2-1 FindFPOF

|  |
| --- |
| 输入： D //事务数据库  Minisupport //用户定义的最小支持度  Top-n //用户定义的top-n个异常事务  Top-k //用户定义的top-k抵触频繁模  输出：所有事务中top-n FPOF最大值作为异常事务 |
| 01 开始:  02 在数据库中，根据最小支持度(*minisupport*)挖掘所有的频繁模式集  03 /\*所有的频繁模式被记作：\*/  04 foreach 事务*t* do begin  05 foreach 在中的频繁模式*X*  06 if *t*包含*X* then  07 FPOF(*t*)=FPOF(*t*)+support(*X*)  08 return FPOF(*t*)  09 根据FPOF值按照升序排序并输出top-n个事务  10 foreach 在top-n中的事务*t* do begin  11 计算top-k抵触频繁模式并输出  12结束 |

1. WCFPOF(Weighted Closed Frequent pattern outlier factor)

FindFPOF的缺点在于：

1. FindFPOF需要找出所有的数据库中的所有频繁模式，然而在通常情况下，频繁模式的数量是很大的。尤其是包含长闭频繁模式的项集，它的频繁模式数量是指数级增长。
2. 事务会重复计算每一个闭频繁模式包含的频繁模式。如果一个事务仅仅只包含一个长闭频繁模式，由于重复计算内部频繁模式，因此这个事务比包含多个短闭频繁模式的事务更有可能被判为正常。

Jiadong Ren基于FindFPOF提出了一些改进。

定义包含n个事务的事务集。给定一个最小支持度阈值，该事务集的闭频繁模式集合定义为：

(2-7)

对每一个事务t，该t 的带权闭频繁模式异常因子定义为：

(2-8)

并且，表示集合中的元素个数。表示这个项集的长度。表示项集合*X*的权重。如果这个项集接近包含这个闭频繁模式，那这个闭频繁模式的权重就约等于1。如果一个事务*t*包含越多的闭频繁模式并且对应的闭频繁模式权重较大，那么这个事务的WCFPOF值就越大，这就意味着，这个事务更有可能是普通点。同样地，当事务*t*的WCFPOP越小，这个事务就越可能是异常点。所有的WCFPOF得值在0到1之间。

与FindFPOF算法类似，每一个事务*t*和每个闭频繁模式之间能计算一个带权重的抵触因子。对每一个事务*t*，当一个闭频繁项集*X*满足时，就认为该*X*与该*t*是相抵触的。*X*的抵触因子为：

(2-9)

第一部分表示在*X*中但是不在*t*中的元素个数，即越长的项集在*contradict-ness*中的贡献越大。第二部分*support(X)*表示在数据库中，*X*的频繁程度，意味着*X*相对于*t*的支持度越高，该*Contradict-ness*越大，能更好的将正常点和异常点分开。最后一部分是是闭频繁项集*X*对于*t*的权重。

该公式可以度量每一个闭频繁模式对于事务*t*的异常因子的贡献。同样地，随着数据量的增大，也不可能列出所有的抵触项集，而是选择top-k。

该方法解决了FindFPOF 算法中的部分问题，但是并没有解决大量频繁模式的问题。

1. LFPOF(Longer Frequent patter outlier factor)

Weiwei Zhang[41]提出了另外一种基于FPOP的改进方法。该方法基于的假设是：包含长频繁模式的样例更有可能是正常样例，因为这样的样例比其他的含有更多的频繁模式子集。而包含少量短频繁模式更有可能是异常点。

假设数据库包含n个事务，且n个事务均来自项集*I*。给定一个最小支持度，所有的频繁模式被记为:

(2-10)

F(*t*)是被包含在中的频繁模式，并且。由此得到

(2-11)

对于每一个事务t，该t的最长频繁模式异常因子定义为：

(2-12)

代表了F(*t*)中最长频繁模式的长度，表示事务*t*的长度。

如果事务*t*包含长频繁模式，那么它的LFPOF值就会很大，意味着这个事务*t*更有可能是正常点。反之，拥有更小的LFPOF值的事务更有可能是异常点。LFPOF的值也在0到1之间。

## 数据流的频繁模式异常检测

1. FPOF(Frequent pattern outlier factor)

Zengyou He将FPOF方法应用到数据流环境中检测异常点。同样地，样例包含越多的频繁模式，也就意味着这个样例更有可能是一个正常数据，因为这个样例包含更多的共同特征。相反，如果数据点包含更少的频繁模式，也就意味着这个点更有可能是异常点。

FPOF算法以及*contradict-ness*用于检测异常点并描述检测出的点异常的原因。算法一个关键的部分则是在使用频繁模式挖掘异常点时需要获取整个频繁模式集。然而在数据流中，不可能一次性获得所有的频繁模式。因此，不再挖掘所有的频繁模式，而是由近似算法根据数据流估计当前频繁模式。论文[42]即是用数据流的近似频繁模式来估算当前数据集的异常因子。由于数据量很大，通常计算top-k。

1. WFPOF(Weighted frequent pattern outlier factor)

Zhou xiao-yun[43]提出了一种与静态数据集完全不同的异常检测方法。该文也强调了维度在异常检测中对准确度的影响并提出了一个新的异常度量方法，该方法基于的假设同FPOF相同：样例点包含的频繁项集越多，这个点就越接近正常点，否则该点更有可能是异常点。

FindFPOF算法的计算复杂度较高。首先用Apriori方法发现所有的频繁模式，至少需要扫描数据库两遍，而在计算FPOP异常值的时候，还需要扫描数据库。其次，整个过程都是针对整个频繁，模式集而言的。

Zhou提出了一种新的算法，能在有限的时间和内存环境下，适用于数据流。该算法动态保持着频繁模式，并计算每个事务对应的WFPOP，并通过WFPOP来判断该点是否是异常点。该方法仅需一次扫描数据库，因其效率较高，则更适用于数据流环境下的异常检测。

(2-13)

是模式*x*的长度，*k*是数据空间的维度，

(2-14)

该算法认为，如果一个事务t包含越长的频繁模式并且包含越多的频繁模式，那么该*t*的WFPOF值会越大，也就意味着该事务更不能是异常点。相反，WFPOF值越小，该点越有可能是异常点。对频繁模式加权能提高对每个样例点异常成都的计算准确度。WFPOF的值在0-1之间。

1. FPCOP(Frequent pattern contradiction outlier factor)

Xianghong Tang[44]研究在数据流中采用的滑动窗口技术，并提出了基于频繁模式的数据流异常检测。该算法基于的假设是：事务在一定的滑动窗口内包含越多的与该事务有越少冲突的频繁模式，则该事务更有可能是正常点。反之，若该事务在一定的滑动窗口内包含越少的与该事务有越多的冲突的频繁模式，则该事务更有可能是异常点。

是包含所有的集合，是即将达到的数据流中的事务，且每一个事务都包含来自项集I中的若干项。定义一个滑动窗口，SW，包含N个最近的事务。N表示这个滑动窗口的宽度。一个在长度为N的滑动窗口中的项集*X*的支持度如果大于最小支持度，那么这个*X*就被认为是频繁的。给定一个最小支持度，这个数据流的频繁模式集被定义为：

(2-15)

,是包含在集合内的元素个数，是集合长度。该FPCOP值受频繁模式长度的影响。越大，就意味着频繁模式与事务*t*的冲突越大，因此这将导致一个更大的结果值。而越大的FPCOP值就表示这个事务更可能是异常点。FPCOP的值也在0到1之间。

1. MFPOF(Maximal frequent pattern outlier factor)

FengLin[60]提出了异常检测在高维时间序列数据流中的解决办法。同时还强调了在数据流中高效地挖掘频繁模式。

一个极大频繁模式的非空子集也是频繁模式，且一个极大频繁模式包含多个频繁模式。而极大频繁模式的个数远小于频繁模式的个数。那么一个事务包含更多的极大频繁模式也就意味着这个事务更不可能是异常点。该度量为：

(2-16)

其中是数据集D中的极大频繁模式。由于极大频繁模式远远小于频繁模式，因此能极大的减少计算的复杂度。

MFPOF极大频繁模式对事务异常因子的影响程度。事务t包含越多的极大频繁模式，也就意味着该数据更有可能是正常点。相反，事务t包含较少的极大频繁模式，就意味着该数据更有可能是异常点。

现在关于频繁模式的异常检测已经被越来越多的研究，然而这个领域依然有很多的问题需要注意：

1）提高该算法的准确度。

2）提高该算法的效率。

# 分类中数据不平衡问题

目前存在的大量成熟的分类算法都对正负样本不均衡的数据敏感。而异常检测的数据恰好又是正负样本极不平衡的。当正负样本数据不均衡时，分类算法表现不好的原因大约有以下两个方面：

1. 少数类包含的信息有限，难以发现其内部规律或确定其分布，进而导致难以识别。
2. 不恰当的类别划分。评估体系难以选择，算法往往倾向于忽略少数类，而把样本划分为多数类以获得更高的评分。传统以降低分类错误率，提高准确率作为模型优化目标，而不平衡数据则不适用。例如二分类问题中，癌症患者为“正例”，非癌症患者为“负例”，而实际得到的数据集中，非癌症患者远远大于癌症患者。假设癌症患者：非癌症患者=1:99，即使将所有测试集都判断为正例，此时的准确率也达到了99%。在用准确率的评估模型下，准确率极高，但这样的分类器造成的后果却是严重的，甚至根本是无法有效分出实际需要的样本的。

关于不平衡数据分分类问题一直是过去二十余年来的一个重要主题，学者们以其为基础展开了大量的探索和研究，提出了很多有意义的方法。大约从训练数据集，误分类代价等方面入手。具体方法：

1. 过抽样和欠抽样。不平衡分类问题由于不同类别样本数据差异甚大，因此可以通过采用调整训练样本集数量来处理。 过抽样的基本思想是增加少数类样本的数量来提高分类性能。简单的复制少数类样本的方法操作简单，却容易造成过拟合，因为没有给少数类增加新的信息。改进的过抽样方法通过复制少数样本并加入随机噪声或产生新的合成样本。欠抽样的方法通过随机减少多数类的数量来提高少数类的分类质量。但却会丢掉多数类的重要信息，不能充分利用已有的信息，尤其是数据来源很珍贵的情况下，不适应。
2. 提高正例误分类代价。如上述，将患病识别为非患病的代价远远大于将非患病识别为患病。在这样的情况下，认为识别正例的意义远大于识别负例。该方法提高正例误判的代价，使在当前数据集下，使所有样本分类后的误分类总代价最低。
3. 集成和特征选择。Boosting提升, random forest等方法等对训练集数据迭代加权，每次针对分错的数据增加权重，而对大量正确分类的数据降低权重，能更加关注容易被误分的少数类。正负样本的不平衡同时伴随着特征分布的不平衡。Boosting等方法能选择有区分力的特征，用于提高稀有类的识别率。
4. 基于频繁模式的异常检测

# 引言

异常点通常被认为是数据集中与其他数据显著不同的数据。异常点出现的原因不同，应用的领域不同，在数据中出现的形式也大不相同，那么在不同数据集中，异常点的定义也不同。这就使得有效发现数据集中的异常点并不是一件容易的事。在目前常用的异常点检测算法，基于统计的方法，基于聚类的方法，基于距离的方法和基于密度的方法，大多都和应用领域相关。因此，目前提出的大多数算法都是针对一个特定的应用领域。

关联规则是数据挖掘中众多模式中很重要的一种，用以揭示数据对象所包含属性之间的相关性。关联规则中发现频繁模式项集是整个挖掘过程中的关键技术和步骤。而发现的频繁模式通常具有相关性。例如，属于同一个类别的数据对象总是具有相同的特征。基于这类相关性，一些学者提出了许多新颖的基于频繁模式的异常点检测算法。该方法基于的事实前提是：异常点在数据集中数量少，密度低，数据对象之间共现性低。而正常点的数据量大，数据对象之间共现性高。在数据集中发现的频繁模式大多是基于正常模式的。通过数据对象包含正常频繁模式的程度估计该点是否属于异常点。如果，异常点总是出现在某一类别，我们也可以发现该类异常模式。在监督学习情况下，根据样本包含正频繁模式程度和负频繁模式的程度训练模型，将正负样本分开。

在异常点检测的数据集中，正负样本数据通常不平衡，抽样会破坏原来数据的内部结构，尤其是针对某些连续性且精确的数据集，导致不能充分利用好采集到的数据。而很多分类算法都需要在样本基本均衡的情况下，分类效果才比较明显。尽管数据集点本身数量很少，异常点更少，还需要一部分做验证集，因此数据本身很珍贵，尽可能不要丢弃。基于频繁模式的异常检测并不需要对正负样本进行抽样，因此我们只计算频繁模式出现的次数。最重要的是高效的频繁模式挖掘算法，而这已经被许多人研究。

