**硕士学位论文**

多元时间序列流跨事务关联规则挖掘

**MINING INTER-TRANSACTION ASSOCIATION RULES FROM MULTIPLE TIME-SERIES DATA**

**孙雅琦**

**哈尔滨工业大学**

**2016年12月**

国内图书分类号：TP399 学校代码：10213

国际图书分类号：621.3 密级：公开

**工程硕士学位论文**

多元时间序列流跨事务关联规则挖掘

|  |  |  |
| --- | --- | --- |
| 硕士研究生 | ： | 孙雅琦 |
| 导 师 | ： | 张春慨副教授 |
| 申请学位 | ： | 工程硕士 |
| 学科 | ： | 计算机技术 |
| 所 在 单 位 | ： | 深圳研究生院 |
| 答 辩 日 期 | ： | 2016年12月 |
| 授予学位单位 | ： | 哈尔滨工业大学 |

Classified Index: TP399

U.D.C: 621.3

Dissertation for the Master’s Degree in Engineering

**MINING INTER-TRANSACTION ASSOCIATION RULES FROM MULTIPLE TIME-SERIES DATA**

|  |  |
| --- | --- |
| **Candidate：** | Yaqi Sun |
| **Supervisor：** | Associate Prof. Chunkai Zhang |
| **Academic Degree Applied for：** | Master ’s Degree of Engineering |
| **Speciality：** | Computer Technology |
| **Affiliation：** | Shenzhen Graduate School |
| **Date of Defence：** | December, 2016 |
| **Degree-Conferring-Institution：** | Harbin Institute of Technology |

# 摘 要

随着经济的发展，在很多领域，比如金融，医药，地质，气象，电子商务，传感器网络等都会产生大量的时间序列数据。挖掘时间序列中隐藏的关联关系并且对于后续时间序列进行预测是一个重要的研究方向，对于生产生活具有十分重要的意义。由于时间序列流一般数据量比较大，数据随着时间变化，具有连续性等特点，所以难以采用传统的关联规则挖掘方法直接对时间序列进行有效挖掘。

目前对于时间序列数据流关联规则挖掘已经有一定的研究成果。这些研究多针对单时间序列和事务内关联规则的挖掘。而且通常在已经模式化的时间序列上进行关联规则挖掘，对于多时间序列流的跨事务关联规则挖掘的深入研究比较少，且对于关联规则的研究通常忽略了顺序性，大多不能实现增量挖掘关联规则。

本课题在时间序列预处理阶段采用ITEO的分段线性表示方法，对时间序列数据进行压缩表示。在时间序列聚类时提出IK-Means算法，增加聚类的类间间距，降低了随机选择初始聚类中心的不确定性。本课题采用3个参量（时间长度，数值截距占最低点比率，线段斜率）表示模式，进行归一化度量相似度。在关联规则挖掘过程，设计了基于多时间序列跨事务关联规则的十字链表表示（IAMTL）方法。基于这种存储结构能够表示多条时间序列在固定的时间T内的关联，在时间序列关联规则中加入固定时间T限制，增强关联规则的顺序性。I-IAMTL算法采用修正支持度和置信度，增量的挖掘关联规则，更加符合时间序列数据的特点。在算法挖掘出的关联规则的验证部分采用前后件窗口（PCW）的方法对于关联规则预测的准确性进行了验证。

在实际热电厂的数据和股票数据集上通过将已有算法与本课题的IAMTL和I-IAMTL算法对比，显示了本课题所提算法的有效性。

关键词：多元时间序列；跨事务关联规则；预测；ITEO；IAMTL

# Abstract

With the rapid economic development, there will generate numerous time series data in many different areas, such as finance, medicine, geology, meteorology, e-commerce, sensor networks and other fields. It is a significant orientation of research for mining time series to predict the hidden relation in time series, which is very important for the production and life. Because of the mass , realtime, continuity of time series data stream, it is difficult to use the traditional association rule mining method to mine the time series directly.

At present, there are some research results about the association rules mining of time series data stream. These studies have been studied on the single time series and the mining of association rules in transaction. However, the research on the inter transaction association rules mining of multiple time series is less. And the timeliness, sequential is ignored and incremental mining association rules can not be achieved in most of the research on association rules.