# 相关定义

由Agrawal[61]提出的频繁模式挖掘：设 是m个不同项目的集合，给定事务数据库是包含n个事务的集合，且每个事务*t*由中的若干项组成。项集是属于的非空子集。中包含项的个数称为的长度，一个长度为*k*的项集称为*k-*项集。事务如果满足，称为事务*t*包含。在*D*中的支持度是指*D*中包含的百分比，即，其中指事务*t*的长度。如果的支持度不小于用户给定的最小支持度阈值*minisupport*，则称为*D*中的频繁项集。如果包含个项目，，那么又称频繁*k*-项集。

性质1：如果是频繁项集，那么的任何子集都是频繁项集。

性质2：如果是非频繁项集，那么*X*的任何超集都是非频繁项集。

是的真超项集，如果是的真子项集，即如果。项集在*D*中是闭的，如果不存在真超项集使得与在中具有相同的支持度计数。

定义1：在给定事务数据库中挖掘所有闭频繁项集就是：给定一个最小支持度阈值，找出中支持度大于等于最小支持度的闭频繁项集。这样的频繁模式满足:

1. 不存在的超集有

闭频繁模式的集合记为:

(3-1)

所有的闭频繁项集也称为闭频繁模式。

# 算法思想

## FPOF

Zengyou He[39]提出的基于频繁模式的异常检测算法FindFPOF，并给出了新的度量-频繁模异常因子FPOP(Frequent pattern outlier factor)和事务模式抵触量Contradict-ness。因为频繁模式反应了适用于大多数数据对象的“常见模式”，也就意味着包含更多“常见模式”的数据对象更可能是正常点。与此相反，包含的“常见模式”越少，则其成为离群点的可能性越大。

对每一个事务，该事务的频繁频繁异常因子定义为：

(3-1)

对于每一个事务t，如果项集X满足，那么频繁模式相对于事务的抵触量量定义为：

(3-2)

公式3-2能给出每一个频繁模式对指定事务的异常度量贡献程度。

FindFPOF能找出较少包含“平常模式”的事务。同时能给出异常点不包含的频繁模式与该事务的冲突。但该算法存在一些不足。FindFPOF通过导出数据的所有频繁模式来计算FPOF。当最小支持度较大且数据集有限时，难以得到恰当的“平常模式”，因此，无论是正常点还是异常点，FPOF都可能偏小，很难区别开。而当最小调整最小支持度，使产生过多的“正常模式”，导制频繁模式数量成指数级增长。尤其是满足条件的长频繁模式。例如一个长为k的频繁模式，其导出的频繁子集包含个频繁项集。当时，这个数量太大了。这样，不仅增加了算法计算复杂度，也会降低算法的准确度。

## WCFPOF

Jiadong Ren[28]提出的带权重的闭频繁模式异常检测。该方法不再挖掘所有的频繁模式集，而是挖掘数据对象的闭频繁模式。

对每一个事务，该事务的带权闭频繁异常因子定义为:

(3-3)

其中，为该事务集的闭频繁模式，即：。

对于每一个事务t，如果项集X满足，那么频繁模式相对于事务的抵触量量定义为：

(3-4)

但该方法只考虑到了正常情况下的频繁模式对数据对象的影响。在研究过程中发现，正常情况下的频繁模式也有可能出现在异常情况下。即某些在几乎所有正常模式和异常模式中出现的频繁模式会影响计算效果，权重应该适当降低。

## IWF-CFPOF

在研究中，我们发现，以上方法都只考虑了正常数据点的频繁模式。在本文中，我们提出加入异常数据集的频繁模式。在这种情况下，还需要考虑在所有模式中都出现的频繁模式。最坏情况下，如果有一个模式，在所有的数据中都出现，那么这个模式的权重应相对小些。该思想的来源是TF-IDF。IF-IDF是用以评估一个词对语料中某一份文件的重要程度。该指标随着词在文件中的次数成正比增加，却同时和在语料中的频率成反比。

对每一个事务，该事务的IWF-CFPOF(Inverse Word Frequency-Closed Frequent Pattern Factor)定义为:

(3-5)

其中，是来自闭频繁模式集中的一个闭频繁模式，满足。表示闭频繁模式的长度。当事务t包含模式时，对中的每一项分别计算该频繁模式对事务t的贡献：

(3-6)

事务t包含的闭频繁模式支持度越大，该项值越大，意味着该点越是靠近该频繁模式所代表的类。越大，该频繁模式越长，对IWF-CFPF影响越大，该点越有可能靠近该频繁模式所代表的类。公式3-6度量了每一个项在统一频繁模式下的不同贡献。计算了该点的权重。若该点在在大部分模式中都出现，则相对权重应降低，则该点为该模式IWF-CFPF贡献越少。

# 算法描述

算法3-1是Han等提出的FP-Growth[29]算法。我们用于挖掘频繁模式。

算法3-1 FP-Growth

|  |
| --- |
| 输入： D //事务数据库  Minisupport //最小支持度阈值  输出： 闭频繁模式的完全集 |
| 1. 构造FP树。 2. 扫描事务数据库D一次。收集频繁项的集合F和他们的支持度计数。对F按支持度计数降序排序，结果为频繁项列表L。 3. 创建FP树的根节点，标记为”null”。对D中每个事务Trans，执行：选择Trans中的频繁项，并按L中的次序排序。设Trans排序后的频繁项列表为[p|P]，其中p 是第一个元素，而P是剩余的元素的列表。调用insert\_tree([p|P],T)。该过程执行情况如下：   如果T有子女N使得N.item-name==p.item-name，则N的计数增加1,；否则创建一个新节点N，将其计数设置为1，链接到它的父节点T，并且通过结点链结构将其链接到具有相同item-name的结点。如果P非空，则递归地调用insert\_tree(P,N)。   1. 挖掘过程由递归调用FP-growth(FP-tree,null)实现   if Tree包含单个路径P Then  for 路径P中结点的每个组合  产生模式，其支持度计数support\_count等于中结点的最小支持度计数；  Else for Tree的头表的每一个  {  产生一个模式，其支持度计数  构造的条件模式基，然后构造的条件FP树  ifthen  调用  } |

此处我们采用启发式算法获取闭频繁模式，即先由FP-Growth得到所有频繁模式，再剪枝。

算法3-2：挖掘闭频繁模式集

|  |
| --- |
| 输入： 频繁模式集List<FrequentPattern>  输出： 闭频繁模式集CFPS(ClosedFrequentPatternSet) |
| for 频繁模式集从最后一个到第一个的频繁模式  {  If CFPS不存在的超集  CFPS.add(X)  Else if CFPS中存在的超集 )  CFPS.delete(X), CFPS.add(Y)  }  return CPFS |

算法3-3：Find

|  |
| --- |
| 输入： D //事务数据库  CFPS(D,support)//闭频繁模式集  输出： 对每一个事务t输出对应的IWF-CFPF |
| foreach 事务t in D {  foreach 闭频繁模式X in CFPS(D, support) {  if t 包含X:  foreach in X:  ifelse:    end if  end foreach  end foreach  return , |

# 算法流程

为了叙述方便，我们作如下定义：

定义1：

:正常频繁模式集

:异常频繁模式集

定义2：

:在频繁模式中出现的次数，即

:在频繁模式中出现的次数，即

则该点在正常频繁模式和异常频繁模式下的IWF-CFPF分别为：

(3-7)

(3-8)

定义3：

在频繁模式集中，频繁模式与事务抵触因子为：

(3-9)

在频繁模式集中，频繁模式与事务抵触因子为：

(3-10)

该值度量了未包含频繁模式对数据对象的贡献程度。对第一个式子来说，其表达了所有未包含在该数据对象的频繁模式的冲突程度。显然，频繁模式与数据对象t的相交部分越少，该值越大，也就意味着该点离这类越远。而频繁模式与数据对象t相交的部分越多，该值越小，数据对象t与该类冲突越小。同样，频繁模式的支持度也影响该值。

步骤：

1. 将数据预处理，并切分为训练集和测试集。
2. 将训练集的正负样本分开，正样本表示正常点，负样本表示异常点。
3. 给出正样本最小支持度阈值，并计算正样本数据集的闭频繁模式集
4. 给出负样本最小支持度阈值，并计算负样本数据集的闭频繁模式集
5. 根据正负闭频繁模式集计算每个训练集和测试集对应的IWF-FPOF作为特征1,2
6. 计算训练集和测试集的CF作为特征3，4
7. 将新生成的特征放入分类模型中训练。
8. 用训练好的模型对测试数据集预测
9. 评定结果

算法流程如图3-1所示。



图3-1 算法流程图

图1是分别计算正常和异常闭频繁模式集得到的IWF-FPOF值。能看出，原本难以分开的数据，经过模式的转换，已经初步具有可分的形式。此时，我们构建另外的特征，将点映射到更高维上。CF考虑了闭频繁模式集以外的因素，即与该模式点之间的冲突。

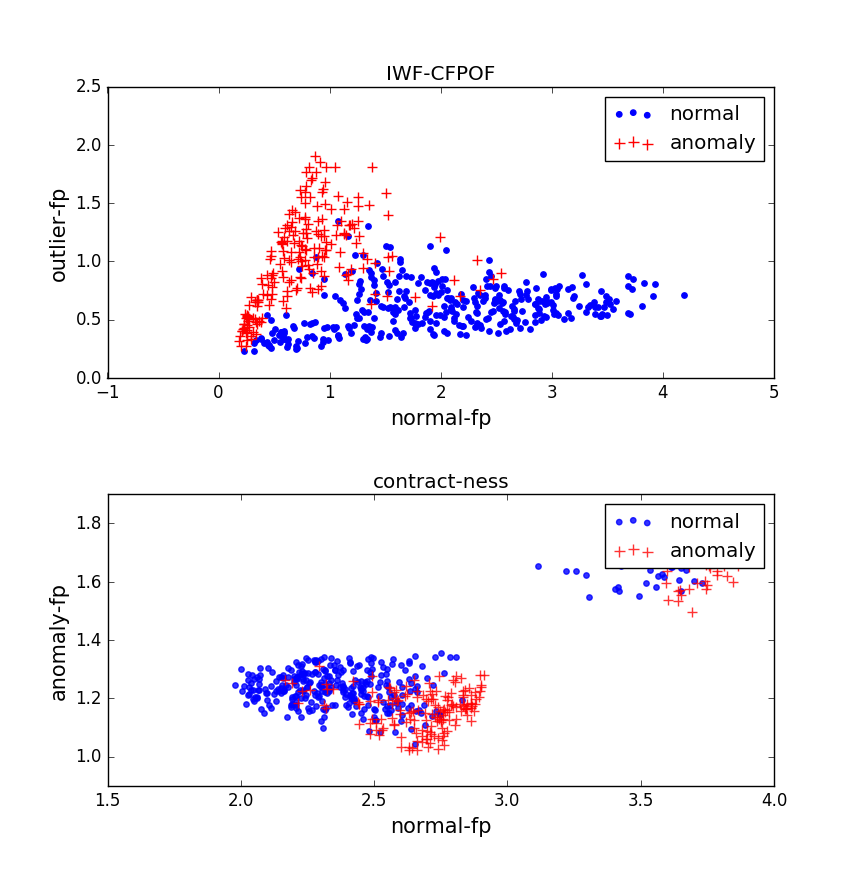


图3-2 特征转换为频繁模式得分图

# 实验与讨论

在这章，我们将分别研究IWF-CFPOF在不同模型，不同数据集下的表现。通过对比实验结果，进一步说明IWF-CFPOF在异常检测中对结果的影响。

在选择数据集时，针对异常检测领域的特点，我们按照以下准则选取数据集：

1. 选择公开数据集，要求这些数据集是真实数据，并且带有类别标签。
2. 选择用于二分类或多分类且都带有类别标签的数据集。
3. 特征数量不超过200，且特征包含连续型或离散型，或二者都有。缺失值尽可能少。
4. 这些数据集尽可能涵盖更广泛的领域。

为了保证结果的客观性，我们使用5个UCI[62]的公开数据集进行对比试验。

选择了数据之后，我们需要定义正常点和异常点。在二分类问题中，当数据集本身的正负有意义时时，我们按照数据集的标签来定义。例如标签为疾病的“良性”和“恶性”时，定义“恶性”为异常点，“良性”为正常点。当数据集本身不平衡且没有客观上的正常点和异常点时，我们定义点数量较少的为异常点，点数量较多的为正常点。当数据集标签本身没有意义时且正负样本均衡时，选择方差更大的候选集定义为异常点，而另一个候选集则为正常点。在多分类问题中，我们将标签按照质量、比例等划分正常点和异常点。例如，winequality-white数据集中，标签为酒的质量，取值范围0-10，而样本数据中只有3,4,5,6,7,8,9七个类别，数值越高代表酒的质量越好，那么我们就可以选3,4类别，占总数的3.74%为异常点，来构造不平衡数据集。