In this paper, a algorithm of piecewise linear representation which is named ITEO is used to compress the time series data. In the time series clustering algorithm, the IK-Means algorithm is proposed to select the initial clustering point. This algorithm not only increases the class spacing between clusters but also reduces the uncertainty of random selection of initial cluster centers. This topic uses 3 parameters (the length of time, numerical intercept divided by minimum point value, slope segment) to represent mode, normalized the similarity measure. In the mining process of association rules, this paper design the (Inter-Transaction Association Rules from Multiple Time-Series List )IAMTL. Based on the storage structure, the association of multiple time series in a fixed time T can be represented, adding fixed time T constraints in time series association rules. It enhances the order of association rules. I-IAMTL algorithm adapts modified support and confidence to mining association rules incremental. The accuracy of the prediction of association rules is verified by the method of Prereqiusite and the Consequent Window(PCW).

On the actual industrial data and stock data, by comparing the existing algorithm with the IAMTL and I-IAMTL algorithm, it indicates the validity of our proposed algorithm.

**Keywords**: multiple time-series data, inter-transaction association rules, prediction, ITEO, IAMTL

目 录

[摘 要 I](#_Toc466356497)

[Abstract II](#_Toc466356498)

[第1章 绪 论 1](#_Toc466356499)

[1.1 课题研究背景及意义 1](#_Toc466356500)

[1.2 国内外研究现状 2](#_Toc466356501)

[1.2.1 传统数据上关联规则的挖掘 2](#_Toc466356502)

[1.2.2 时间序列数据上的关联规则的挖掘 2](#_Toc466356503)

[1.2.3 关联规则在时间序列预测方面的应用 3](#_Toc466356504)

[1.3 本课题的主要研究内容 4](#_Toc466356505)

[1.4 论文的结构安排 5](#_Toc466356506)

[第2章 时间序列关联规则挖掘基础理论 6](#_Toc466356507)

[2.1 时间序列数据 6](#_Toc466356508)

[2.2 时间序列数据的表示方法 7](#_Toc466356509)

[2.3 数据聚类算法 9](#_Toc466356510)

[2.4 关联规则挖掘 10](#_Toc466356511)

[2.4.1 关联规则挖掘的基本概念 10](#_Toc466356512)

[2.4.2 关联规则挖掘的基本算法 11](#_Toc466356513)

[2.5 关联规则应用在时间序列预测 15](#_Toc466356514)

[2.6 本章小结 16](#_Toc466356515)

[第3章 多元时间序列数据的预处理 17](#_Toc466356516)

[3.1 时间序列数据的压缩 17](#_Toc466356517)

[3.2 改进的边缘算子法压缩时间序列 19](#_Toc466356518)

[3.2.1 图像处理中的边缘算子 19](#_Toc466356519)

[3.2.2 时间序列的ITEO表示 21](#_Toc466356520)

[3.3 聚类并符号化生成模式序列 23](#_Toc466356521)

[3.5 本章小结 25](#_Toc466356522)

[第4章 基于IAMTL的多元时间序列跨事务关联规则挖掘 27](#_Toc466356523)

[4.1 多时间序列跨事务关联规则的相关概念 27](#_Toc466356524)

[4.2 基于时间序列的关联规则的挖掘算法 29](#_Toc466356525)

[4.3 时间序列频繁模式的定义 29](#_Toc466356526)

[4.4 多元时间序列跨事务关联规则的挖掘 31](#_Toc466356527)

[4.4.1 基于IAMTL的时间序列跨事务关联规则挖掘算法 31](#_Toc466356528)

[4.4.2 基于IAMTL的时间序列跨事务关联规则的增量挖掘算法 36](#_Toc466356529)

[4.5 多元时间序列关联规则预测序列 38](#_Toc466356530)

[4.6 本章小结 40](#_Toc466356531)

[第5章 实验结果与分析 41](#_Toc466356532)

[5.1 实验数据的描述 41](#_Toc466356533)

[5.2 时间序列数据的预处理 41](#_Toc466356534)

[5.2.1 采用ITEO算法对时间序列进行压缩线性化 41](#_Toc466356535)

[5.2.2 对相似的线段进行聚类并符号化 45](#_Toc466356536)

[5.3 算法性能上的比较 45](#_Toc466356537)

[5.3.1 时间效率上的比较 45](#_Toc466356538)