我们定义正常样本点为正例，即Positive。异常样本点为负例，即Negtive。

表3-1 数据集

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编号 | 数据集 | 样本量 | 特征数 | 类别数 | 正样本 | 异常点样本 |
| 1 | winequality-white | 4898 | 12 | 10 | 4715 | 183 |
| 2 | Breast-cancer-wisconsin | 699 | 10 | 2 | 458 | 241 |
| 3 | wdbc | 569 | 32 | 2 | 357 | 212 |

有了样本数据集，我们就可以按照算法流程进行对比试验。表3-2是按照频繁模式挖掘算法得出的各项频繁模式集。

表3-2 数据集频繁模式计算

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据集 | 最小支持度 | | 频繁模式(normal) | 频繁模式(anomaly) | 闭频繁模式(normal) | 闭频繁模式(anomaly) |
| 正常 | 异常 |
| winequality-white | 5 | 2 | 47871 | 17589 | 15668 | 520 |
| Breast-cancer-wisconsin | 5 | 2 | 6048 | 9301 | 2668 | 3492 |
| wdbc | 5 | 3 | 34206 | 24856 | 4962 | 3597 |

为了防止数据产生过拟合，我们先将数据分为训练集和测试集。在挖掘闭频繁模式集的时候，仅使用训练数据。用训练数据得到的频繁模式计算测试集的IWF-CFPOF。

由于模型都是带标签的，所以我们使用以下指标来评估：

积极准确率TPR（公式）、积极召回率PPV（公式），积极F1值（公式），消极准确率（公式），消极召回率NPV（公式），消极值（公式）。

表3-3 真实标签和预测数据关系表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | 预测数据 | |  |
| 预测:True | 预测:False |
| 真实标签 | 标签:Positive | True Positive(TP) | False Negative (FP) | P |
| 标签:Negative | False Positive (FN) | True Negative(TN) | N |
|  | | T | F |  |

(3-11)

(3-12)

(3-13)

(3-14)

(3-15)

(3-16)

我们选用SVM，KNN，朴素贝叶斯，决策树，逻辑斯蒂回归等算法进行对比试验。其中，SVM中，我们采用径向基函数(RBF)。KNN算法，中，我们选用KD树。同时采用不同的K值，即K=2，k=3，k=4等多个参数对比。在朴素贝叶斯中，我们选用高斯朴素贝叶斯。其特征的释然函数假设为高斯分布（公式）。决策树中，我们用基尼系数选择最优特征，同时决定该特征的最优二值切分点。

(3-17)

(3-18)

实验一：

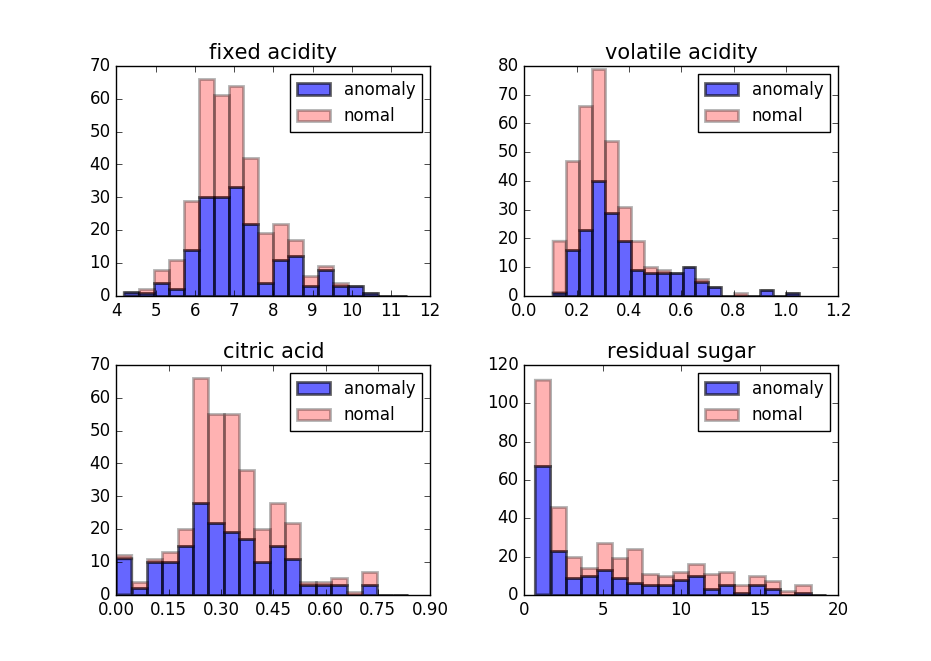
我们选用来自UCI的WineQualityData数据集。其中说我们选用白酒数据，输入11个属性值， 包括非挥发性酸，挥发性酸，柠檬酸，残留糖分，氯化物，游离二氧化硫，总二氧化硫，密度，pH值，硫酸盐，酒精度，以及输出-质量(0-10)。我们定义1-4为异常点，共计183个，占总数的3.74%，5-9为正常点，共计4715个。占总数的96.26%。

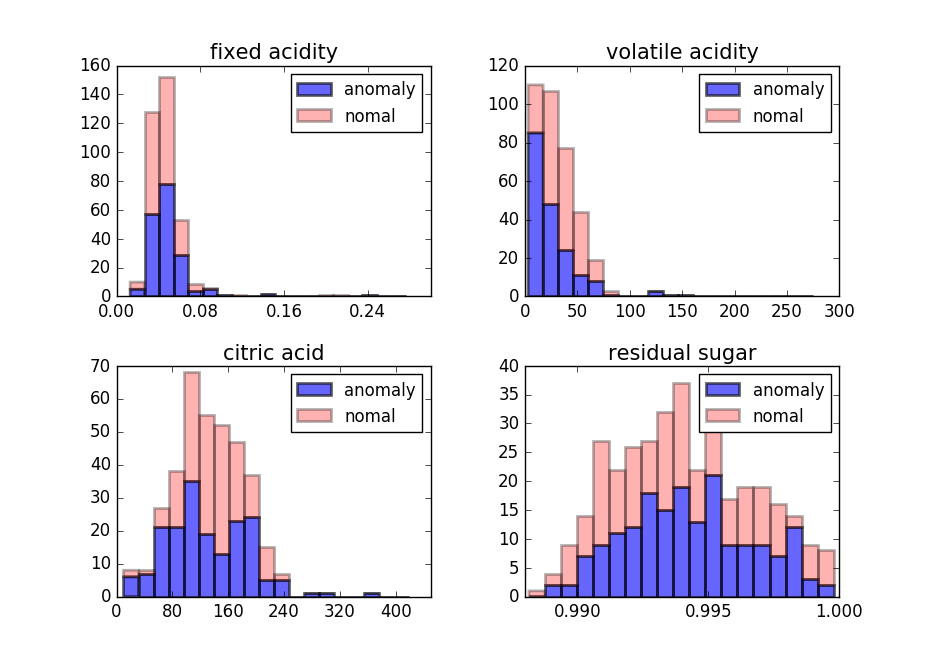
表3-4 Wine数据集样本描述

|  |  |  |
| --- | --- | --- |
| 样例 | 类标号 | 所占百分比 |
| 正常样本 | 5,6,7,8,9 | 96.26% |
| 异常类 | 1,2,3,4 | 3.74% |

我们统一随机抽样方法抽样，得到正负样本数量均衡的样本数据集。然后考虑从数据规律入手，观察源数据，并尝试进一步统计以发现数据内部规律。

1）对抽样数据进行统计描述。图中显示了将异常点的统计分布叠加在正常点统计分布之上的对比效果。峰值显示了该类别出现次数最多的取值。对比二者，可发现两类点分布近似，仅略有差别





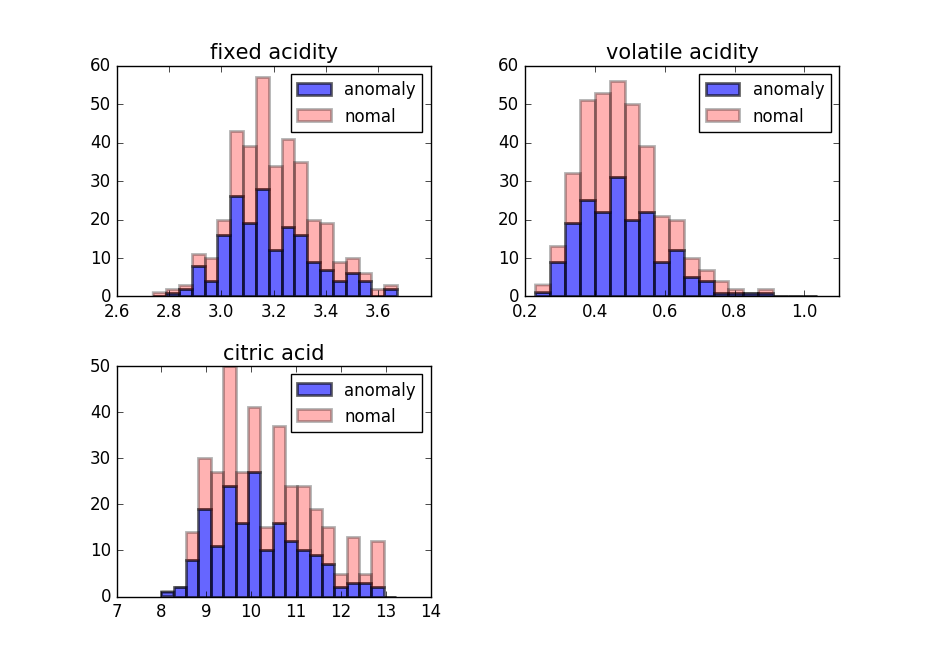
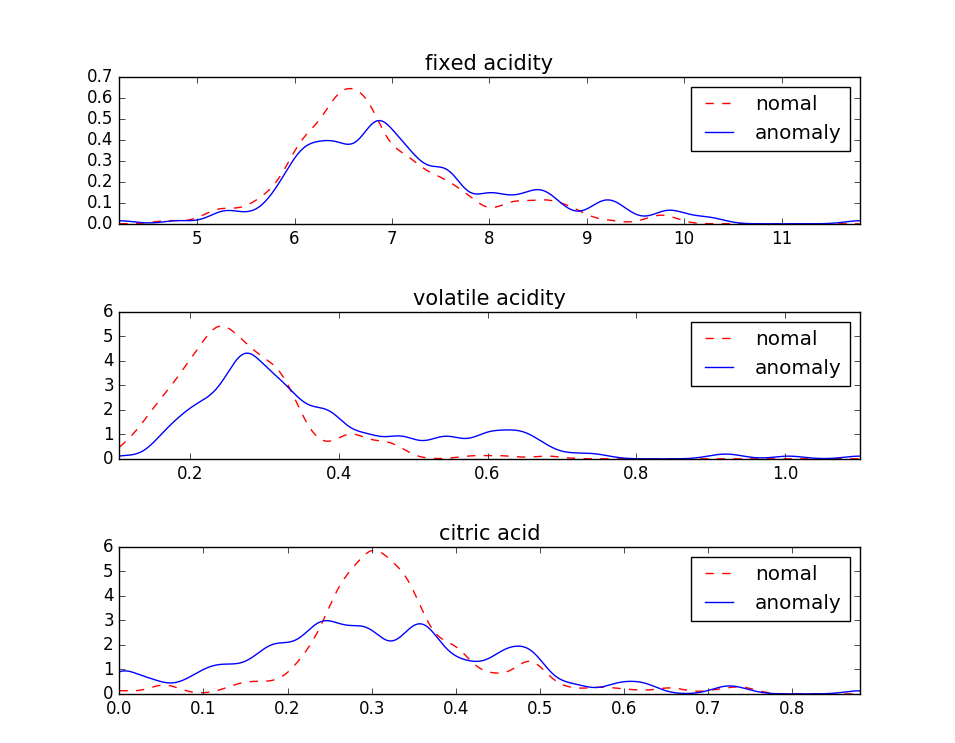
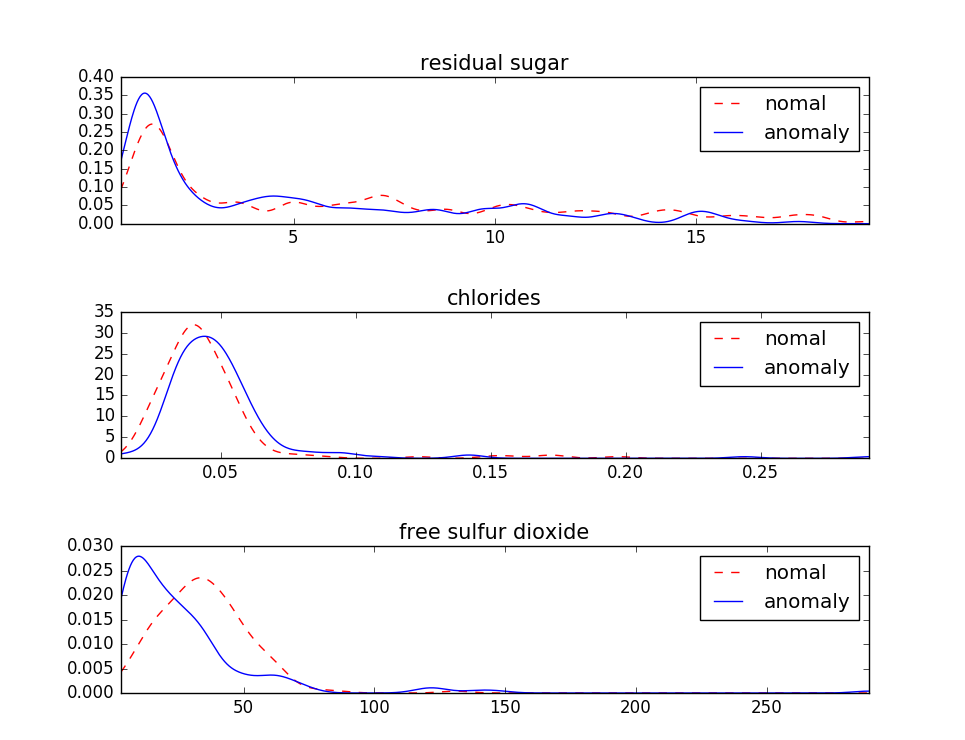


图3-3

2）对数据正常点和异常点进行高斯核概率密度估计，得到图。图为11个属性分别在高斯核概率密度估计下的数据分布图。进一步说明了，该数据集的正常点和异常点很类似，难以区分。





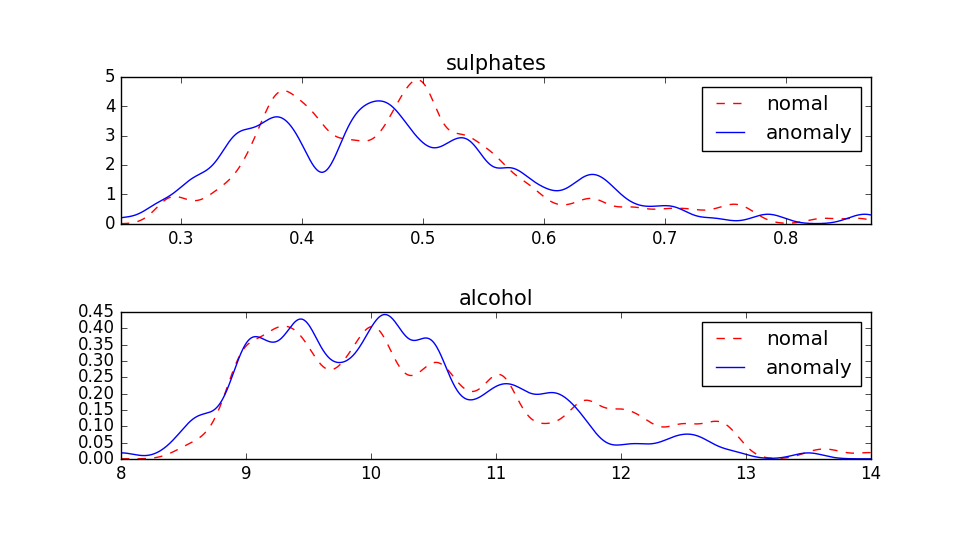
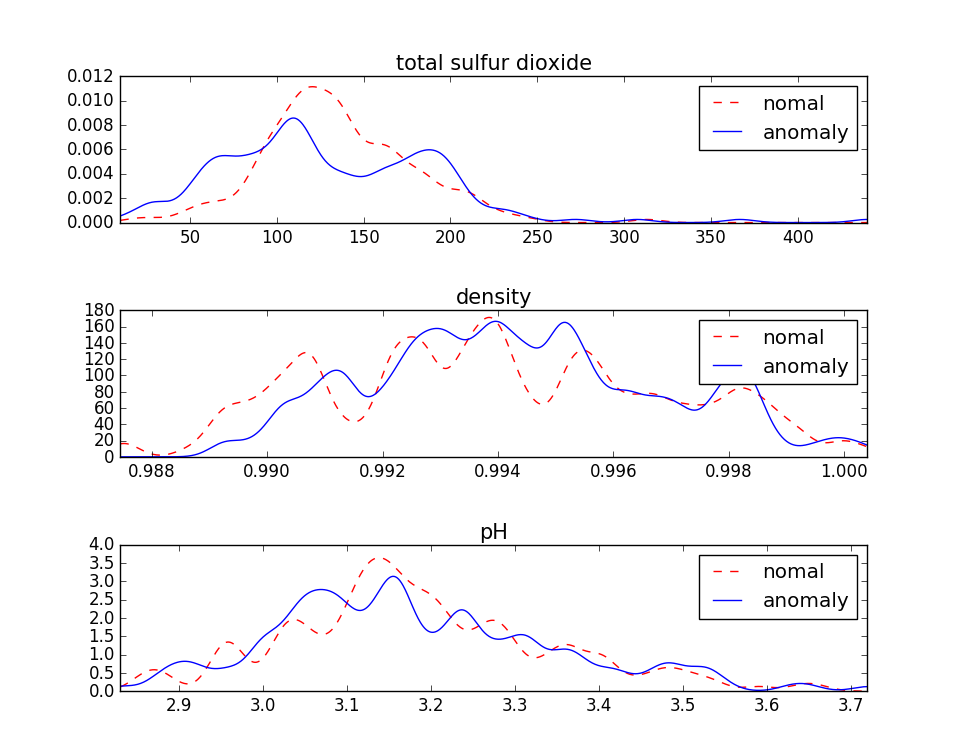


表3-4显示了Wine数据集在各个模型下的表现。该统计是在一次随机实验下的结果。该表显示了在普通模型下，直接训练特征得到的真值表。而表3-5则是将特征经过IWF-CFPOF和Contract-ness变换后，再放到统一模型中训练得到的结果。可以看到，大部分模型的表现都更好。其中，逻辑斯蒂回归有较大的提升。

表3-4 Wine数据集在各模型下的表现

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 算法名称 | 准确率 | | 召回率 | | F1 | |
| 评分类别 | 1(TPR) | 0(TNR) | 1(PPV) | 0(NPV) | 1 | 0 |
| 逻辑斯蒂回归 | 0.81 | 0.62 | 0.62 | 0.81 | 0.70 | 0.70 |
| 朴素贝叶斯 | 0.76 | 0.60 | 0.75 | 0.62 | 0.67 | 0.68 |
| SVM | 0.83 | 0.68 | 0.71 | 0.81 | 0.77 | 0.74 |
| KNN(k=2) | 0.86 | 0.61 | 0.57 | 0.88 | 0.69 | 0.72 |
| KNN(k=3) | 0.88 | 0.70 | 0.71 | 0.88 | 0.79 | 0.78 |
| KNN(k=4) | 0.88 | 0.70 | 0.71 | 0.88 | 0.79 | 0.78 |
| 决策树 | 0.94 | 0.75 | 0.76 | 0.94 | 0.84 | 0.83 |

表3-5 特征变换后表现

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 算法名称 | 准确率 | | 召回率 | | F1 | |
| 评分类别 | 1(TPR) | 0(TNR) | 1(PPV) | 0(NPV) | 1 | 0 |
| 逻辑斯蒂回归 | 0.95 | 0.83 | 0.86 | 0.94 | 0.90 | 0.88 |
| 朴素贝叶斯 | 0.82 | 0.80 | 0.86 | 0.75 | 0.84 | 0.77 |
| SVM | 0.89 | 0.78 | 0.81 | 0.88 | 0.85 | 0.82 |
| KNN(k=2) | 0.84 | 0.72 | 0.76 | 0.81 | 0.80 | 0.76 |
| KNN(k=3) | 0.86 | 0.81 | 0.86 | 0.81 | 0.86 | 0.81 |
| KNN(k=4) | 0.86 | 0.81 | 0.86 | 0.81 | 0.86 | 0.81 |
| 决策树 | 0.90 | 0.82 | 0.86 | 0.88 | 0.88 | 0.85 |

我们再将各个算法对同一数据集进行10次交叉验证，并计算平均结果，得到表3-5所示的结果对比。

表3-6原始数据训练结果与变换FP特征后训练结果对比（Wine）

|  |  |  |
| --- | --- | --- |
| 算法 | 准确度ACC(原始特征) | 准确度ACC(FP特征) |
| SVM | 0.66(+/-0.16) | 0.86(+/-0.18) |
| KNN(k=2) | 0.70(+/-0.11) | 0.89(+/-0.11) |
| KNN(k=3) | 0.69(+/-0.17) | 0.90(+/-0.13) |
| KNN(k=4) | 0.69(+/-0.12) | 0.90(+/-0.13) |
| 朴素贝叶斯 | 0.73(+/-0.20) | 0.85(+/-0.17) |
| 决策树 | 0.74(+/-0.10) | 0.87(+/-0.11) |
| 逻辑斯蒂回归 | 0.74(+/-0.11) | 0.85(+/-0.15) |

我们发现，经过FP特征变换之后，数据在分类器下的预测准确率有了显著地提高。尤其是KNN和逻辑斯蒂回归等算法。我们将数据点的特征通过统计图的形式来观察，发现属于同一类别的点在欧式空间距离上更近。图3-3则显示了特征变换后的数据特点。该图上部分中，横坐标表示该点在正常频繁模式下的IWF-CFPOF值，纵坐标表示该点在异常频繁模式下的IWF-CFPOF值。由图可知，正常点更偏向于右下方，即取值更大，而相对较小。图下半部分显示了正负样本在正负频繁模式下的频繁模式与事务抵触值。

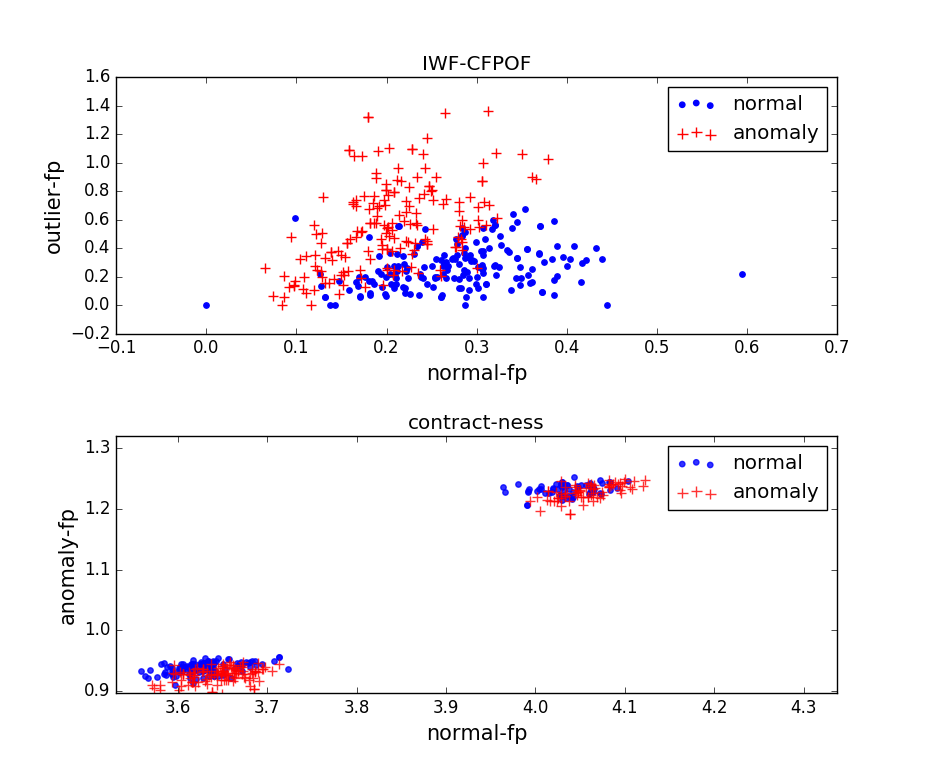
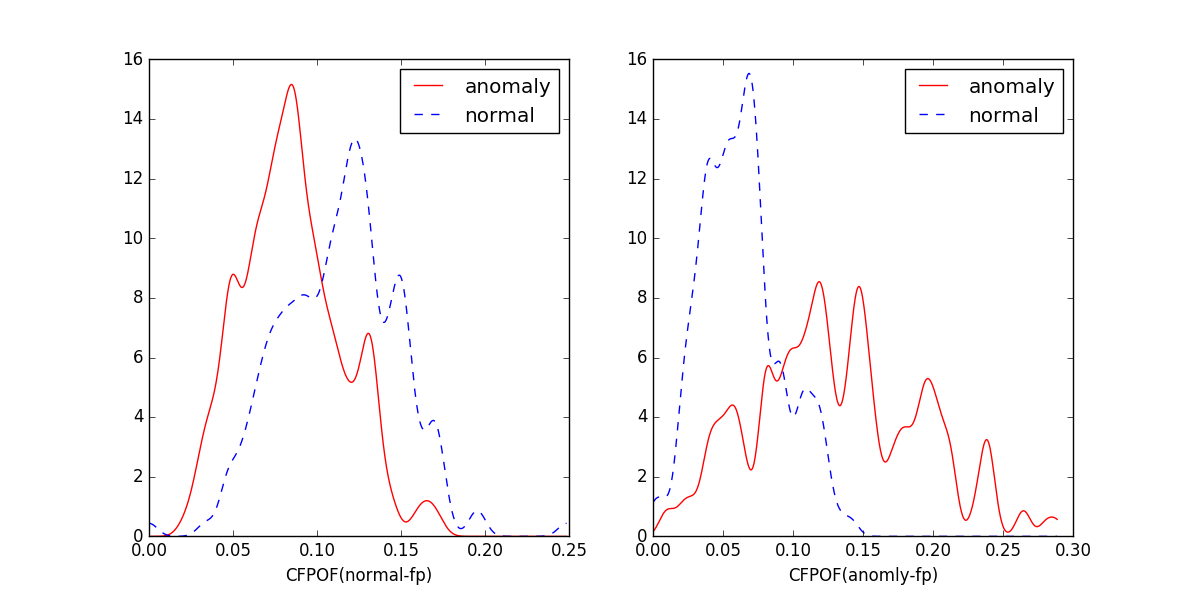