[5.3.2 内存占用上的比较 47](#_Toc466356539)

[5.3.3 预测时间序列准确率上的比较 49](#_Toc466356540)

[5.3.4 算法在不同数据集上的性能比较 51](#_Toc466356541)

[5.4 本章小结 53](#_Toc466356542)

[结 论 54](#_Toc466356543)

[参考文献 55](#_Toc466356544)

[攻读硕士学位期间发表的论文及其它成果 59](#_Toc466356545)

[哈尔滨工业大学学位论文原创性声明和使用权限 60](#_Toc466356546)

[致 谢 61](#_Toc466356547)

# 第1章 绪 论

本课题研究多元时间序列流跨事务关联规则的挖掘。事件的发生通常会有关联关系，通常一些事情的发生会导致其他事情的发生。关联规则挖掘是通过挖掘这些关联关系来预测事件的发生，从而更好的了解事件的发展趋势。到目前为止，针对时间序列关联规则的挖掘的研究主要是单时间序列，或多时间序列事务内关联规则的挖掘。而多元时间序列流跨事务关联规则的挖掘研究不同时间段不同模式之间的关联关系，对于后续时间序列的趋势的预测具有重要的意义。在本章中首先介绍本课题的研究背景及意义，从而明确本课题的研究的作用和意义；然后，对于国内外研究中提出的关于时间序列关联规则挖掘的算法进行简要的介绍，分析已有算法的特点。最后是课题主要研究内容，论文的结构安排。

## 1.1 课题研究背景及意义

随着科技的进步，经济的发展，时间序列广泛的存在于我们的生活中，越来越多的数据基于时间标签[1]记录对象和有关现象的变化。工业生产中的温度热度等传感器产生的数据[2]，水文时间序列[3,4]，金融方面股票数据[5-7]，医学方面心电数据[8,9]都是生产生活中常见的时间序列数据。对于时间序列中蕴藏的信息则需要数据挖掘技术来帮助我们理解。数据挖掘真正的目的是对数据进行分析，以获取现在未知的有趣的模式。挖掘时间序列中的有用的信息和知识对于用户更好的理解事务现象和行为活动有很重要的意义。

在现实生活中，时间序列的分析通常是在多元的时间序列流数据上进行的。我们考虑的关联规则中，可能是两个来自同一条序列的模式，也可能是来自不同序列的不同事务中的两个模式，这就是跨事务的概念。跨事务的关联规则含义包括两个方面。一方面事务可以来自不同时间序列，另一方面可能发生于不同的时间段。如：平安银行某天上涨，浦发银行两天后下跌中国银行在三天后上涨（10%，70%），其中括号中的数字代表关联规则的度量标准，支持度和置信度，后续会详细介绍。这条股票数据的关联规则前后件的发生是具有先后顺序并且具有时间跨度，通过关联规则我们可以在一定程度上预测后续发生的事件。如：南太平洋海表面温度的逐渐增加，往往伴随着美国西部的降雨量急剧增加。这些关联规则通常假设一个基本的符号，或者命题表示，而我们更感兴趣的是由实际数值组成的时间序列。

本课题挖掘时间序列中的关联规则。因为时间序列流数据具有的一系列的特性，如时间序列流随着时间实时产生决定我们不能像传统的关联规则挖掘一样对数据完全存储后进行多次遍历之后进行挖掘。在挖掘关联规则时，先预处理原始数据形成模式序列的形式。然后在模式序列上挖掘关联规则。本课题利用时间限制窗口T对流数据进行约束和限制，将一个时间窗口内的多个时间序列的模式作为一个事务集来进行关联规则的挖掘。在时间序列分段线性化表示部分基于数字图像处理中边缘检测思想提出改进的时态边缘算子算法（ITEO）选取时间序列中的关键点。线性化之后的线段采用三个参数表示，进行归一化处理。并采用IK-Means方法对归一化后的模式进行聚类，将原始的时间序列线性化为模式序列。提出增量的挖掘跨事务关联规则算法，更具有现实意义。在算法挖掘出的关联规则的验证部分采用前后件窗口（PCW）的方法对于关联规则预测的准确性进行了验证。

本文的研究过程分为时间序列数据的预处理，关联规则的挖掘，以及对于关联规则有效性的验证。

## 1.2 国内外研究现状

### 1.2.1 传统数据上关联规则的挖掘

在传统数据集上的关联规则的挖掘的问题中，最常见的是购物车问题。最经典的算法[10]采用广度优先搜索策略计算项集支持度。算法使用一个候选生成函数，利用了支持度的向下封闭性。Eclat[11]是一个利用集合交集的深度优先搜索算法。它是一个自然的算法，比较适用于两个并行顺序产生的数据集合。算法是由Zaki等人在1997年第一次提出的。AprioriDP算法[12]是在频繁模式挖掘中运用动态规划的思想。算法的思想是消除候选项集，用特殊的数据结构保存支持度计数。FPA算法[13]，是一种基于矩阵的算法。这种算法能高效的挖掘出事务数据集中的最大频繁模式，设计一种关联矩阵，将数据的关联信息映射到矩阵中，只需要扫描一次数据集，不会产生候选项，从一定程度上改善了算法的效率。

### 1.2.2 时间序列数据上的关联规则的挖掘

传统的关联规则挖掘算法针对的是离散数据，挖掘出的关联规则并不能体现时间的先后顺序。第一次应用在时间序列上的关联规则的挖掘算法的提出是在1998年Das提出的[14]。研究对象是从单时间序列挖掘的关联规则开始，后来扩展到多时间序列的挖掘。在处理时间序列数据时，把时间序列划分成长度相等的子序列，然后为每个趋势不同的子序列分配一个符号表示。这种算法关注的子序列的趋势主要有三种，上升，下降，平衡。因此，对于不同时间长度，趋势相同的子序列并不能区分。后来学者把算法[15]应用在时间序列关联规则挖掘方面。算法是一种高效可扩展的算法，通过模式增长的方式，用扩展前缀树结构，这个概要存储结构[16]用于存储关于频繁模式的压缩和关键信息，在很多情况下都比效果更好。后来在此基础上又出现了很多改良的算法。CFP-mine[17]算法是基于压缩的FP树，基于约束子树的方法，减少内存的调用，并且用了数组的方法，减少了遍历次数。ITARM算法[18]采用分治的挖掘事务间关联规则的方法，在CFPmine的基础上进行改进，对挖掘的任务进一步分解。ITF-tree[19]是在FP-growth算法上改进了概要存储结构。将经典的算法应用在时间序列挖掘上的tsapriori算法[20]通过将一个限制窗口中的数据看做一个事务集，然后运用Apriori算法挖掘关联规则。

### 1.2.3 关联规则在时间序列预测方面的应用

时间序列的关联规则可以应用在预测方面。传统的时间序列预测多用于预测未来连续的数值。但随着数据量的增多，运用海量的数据信息对连续数值的预测并不适用，用户更多的关心数据在某段时间的趋势。所以在一些研究中常常把连续的数值转化为模式，不需要对未来连续数据进行精准预测。在时间序列方面关联规则预测方法更多的应用于股票的预测。例如在股票预测中，模式可能表示直线上升，折线下降等趋势，假设A股票上涨是模式，B股票下跌是模式*M*2，通过关联规则，则可以一定概率的推测A股票上升则在时间内B股票会下跌。这些模式可以对投资者的投资提供决策帮助。再例如在点击流预测中，模式可能表示用户点击的网页，图片，广告等，记录点击时间，可以按照时间顺序将这些模式排列成一个时间序列。这种模式序列算法是根据一组按照时间排序的，离散的，非数值的模式例如，其中在不同的研究中模式表示的含义是不同的，根据挖掘到的关联规则来预测未来模式。这类的模式预测可以用于推荐系统中，根据过去的用户数据如点击，喜好等时间序列数据，向用户推荐新的网页，产品等。

Nair[21]利用符号聚合近似（SAX）方法在股票的数值型数据提取出模式，再使用遗传算法优化的Apriori算法挖掘频繁模式，再把这些频繁模式用于股票交易推荐系统，这样可以从股票的交易价格数据中挖掘时间序列关联规则，从而生成股票的交易建议。Seema使用树结构SMAP-TREE来存放用户位置信息和访问信息，再通过搜索树来挖掘和预测未来访问页面，把用户可能访问的网页缓存起来从而提高WEB性能[22]。Vincent提出了运用时间序列关联规则在移动网络系统中，通过Apriori算法挖掘用户访问日志中的访问时间，IP地址，浏览器信息，再使用马尔科夫链的方式预测用户下一步访问行为，从而推荐给用户及时有效的信息[23]。Rudin提出使用选择修正置信度的方式来选择关联规则算法挖掘出的频繁项集，使用这种方式选择到的关联规则可以提高预测的准确度[24]。