图3-3 Wine 频繁模式特征图

我们进一步考察数据点的IWF-CFPOF值，估计正负样本的概率密度，分析新特征在不同情况下的概率密度分布，得到如图3-4。该图左侧显示了正常点和异常点在正常频繁模式下的IWF-CFPOF。蓝色虚线表示的正常点相对于异常点整体右移，表示正常点的IWF-CFPOF值略偏高。图3-4右侧图表示正常点和异常点对于异常模式的得分概率分布差别很大。

图3-5左侧显示了异常点在正负频繁模式下得分的概率密度估计。蓝色虚线表示异常点异常频繁模式得分的密度估计。红色实线表示异常点正常频繁模式得分的密度估计。右图同理，表示正常点在正负频繁模式下得分的概率密度估计。由图可以看出，异常点在0-1.4的区间异常IWF-CFPOF得分较均衡，而正常点大部分都集中在0.2区间。



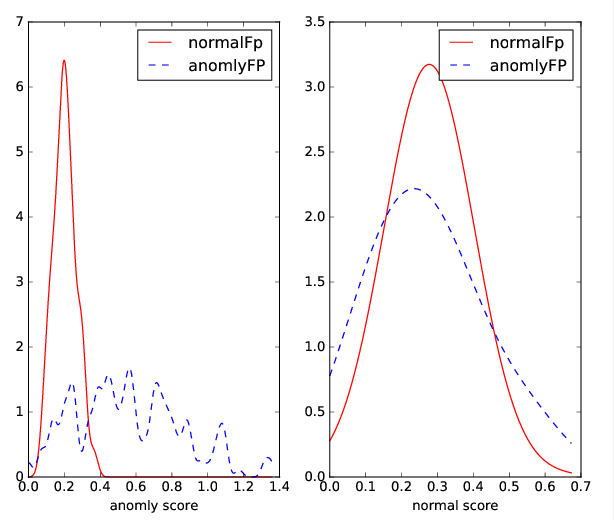
图3-4 经频繁模式变换后的数据点高斯密度估计

图3-5 经频繁模式变换后的数据点高斯密度估计

实验二：

我们选用UCI中breast cancer数据集。

表3-7 breast cancer数据集样本描述

|  |  |  |  |
| --- | --- | --- | --- |
| 样例 | 类标号 | 样本数量 | 所占百分比 |
| 正常样本 | Malignant | 239 | 35% |
| 异常类 | benign | 444 | 65% |

该数据集在各个模型下与变换特征后的结果对比如表3-6所示。这组数据我们看到，原始模型已经能很好地预测点的属性。但变幻FP特征后，结果至少也保持了原始数据的准确度。

表3-8 原始数据训练结果与变换FP特征后训练结果对比（breast cancer）

|  |  |  |
| --- | --- | --- |
| 算法 | 准确度ACC (原始特征) | 准确度ACC (FP特征) |
| SVM | 0.96(+/-0.06) | 0.96(+/-0.05) |
| KNN(k=2) | 0.95(+/-0.06) | 0.96(+/-0.05) |
| KNN(k=3) | 0.97(+/-0.04) | 0.96(+/-0.06) |
| KNN(k=4) | 0.96(+/-0.05) | 0.96(+/-0.08) |
| 朴素贝叶斯 | 0.97(+/-0.06) | 0.97(+/-0.06) |
| 决策树 | 0.95(+/-0.08) | 0.97(+/-0.05) |
| 逻辑斯蒂回归 | 0.96(+/-0.05) | 0.96(+/-0.06) |

进一步分析该数据的IWF，能发现该两项指标已经能很好地区分数据。图3-6上图是各个样本在正负频繁模式集下各自的IWF-CFPOF值的分布。下图是各个样本点在正负频繁模式下的模式和事务抵触因子的分布。

正常点的normal-fp（正常频繁模式下的IWF-CFPOF）显著地高于异常点，而异常点的anomaly-fp（异常频繁模式下的IWF-CFPOF）也显著地高于正常点。而数据对象的contract-ness也出现了明显的差异。

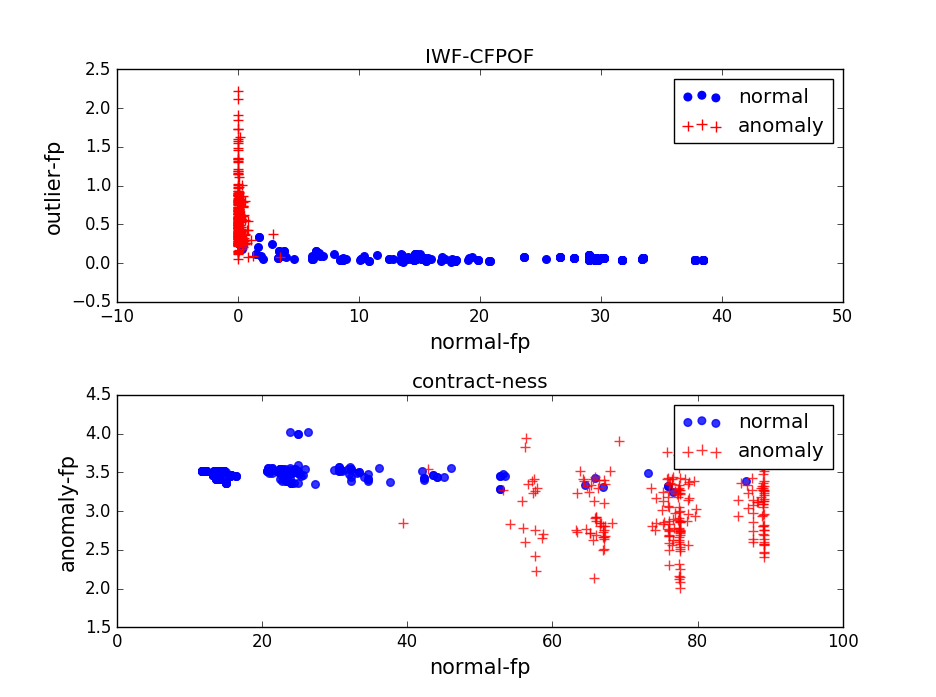


图3-6频繁模式特征变换后对比图

实验三：

我们选用UCI中WDBC数据集。

表3-9 wbdc数据集样本描述

|  |  |  |  |
| --- | --- | --- | --- |
| 样例 | 类标号 | 样本数量 | 所占百分比 |
| 正常样本 | Malignant | 212 | 37.26% |
| 异常类 | benign | 357 | 62.74% |

表3-10原始数据训练结果与变换FP特征后训练结果对比(wdbc)

|  |  |  |
| --- | --- | --- |
| 算法 | 准确度ACC (原始特征) | 准确度ACC (FP特征) |
| SVM | 0.63(+/-0.01) | 0.97(+/-0.04) |
| KNN(k=2) | 0.93(+/-0.04) | 0.94(+/-0.07) |
| KNN(k=3) | 0.92(+/-0.04) | 0.97(+/-0.04) |
| KNN(k=4) | 0.93(+/-0.03) | 0.96(+/-0.04) |
| 朴素贝叶斯 | 0.94(+/-0.06) | 0.97(+/-0.03) |
| 决策树 | 0.92(+/-0.08) | 0.97(+/-0.04) |
| 逻辑斯蒂回归 | 0.95(+/-0.05) | 0.97(+/-0.05) |

本实验数据集的频繁模式分布点为图3-7所示。该图与上文描述一致。由图可知，正负样本的各项目概率分布已经有明显的不同。

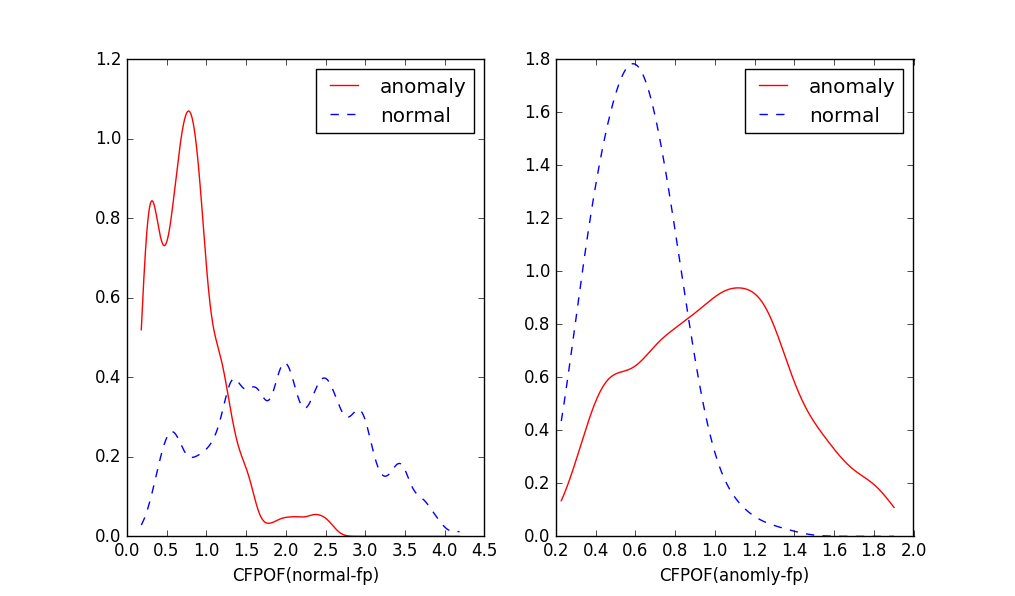


图3-7

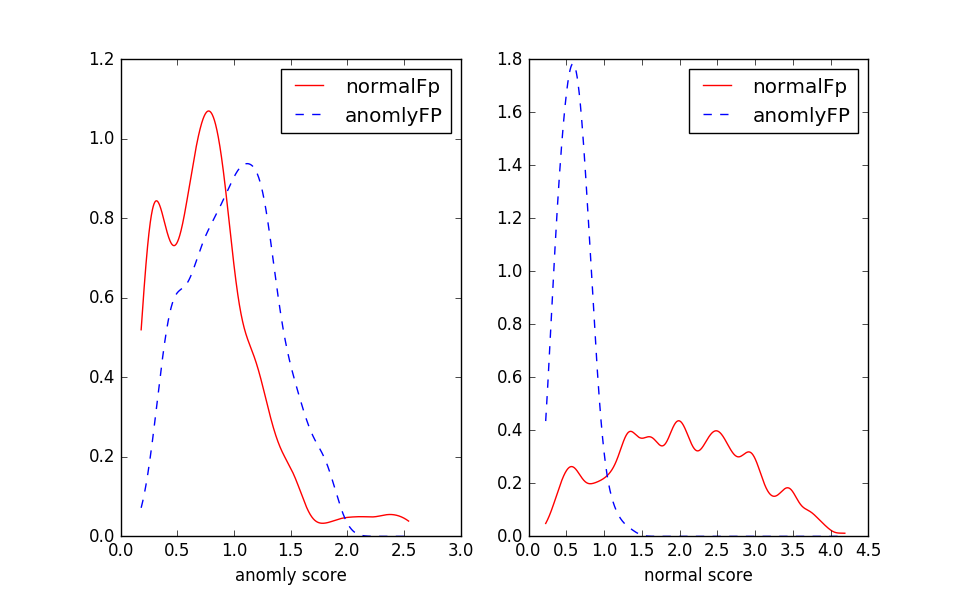


图3-8

# 小结

基于频繁模式的异常检测基于这样一个假设：某一类的点总是具有相同的模式。正常点总是包含着大部分正常点的模式，这些点具有很多相似的地方。而异常点除了较少包含正常模式之外，同时还包含较多异常点的模式。因此，我们分别计算正常点频繁模式和异常点频繁模式带给数据点的影响。由于频繁模式本身的限制，如果导出所有的频繁模式，会存在以下几个问题：