## 1.3 本课题的主要研究内容

（1）时间序列关联规则挖掘相关的问题的基础研究。

首先是对于时间序列数据的特点进行了研究。然后研究传统的关联规则挖掘算法，以及在时间序列数据上的关联规则的挖掘。本课题主要研究的是时间序列上跨事务的关联规则挖掘，反映不同时间段不同时间序列之间的关联关系。

（2）时间序列的预处理

基于时间序列数据量的特点，在挖掘关联规则前有必要对数据进行压缩预处理，转化成模式序列的形式。时间序列的压缩可以提高后续挖掘的效率，而且压缩能够将数据转化为模式序列，为后续工作做好准备。本课题采用线性化的方式对时间序列进行压缩。时间序列预处理首先是数据的分段线性化表示，根据数字图像处理中的边缘算子思想，结合时间序列的数据特点，提出了ITEO算法。根据ITEO算法选取时间序列的分段点，将分段点连接起来形成线段。完成分段线性化之后进行模式的聚类符号化，将原始的时间序列转化为模式序列的形式。经过以上处理步骤后，预处理阶段完成。

（3）时间序列关联规则的挖掘。

经过预处理之后，原来连续的数据转变为模式序列，下一步是关联规则挖掘。时间序列的关联规则与传统的购物车关联规则有很大不同，时间序列关联规则需要考虑时间的限制，时间长短以及时间间隔的限制。并且对于时间序列数据上的关联规则挖掘需要考虑数据的实时产生性，增量的处理数据显得至关重要。本课题提出两种适用于不同情况下的关联规则挖掘算法，更加适用于时间序列数据海量和实时产生的实际特征。

（4）时间序列关联规则的验证。

从时间序列中挖掘出关联规则，根据修正支持度筛选出关联规则，但是这些关联规则对于后续时间序列的预测效果需要验证。本课题采用最小支持度阈值挖掘关联规则，选择按照时间衰减的修正置信度排序的关联规则，利用前后件窗口（PCW）方法对关联规则预测做出验证。

## 1.4 论文的结构安排

本课题的结构安排如下，按照章节划分：

（1）第一章主要讲述了本课题的研究目的意义，国内外研究现状，课题的主要研究内容。国内外研究现状主要包括对关联规则挖掘的研究现状，对于时间序列分析的研究现状，对于本课题的研究延伸。本课题在研究中用到的算法，研究流程。

（2）第二章主要讲述时序关联规则挖掘的基础知识。对时间序列数据进行了介绍，然后介绍了课题对于时间序列的预处理，时间序列线性化表示的方法，传统的聚类的方法的总结。然后介绍了关于关联规则挖掘方面的基础知识。包括相关概念，传统的挖掘算法，时间序列上的关联规则的挖掘。最后介绍关联规则在时间序列预测方面的应用。

（3）第三章主要讲述在时间序列数据预处理阶段所做的工作。基于时间序列数据的海量性，实时产生等特征，直接在原始的时间序列上挖掘是不现实的，因此在挖掘关联规则之前需要对它进行预处理，转化成模式的形式，便于后续时间序列关联规则的挖掘，以及对于关联规则的解释与评价。时间序列的预处理过程包括分段线性化表示（压缩），聚类符号化表示形成模式序列。

（4）第四章主要在第三章已经进行预处理过的数据上进行跨事务关联规则的挖掘。本课题设计了双头结点的链表表示方法，表示固定的时间T内的多时间序列的跨事务关联规则，划分固定长度的事务集。在内存中维护一个基于多时间序列的跨事务关联规则链表IAMTL（Inter-Transaction Association Rules from MultipleTime-series List）。提出了基于IAMTL进行多元时间序列跨事务关联规则的挖掘，以及增量的挖掘算法I-IAMTL。然后介绍了将关联规则应用在时间序列预测方面，采用时间衰减的置信度筛选关联规则，用前后件窗口（PCW）的方法验证时间序列预测的准确度。