频繁模式数量太大，对每个点迭代求频繁模式因子的时候，计算复杂度很高。

对于长频繁模式，会不断重复该频繁模式的子模式的计算。会将每一个子模式都计算。本身就会造成某些点的值远远高于其他点。数据分布会过于分散。

因此我们考虑用挖掘闭频繁模式集的方法。挖掘闭频繁模式的算法很多。

在采用FPOF对数据集计算正负频繁模式集的实验过程中，我们发现，超过一半的点，即具有较高的正常闭频繁模式集因子，也具有较高的异常闭频繁模式因子。分析发现，是由于某些数据在两类中出现数量均衡。因此，我们加入对每个点单独计算的权重。

在计算IWF-CFPOF时，由于正负样本不均衡，因此正负闭频繁模式数量往往差别很大。但这并不会影响结果的准确度。在计算IWF-CFPOF时，均考虑了结果与对应闭频繁模式集的归一化。使得，尽管正负闭频繁模式集数量不一，但IWF-CFPOF值也在同一范围内。更加适用于模式的计算。

1. 在线学习的异常检测

# 在线频繁模式挖掘

在机器学习和数据挖掘领域中，数据流也是很重要的一个部分。通常将数据流建模为一个随时间延续而无限增长的动态数据集合。由于数据流本身的特点，例如持续产生，数据海量且无限等，使得对于数据流的研究与传统应用于静态数据的方法有很大差距。传统的频繁模式挖掘方法需要多次扫描数据库，且需要扫描全局数据库，无疑是不能适用于数据流处理所需要的“one-pass”特性。数据流的数据分布不是一成不变的，这样的动态进化[63]特征，也就很难建立精确的预测模型。

目前国内外学者针对数据流的异常检测已经提出了一些新的思路和方法，主要包括改进静态方法使其适应数据流和增量学习、在线集成学习。在数据流上进行频繁模式挖掘也被陆续提出。尽管基于数据流的算法形式各异，但是都需要解决无限的数据和有限的存储空间之间的冲突。根据各个算法不同的数据结构和策略，目前的数据流频繁模式挖掘主要分为三类[64]，基于抽样的方法，基于计数的方法和基于散列的方法。

## 基于抽样的方法

基于抽样的技术直观地解决了超大数据集的问题，数据通过概率选择决定是否进入内存继续处理。该方法能监测到内存中的每一个数据，同时能以的概率保证错误下线。抽样方法的难点包括：首先结果的精确度随着样本数量的增大而提高，因此为了保证精确度，样本大小可能会超过容量。其次，基于抽样的方法并不是适用于所有的模型。如果已抽到的样本被删除，是不可能从历史数据中重新获取该样本的。最后，在数据流大小不确定的情况下，样本大小需要人为确定。而预设和调整样本大小都是有难度的。基于抽样的方法大多都提出了解决这些问题的方法。

随机抽样是典型的抽样方法。该方法随着可用空间的减少可能会导致更大的估计误差。但该方法可以通过混合计数抽样方法来提高。Gibbons和Matias提出的简洁抽样[73]，就是通过将样本点保存到<key，计数>的方式，来保存随机样本变量。因此简洁抽样可以包含更多的点，同时与随机 抽样相比，在相同的空间下，该方法有更高的准确度，尤其是当数据有偏差的时候。当空间耗尽时，简洁抽样通过逐步减少抽样概率来控制内存占有率。Probabilistic-Inplace[65]采用固定大小的样本集，能以较大概率发现至少出现次的项集，并以包含这些项集的超集形式返回。然而这两类算法都是任意减少抽样的概率，在空间消耗很大的时候，可能会忽略掉很多原本频繁的项集。Manku和Motwani提出的Sticky算法[66]，通过规律的降低抽样的概率才避免这类情况。该方法也是基于抽样的近似方法，抽样率（即抽样概率的倒数）与即将到来的流数据的大小的对数成正比。

## 基于计数的方法

该类方法采用一系列计数器监测数据流。首次到达的数据，进入监测状态并计为1。当到达的数据与监测的某一数据相同时，该数据计数值自增。当到达内存极限或计数器边界时，会触发清理策略，清除当前计数最小的项。该方法基于的假设是，当前估计值不频繁的项集，有更大的可能是不频繁项集。Space-Saving[67]提出了一种概要数据结构，用以保存当前的项集，并记录其出现的次数。结果返回检测到的计数值加上误差区间满足最小支持度的频繁模式。Manku，Motwani[69]提出的Lossy Counting是第一个追踪不同项的估计误差的算法。该方法将数据流依据特定的宽度，分成不同的桶，每个桶的编号刚好就是对应的阈值。所有的项都加入各个桶。当到达桶边界的时候，触发剪枝。Sticky Sampling和Lossy Counting使用历史数据流增量地估算数据的频繁模式，能保证挖掘出数据流中满足误差界和最小支持度的频繁模式。

## 基于散列的方法

基于散列的方法采用了新的概要数据结构的，是一种适用于数据流的近似算法。基于散列的方法能以概率近似保证结果的准确度。该方法使用sketch[68]这种概要数据结构，能根据随机映射，提取数据流的主要信息。Cormode和Muthhukrishnan[70]提出了通过群测验的方法找出出现最多的k项。通过随机近似的方法估算最频繁出现的项。sketches对于频繁项识别有很多优点。1）该方法有预先可控制的内存，处理过程也需要依赖该固定内存大小。2）算法是线性的，适用于分布式。3)该方法记录了所有项的出现，从某种意义上来说，该方法是无损的。4）对数据流到达的顺序不敏感。5）对在线学习中的很多模型都适用。

# 频繁模式近似计算

基于频繁模式的异常检测中，最关键的步骤之一就是如何在数据流中挖掘频繁模式。现存的频繁模式挖掘算法，Apriori或者FP-growth都需要多次扫描数据库。而由前文讨论的数据流的特点，多次扫描是不可能的。那么，我们需要在内存有限的情况下，只提取数据的重要信息并存储，即采用“概要数据结构”保留信息。该数据结构不仅要能在新数据到来时，能适当调整内部结构以存储数据对象信息，还要能在到达内存极限或某个边界时触发清理机制，去除掉不可能是频繁项集的那些数据。该数据结构通常需要尽可能地减小系统反应时间，增大系统的准确度和置信度。

算法4-1滑动窗口FP-Miner

|  |
| --- |
| 输入：数据流S，误差，计数器个数c，滑动窗口大小w，支持度阈值support.  输出：大于support的频繁项集 |
| 开始：  数据流滑动窗口集合  频繁模式项集  初始化滑动窗口和频繁项集均为空。  Foreach 数据流中的事务  Temp.add((s,1))  If SW包含  同时调整的位置  Foreach 滑动窗口集合中的  If >2  Temp.add  If SW不包含    If SW大小超过c  SW删除时间点距离当前最远的一个  Foreach itemset e in temp  If FP包含该项集e    elseif |Fp|>c  删除FP中f最小的元组  添加到FP中 |

该算法用滑动窗口模型维持当前部分数据集，用维持数据主要信息。滑动窗口的设立，使得不[51]需要存储所有的数据流。为保证滑动窗口内部数据都是最新的数据，每次都删除距离当前时间结点最远的元组。对每次到来的数据来说，其被检测的概率是均等的。滑动窗口内也设有计数，只有初次到来的数据计数为1。之后到来的数据，若已经被该窗口检测到，只需增加计数。提高了滑动窗口的利用率。刚到来的数据与滑动窗口中的每一个元组求交集，该交集的频繁计数为元组的频繁计数。每一个交集都可能是频繁模式。

算法的第二部分是监察交集中的集合项。如果该集合项已经存在于频繁模式中了，只需将计数加1。如果没有出现在频繁模式中，则需要添加该集合的元组，并设置其频繁计数为在生成交集时产生的频繁计数。这样做的好处是，所有的元组频繁计数并不是都从1开始，而是到目前为止，该元组可能出现的次数。因此并没有忽略之前数据带来的影响。

算法有两次涉及到剪枝。第一次是在滑动窗口时。当窗口大小超过设定值时，剪掉距当前时间点最远的元素。该步骤是为了保证数据不因为数据流的无限增长而超出处理的能力范围。第二次剪枝是在频繁模式计数器的部分。当频繁模式超过了给定的阈值时，需要剪掉当前频繁模式集频繁计数最小的元组。该步骤基于这样的假设：当前不频繁的项集很有可能是不频繁项集。由于算法在初期，对每一次到达的数据都计算了频繁计数，因此也有很多计数为1的项。该步骤即从计数为1的项集开始剪枝。这步剪枝控制着频繁模式的数量。

该算法得出的频繁模式更紧凑。近似于闭频繁模式。避免了大量短频繁模式的生成。以表4-1中的数据为例。假设滑动窗口的大小，频繁模式计数器。滑动窗口元组记为，频繁模式计数器中元组记为。假设，。 表5-1事务数据

|  |  |
| --- | --- |
| TID | 项列表 |
| T1 | I1,I2,I5 |
| T2 | I2,I4 |
| T3 | I2,I3 |
| T4 | I1,I2,I4 |
| T5 | I1,I3 |
| T6 | I2,I3 |
| T7 | I1,I3 |
| T8 | I1,I2,I3,I5 |
| T9 | I1,I2,I3 |

Counter=3

表5-2 算法步骤

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 当前数据 | 滑动窗口 | Temp | FP | 当前最小频繁计数项 |
| a | 初始化 | {} | {} | {} | 0 |
| b | (I1,I2,I5) |  |  |  |  |
| c | (I2,I4) | , |  | , |  |
| d | (I2,I3) |  |  |  |  |
| e | (I1,I2,I4) |  |  |  |  |
| f | (I1,I3) |  |  |  |  |
| g | (I2,I3) |  |  |  |  |
| h | (I1,I3) |  |  |  |  |
| i | (I1,I2,I3,I5) |  |  |  |  |
| j | (I1,I2,I3) |  |  |  |  |

由表5-2可见，每次删除的当前最小频繁模式在通常情况下计数都很小。在，情况下，中,不满足条件。而中,，和中,满足条件。因此结果返回{}。

在步骤a-d中，滑动窗口和频繁项集计数器都没有超过阈值，因此，直接加入该集合。每一次temp的生成都是在数据对象达到后，滑动窗口还未改变时，与各个项集求交集。只有当该滑动窗口包含该数据对象时，先改变滑动窗口计数，然后再进行求temp，如g所示。此时，需要增加该元组的计数，由变为。步骤i显示了当频繁模式计数器大小到达阈值时，如果新生成的元组频繁计数比当前频繁模式计数器中频繁计数最小的元组还小的话，则直接丢弃该组。否则，如其他步骤所示，删除频繁模式计数器中频繁计数最小的元组。

# 时间衰减

FP-outlier计算是在基于当前窗口下的频繁项集计算的。随着时间的不断增长，频繁项集计数也会不断增长，如图4-1所示。刚开始的点，由于滑动窗口和频繁计数都为0，因此，整个过程值接近于0，且增长缓慢。随着时间的不断推移，频繁模式计数器中的计数值越来越大。由于数据流是无限大的，那么可以推测，频繁计数不仅影响处理效率，甚至可能会超出计算机所能处理的范围。同时，异常检测只需要能区分开正常模式和异常模式即可。由于异常模式本身值很小，对于过于大的频繁计数，对异常检测并没有太大的意义。因此，我们需要设置时间衰减因子。由于数据流中数据对象是持续发生变化的，因此我们需要区分当前事务和历史事务。在[72]中，应用了时间衰减模型降低了事务模式支持树的权重。在本文中，我们认为各频繁模式支持数随着时间t变化，并且以衰减函数进行衰减。

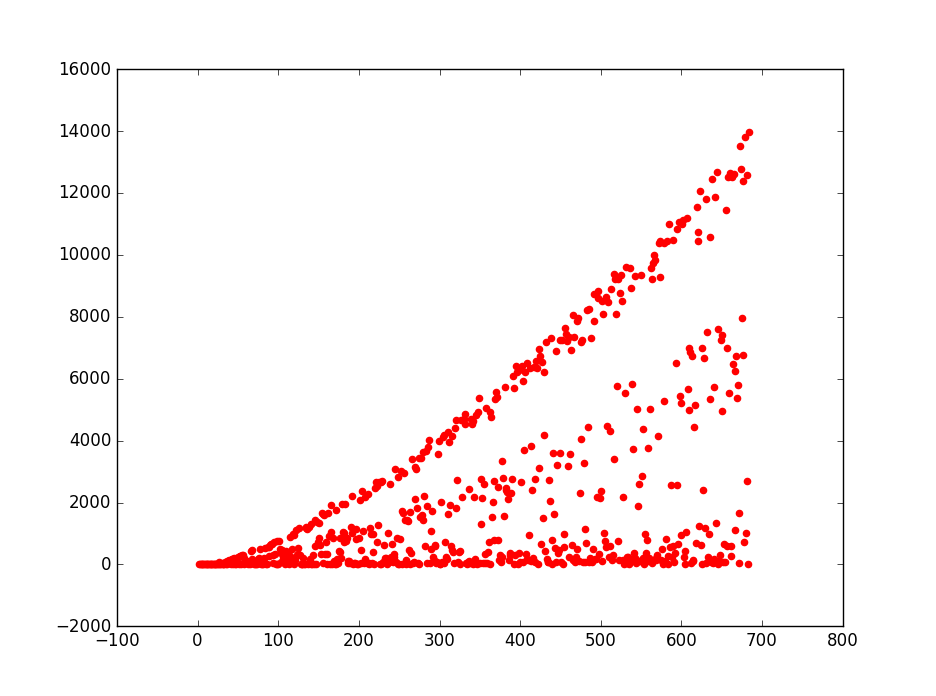


图4-1数据流频繁模式

# 数据流FP-outlier计算

数据流的FP异常检测同普通静态数据集下的异常检测基于的假设相同：如果该事务包含更多的频繁项，那么该FP得分也越大，也就意味着，该点更可能是一个普通点。

算法4-2数据流FP-Detection

|  |
| --- |
| 输入：数据流S，误差，计数器个数c，滑动窗口大小w，支持度阈值support.  输出：Top-n异常点 |
| 开始：对每一个事务：  通过算法4-1更新当前频繁模式集  在当前频繁模式上，计算该事务t 的异常指数  计算该t 的显著变化水平。  输出当前一段时间内top-k异常频繁模式  结束 |

由该算法的假设可以看出，当该物体发生漂移时，仅会在第一次发生这种跃迁时能计算出该点值的明显变化，并发出异常事务的警告。之后，随着这类异常模式的不断增多，该类频繁模式进入滑动窗口和频繁模式计数器的次数也会不断增加，进而变为正常模式。那么，与此相同，当数据集中出现集体异常点时，由于模式的变化，除了第一次到达的异常点会发出异常事务警告之外，在之后的点计算中便“自适应”了该集体异常点。尤其是当该集体异常点数量庞大时。因此，此算法不适用于集体异常点的检测。而对于离群异常点，有着显著地效果。

# 实验和讨论

由于动态数据集本身的特点，我们在做实验时，在数据清理这一步有一些注意事项。

1. 我们在选择数据集的时候，尽量选择不平衡的数据集。不平衡的数据集中，正常点包含的范围更大。那么频繁模式计数器中累计的正常类的频繁模式数量明显多于异常点。当异常点到来时，其突变性会更明显。
2. 滑动窗口指向的当前窗口对频繁模式有着直接的影响。在模拟动态数据流时，尽量使数据点乱序。即将正负样本打乱到最大程度。

我们挑选以下数据集：

表5-3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据集 | 变量个数 | 缺失值 | 数据数量 | 正负样本比例 |
| Breast Cancer Wisconsin | 32 | 无 | 569 | 357:212 |
|  |  |  |  |  |
|  |  |  |  |  |

实验一：

我们采用滑动窗口=500，频繁计数器=3000来进行试验。图4-2显示，大量的正常点聚集在图的底部，即FP-score极小。正常的点都在图形的上方。正好符合我们所构想的，正常点所包含的正常模式较多。在图的左下侧，则是红色和蓝色点混合，而时间计数器显示1，即算法刚开始运行时，滑动窗口中和频繁计数器都为空或者很小。此时新到来的点，FP-score都很小。随着时间的推移，窗口已满且频繁计数器已满，状态逐渐接近稳定。此时，异常事务和正常事务逐渐区别开。

图4-3是放大图4-2的局部所得。从图中能看出，正常点已经极少出现在异常点之中了。

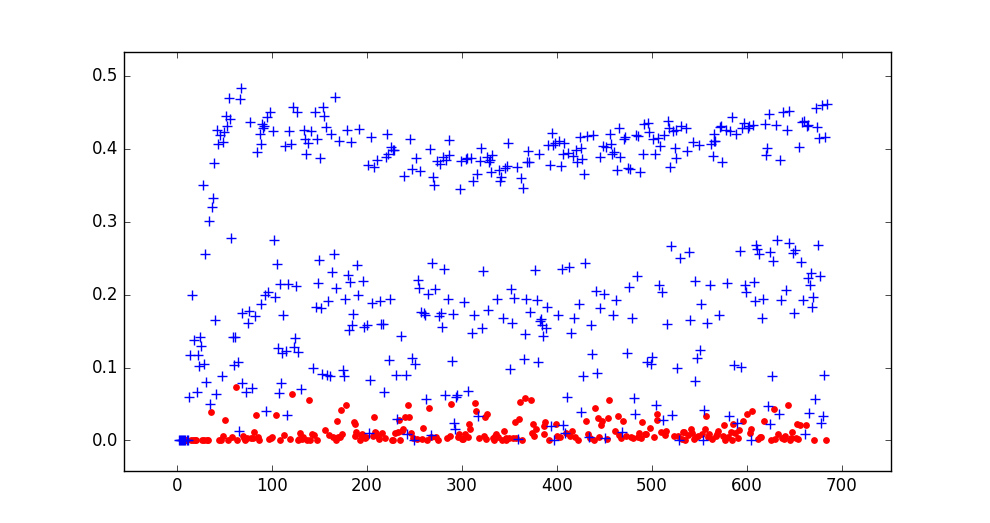


图4-2

得到FP-score后，我们根据数据特征选择适用的分割方法。该数据集得分特征如图4-4所示。图中上升的点是正常模式的积累过程，陡然下降的部分大多是数据集中异常点的出现。我们尝试用前后数据比来作为划分FP­-score的分界线。却发现，结果并不理想。例如，我们采用作为优化函数，并取。其中代表t时刻的事务的FP-score，代表t+1时刻的事务所得的FP-score。得到如表4-5的结果。

表4-5

|  |  |  |
| --- | --- | --- |
|  | P | N |
| T | 370 | 67 |
| F | 74 | 172 |

分析数据发现连续出现的异常点通常只能检测出第一个。由于第一个异常点已经使得FP-score等分很低，第二个点在第一个点的基础上得到补偿，就无法检测到。

表4-6

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 时刻点 | FP-score | |  | | 真实标签 | | 预测 | |
| 294 | 0.059395 | -2.81945 | | 0 | | 0 | |
| 295 | 0.062024 | -0.04426 | | 0 | | 0 | |
| 296 | 0.005157 | 0.916852 | | 1 | | 1 | |
| 297 | 0.009198 | -0.7836 | | **1** | | **0** | |
| 298 | 0.344039 | -36.4025 | | 0 | | 0 | |
| 299 | 0.383605 | -0.115 | | 0 | | 0 | |
| 300 | 0.001809 | 0.995284 | | 1 | | 1 | |
| 301 | 0.002278 | -0.25944 | | **1** | | **0** | |
| 302 | 0.190246 | -82.5022 | | 0 | | 0 | |
| 303 | 0.008647 | 0.95455 | | 1 | | 1 | |
| 304 | 0.386106 | -43.654 | | 0 | | 0 | |
| 305 | 0.38767 | -0.00405 | | 0 | | 0 | |

该规律和图4-4发现的多个连续下降点是一致的。

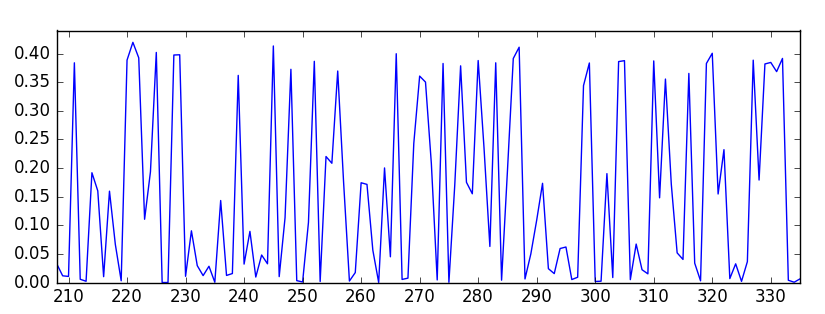


图4-4

为此，我们将标准设置到异常点发生之前的那个正常点。并提高分界点阈值，。得到表5-5。能看出，表中数据已经能全部正确预测。

表4-7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 时刻点 | FP-score |  | 真实标签 | 预测 |
| 295 | 0.062024 | 0.849155 | 0 | 0 |
| 296 | 0.005157 | 0.987458 | 1 | 1 |
| 297 | 0.009198 | 0.977629 | 1 | 1 |
| 298 | 0.344039 | 0.163277 | 0 | 0 |
| 299 | 0.383605 | -0.115 | 0 | 0 |
| 300 | 0.001809 | 0.995284 | 1 | 1 |
| 301 | 0.002278 | 0.994061 | 1 | 1 |
| 302 | 0.190246 | 0.504057 | 0 | 0 |
| 303 | 0.008647 | 0.97746 | 1 | 1 |
| 304 | 0.386106 | -0.00652 | 0 | 0 |
| 305 | 0.38767 | -0.00405 | 0 | 0 |
| 306 | 0.004985 | 0.987141 | 1 | 1 |

此时的真值表为：

表4-8 0.8

|  |  |  |
| --- | --- | --- |
|  | P | N |
| T | 407 | 19 |
| F | 37 | 220 |

表4-9 0.85

|  |  |  |
| --- | --- | --- |
|  | P | N |
| T | 412 | 31 |
| F | 32 | 208 |

表4-10 0.9

|  |  |  |
| --- | --- | --- |
|  | P | N |
| T | 412 | 57 |
| F | 21 | 182 |

第五章 结论与展望

致 谢

三年的研究生生活就要结束了，在这三年宝贵的时间里，不论是在学习还是在生活上，我都收获了很多，成长了很多。我的成长离不开那些所有关心我，帮助我的老师们、同学们和朋友们，在此，我要对他们表示深深的敬意和谢意。

首先，我要由衷的感谢我的导师傅彦老师和陈端兵老师。他们严谨治学的态度、一丝不苟的工作态度和渊博的知识都深深地感染了我，成为我心中的楷模。从如何阅读文献，确定选题到如何开展学术研究，和撰写，两位老师都给予了悉心指导和帮助。使我快速的掌握了科学研究方法，并不断地成长和进步。同时，两位老师良好的工作习惯，热情的生活态度，带领大家为实验室创造了浓郁和谐的科研氛围，使我们在日常生活和学习中都收获颇丰。在此，谨向两位导师表达崇高的敬意和诚挚的感谢，感谢您们在我的学习和生活中给予的指导帮助。

其次，感谢实验室其他老师在我论文完成过程中给予的指导，尤其是董强老师的细心指导。帮我梳理文章结构，调整语言的表达方式。再此，谨向董强老师表达由衷的感谢！

最后，我要感谢实验室的同学们。很高兴能和你们一起度过了一个充实的研究生生活。在学习上，我们相互学习，共同进步；在生活上，我们相互照顾，共同分享。

再一次感谢我身边所有关心我帮助我的人，特别感谢我的父母亲，我的亲戚朋友们。是你们一直默默地支持我，鼓励我，让我不怕困难，勇敢迎接人生中的每一次挑战！

参考文献

1. V. Colizza, A. Barrat, M. Barth´elemy, et al. The Modeling of Global Epidemics: Stochastic Dynamics and Predictability[J]. Bulletin of Mathematical Biology, 2006, 68(8):1893–1921.
2. Han J. Data mining: concepts and techniques / Jiawei Han and Micheline Kamber[J]. Data Mining Concepts Models Methods & Algorithms Second Edition, 2000, 5(4):1 - 18.
3. Donoho S. Early detection of insider trading in option markets[C]// Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, Usa, August. DBLP, 2004:420-429.
4. M. F. Augusteijn, B. A. Folkert. Neural network classification and novelty detection[J]. International Journal of Remote Sensing, 2002, 23(14):2891-2902.
5. Hodge V, Austin J. A Survey of Outlier Detection Methodologies[J]. Artificial Intelligence Review, 2004, 22(2):85-126.
6. Aggarwal C C, Yu P S. Outlier detection for high dimensional data[C]// ACM SIGMOD International Conference on Management of Data. ACM, 2015:37-46.
7. Gupta M, Gao J, Aggarwal C, et al. Outlier Detection for Temporal Data[C]// Morgan & Claypool, 2014:129.
8. D'Haeseleer P, Forrest S, Helman P. An Immunological Approach to Change Detection: Algorithms, Analysis and Implications[C]// Security and Privacy, 1996. Proceedings. 1996 IEEE Symposium on. IEEE, 1996:110.
9. Dasgupta D, Nino F. A comparison of negative and positive selection algorithms in novel pattern detection[C]// IEEE International Conference on Systems, Man, and Cybernetics. IEEE, 2000:125-130 vol.1.
10. Simon Byers, Adrian E. Raftery. Nearest-Neighbor Clutter Removal for Estimating Features in Spatial Point Processes[J]. Journal of the American Statistical Association, 1998, 93(442):577-584.
11. Torr P H S, Murray D W. Outlier detection and motion segmentation[C]// Optical Tools for Manufacturing and Advanced Automation. International Society for Optics and Photonics, 1993.
12. Diehl C P, Hampshire J B. Real-time object classification and novelty detection for collaborative video surveillance[J]. 2002, 3(3):2620-2625.
13. Singh S, Markou M. An approach to novelty detection applied to the classification of image regions[J]. IEEE Transactions on Knowledge & Data Engineering, 2004, 16(4):396-407.
14. Fawcett T, Provost F. Activity Monitoring: Noticing interesting changes in behavior[C]// ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 1999:53-62.
15. Mahapatra A, Srivastava N, Srivastava J. Contextual Anomaly Detection in Text Data[J]. Algorithms, 2012, 5(4):469-489.
16. Schlimmer J C, Granger R H. Incremental Learning from Noisy Data.[J]. Machine Learning, 1986, 1(3):317-354.
17. Schlimmer J C, Granger R H. Incremental Learning from Noisy Data.[J]. Machine Learning, 1986, 1(3):317-354.
18. Widmer G, Kubat M. Learning in the Presence of Concept Drift and Hidden Contexts[J]. Machine Learning, 1996, 23(1):69-101.
19. Widmer G, Kubat M. Learning in the Presence of Concept Drift and Hidden Contexts[J]. Machine Learning, 1996, 23(1):69-101.
20. Abadi D, Carney D, Cherniack M, et al. Aurora:a data stream management system[C]// ACM SIGMOD International Conference on Management of Data, San Diego, California, Usa, June. DBLP, 2003:666.
21. Motwani R, Widom J, Arasu A, et al. Query Processing, Resource Management, and Approximation in a Data Stream Management System[C]// IEEE Vlsi Test Symposium. 2002:245--256.
22. Domingos P, Hulten G. Catching up with the Data: Research Issues in Mining Data Streams.[J]. Workshop on Research Issues in Data Mining & Knowledge Discovery, 2001.
23. Yi B K, Sidiropoulos N D, Johnson T, et al. Online Data Mining for Co-Evolving Time Sequences[C]// International Conference on Data Engineering. IEEE, 2000:13-22.
24. 数据挖掘概念与技术
25. R.Agrawal, R.Srikant. Fast algorithms for mining association rules.In proc.1994 Int.Conf.Very Large Data Bases(VLDB’94), Sept.1994,pp.487-499,Santiago,Chile
26. J.S.Park, M.S.Chen, P.S.Yu. An effective hash-based algorithm for mining association rules[J]. Acm Sigmod Record, 1997, 24(2):175-186.
27. A.Savasere, E.Omiecinski, S.Navathe. An Efficient Algorithm for Mining Association Rules in Large Databases[C].International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc. 1995:432-444.
28. H.Toivonen. Sampling Large Databases for Association Rules[J]. Proc Very Large Data Bases Conf, 1999:134--145.
29. J.Han, J.Pei, Y.Yin. Mining frequent patterns without candidate generation[J]. Acm Sigmod Record, 1970, 29(2):1-12.
30. H.Toivonen. Sampling Large Databases for Association Rules[J]. Proc Very Large Data Bases Conf, 1999, [C@3ff758([C@ec9330):134--145.
31. DENG. A New Algorithm for Fast Mining Frequent Itemsets Using N-Lists[J]. Science China Information Sciences, 2012, 55(9):2008-2030.
32. Deng Z H, Lv S L. Fast mining frequent itemsets using Nodesets[J]. Expert Systems with Applications, 2014, 41(10):4505-4512.
33. Z.Liu, Q.Sun, J.Li. A Matrix Mining Method with FP-Tree for Generation of, Frequent Patterns[J]. 2015.
34. Kumar A V S, Wahidabanu R S D. A Frequent Item Graph Approach for Discovering Frequent Itemsets[C]// International Conference on Advanced Computer Theory and Engineering. IEEE, 2008:952-956.
35. Duemong F, Preechaveerakul L, Vanichayobon S. FIAST: A Novel Algorithm for Mining Frequent Itemsets[C]// International Conference on Future Computer and Communication. IEEE Computer Society, 2009:140-144.
36. Venkateswari S, Suresh R M. An efficient for discovery of frequent itemsets[C]// International Conference on Signal and Image Processing. IEEE, 2010:531-533.
37. Pei J, Han J, Lu H, et al. H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases[C]// IEEE International Conference on Data Mining. IEEE Xplore, 2001:441-448.
38. J Davis, M.Goadrich. The relationship between Precision-Recall and ROC curves[C]// International Conference on Machine Learning. DBLP, 2006:233-240.
39. He Z, Xu X, Huang J Z, et al. FP-outlier: Frequent pattern based outlier detection[J]. Computer Science & Information Systems, 2005, 2(1):103-118.
40. Ren J, Wu Q, Hu C, et al. An Approach for Analyzing Infrequent Software Faults Based on Outlier Detection[C]// International Conference on Artificial Intelligence and Computational Intelligence. IEEE Computer Society, 2009:302-306.
41. Zhang W, Wu J, Yu J. An improved outlier detection algorithm based on frequent pattern[C]// International Conference on Computer Application and System Modeling. IEEE, 2010:V7-344-V7-349.
42. Manku G S, Motwani R. Approximate frequency counts over data streams[J]. Proceedings of the Vldb Endowment, 2012, 5(12):1699-1699.
43. Zhou X Y, Sun Z H, Zhang B L, et al. A Fast Outlier Detection Algorithm for High Dimensional Categorical Data Streams[J]. Journal of Software, 2007, 18(4):933-942.
44. Tang X, Li G, Chen G. Fast Detecting Outliers over Online Data Streams[C]// International Conference on Information Engineering and Computer Science. 2009:1-4.
45. Agrawal R, Lin K I, Sawhney H S, et al. Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases[C]// International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc. 1995:490-501.
46. Agrawal R, Srikant R. Fast algorithms for mining association rules[M]// Readings in database systems (3rd ed.). Morgan Kaufmann Publishers Inc. 1998.
47. Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation[C]// ACM SIGMOD International Conference on Management of Data. ACM, 2000:1-12.
48. Agrawal R, Shafer J C. Parallel mining of association rules[J]. Knowledge & Data Engineering IEEE Transactions on, 1996, 8(6):962-969.
49. Estan C, Varghese G. New directions in traffic measurement and accounting[C]// Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. ACM, 2002:323-336.
50. Gilbert A C, Kotidis Y, Strauss M J, et al. QuickSAND: Quick Summary and Analysis of Network Data[J]. Dimacs Tech Repo, 2001, 103(20):7566-7570.
51. Srivastava A N, Stroeve J. Onboard detection of snow, ice, clouds, and other geophysical processes using kernel methods[J]. 2010.
52. Madden S, Franklin M J. Fjording the Stream: An Architecture for Queries Over Streaming Sensor Data[C]// International Conference on Data Engineering, 2002. Proceedings. IEEE, 2002:555-566.
53. Kargupta H, Park B H, Pittie S, et al. Mobi Mine:monitoring the stock market from a PDA[J]. Acm Sigkdd Explorations Newsletter, 2002, 3(2):37-46.
54. Arasu A, Babu S, Widom J. CQL: A Language for Continuous Queries over Streams and Relations[J]. 2003:1-19.
55. Metwally A, Agrawal D, El Abbadi A. Efficient Computation of Frequent and Top-k Elements in Data Streams[C]// International Conference on Database Theory. Springer-Verlag, 2005:398-412.
56. Metwally A, Agrawal D, Abbadi A E. Using Association Rules for Fraud Detection in Web Advertising Networks[J]. Scientia Horticulturae, 2005, 112(4):422-426.
57. Baumes J, Goldberg M K, Magdon-Ismail M, et al. Discovering Hidden Groups in Communication Networks.[C]// Intelligence and Security Informatics, Second Symposium on Intelligence and Security Informatics, ISI 2004, Tucson, AZ, USA, June 10-11, 2004, Proceedings. DBLP, 2004:378-389.
58. Kargupta H, Bhargava R, Liu K, et al. VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring[C]// Siam International Conference on Data Mining, Lake Buena Vista, Florida, Usa, April. DBLP, 2004.
59. Aggarwal C C. On classification and segmentation of massive audio data streams[J]. Knowledge and Information Systems, 2009, 20(2):137-156.
60. Lin F, Le W, Bo J. Research on Maximal Frequent Pattern Outlier Factor for Online High-Dimensional Time-Series Outlier Detection.[J]. Journal of Convergence Information Technology, 2010, 5(10):66-71.
61. R. Agrawal, R. Srikant. Fast Algorithms for Mining Association Rules. In: Proc of VLDB’94, pp. 478-499, 1994

1. <http://archive.ics.uci.edu/ml/index.html>
2. Widmer G, Kubat M. Learning in the Presence of Concept Drift and Hidden Contexts[J]. Machine Learning, 1996, 23(1):69-101.
3. Liu H, Lin Y, Han J. Methods for mining frequent items in data streams: An overview[J]. Knowledge and Information Systems, 2011, 26(1):1-30.
4. Demaine E D, López-Ortiz A, Munro J I. Frequency Estimation of Internet Packet Streams with Limited Space[C]// Algorithms - ESA 2002, European Symposium, Rome, Italy, September 17-21, 2002, Proceedings. DBLP, 2002:348-360.
5. Manku G, Motwani R. Approximate frequency counts over streaming data[J]. Journal of Ecclesiastical History, 2003, 54(3):536-536.
6. Metwally A, Agrawal D, El Abbadi A. Efficient Computation of Frequent and Top-k Elements in Data Streams[C]// International Conference on Database Theory. Springer-Verlag, 2005:398-412.
7. Gilbert A C, Kotidis Y, Strauss M J, et al. QuickSAND: Quick Summary and Analysis of Network Data[J]. Dimacs Tech Repo, 2001, 103(20):7566-7570.
8. Manku G, Motwani R. Approximate frequency counts over streaming data[J]. Journal of Ecclesiastical History, 2003, 54(3):536-536.
9. Cormode G, Muthukrishnan S. What's hot and what's not:tracking most frequent items dynamically[C]// ACM, 2003:296-306.
10. Gaber M M, Krishnaswamy S, Zaslavsky A. On-board Mining of Data Streams in Sensor Networks[M]// Advanced Methods for Knowledge Discovery from Complex Data. Springer London, 2005:307-335.
11. 李国徽, 陈辉. 挖掘数据流任意滑动时间窗口内频繁模式[J]. 软件学报, 2008, 19(10):2585-2596.
12. Gibbons P B, Matias Y. New sampling-based summary statistics for improving approximate query answers[J]. Acm Sigmod Record, 1999, 27(2):331-342.

攻读硕士学位期间取得的成果

主要参与的项目：

[1] \*\*\*舆情挖掘项目，\*\*\*部门，2014.09-2015.06

[2] \*\*\*检索与画像，\*\*\*部门，2015.09-2016.06

[3] \*\*\*三化，\*\*\*部门，2016.09-2016.12