（5）第五章是实验部分。数据集采用的是热电厂工业数据，通过与tsapriori算法在运行时间，算法占用内存的性能方面，以及对后续时间序列的预测准确度方面进行对比。

最后是结论部分，描述本课题的研究成果，创新点，以及算法需要改进和创新的地方。

# 第2章 时间序列关联规则挖掘基础理论

本章主要介绍在时间序列的关联规则挖掘中要用到的基本理论知识。先对时间序列数据进行介绍，然后介绍课题对时间序列的预处理，时间序列线性化表示的方法，传统的聚类方法。然后介绍关联规则挖掘的基本概念，传统的关联规则挖掘方法，现有的基于时间序列关联规则挖掘方法。最后介绍关联规则应用在预测方面的基本知识。

## 2.1 时间序列数据

**定义2-1** 时间序列。时间序列是指依照事件发生的顺序把数值排列而成的数列。统计中所定义的时间序列，是指在时间序列和时间序列中随机事件的发展过程。时间序列表示形式如图2-1所示，横轴表示时间，纵轴表示随时间变化的数值。

T

Value

图2-1 长度为40的单时间序列数据

**定义2-2** 多元时间序列分析。多元时间序列是指在同一时间点对应多个数据值，多条时间序列之间的关联关系则是多元时间序列的关联规则。如图2-2所示。

Value

T

图2-2 长度为30的三元时间序列数据

## 2.2 时间序列数据的表示方法

由于时间序列数据量大，实时产生等特点，在挖掘关联规则之前需要对数据进行压缩，也即特征表示。时间序列的特征表示是提取数据的特征，转换数据的维度。这样能达到对特征降维的作用。同时，在低维空间中的数据还可以尽可能的保留原始时间序列的信息。几种特征表示方法如图2-3所示。

时间序列数据的特征表示方法有很多种。大致可分为数据的非自适应，数据的自适应，基于模型的方法[25]。

数据的非自适应方法是指将数据转换到另一个数据空间。其中转换进程和特征系数选取独立于数据自身。其中聚合近似和基于域变的方法是两种典型的方法。分段聚合近似（PAA）[26,27]通过平均分原始时间序列，然后利用每段的平均值的序列来表示原始时间序列。其中分段的数目由人为规定，每个时间序列的分段长度都是相等的。因此这种方法的表示效果依赖于分段数参数，很显然如果分段数过小，会遗漏原时间序列过多的信息，时间序列的表示会过于粗糙。相反如果分段数过大，虽然能避免损失原来时间序列的信息，但是降维幅度会变小。基于域变换的时间序列表示方法根据信号处理方法实现时间序列的时域和频域的变换。然后，它使用在频域中的有限的特征数据来近似原来的时间序列。域变换方法中比较典型的有离散傅里叶变换（DFT）[28]和离散小波变换（DWT）[29]。他们既有联系又有区别。基于域变换的表示方法的系数多难为用户所直接理解。

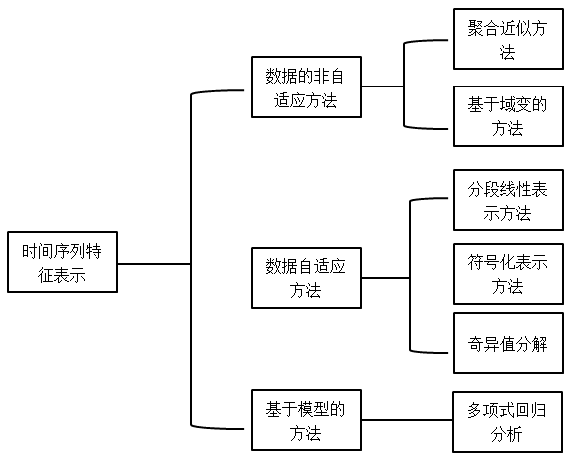


图2-3 时间序列的特征表示方法

数据的自适应方法主要有分段线性表示，符号化表示，奇异值分解[30]。分段线性表示方法是应用相对普遍的表示方法。通过不同的线性化方法把复杂的时间序列数据以直观的线段形式表示。分段算法包括全局和局部两种。全局算法以全体时间序列数据为基点找分段点。局部算法，根据局部特性，从左到右，符合分割的要求。全局的分段算法有自顶向下算法，自底向上算法。自顶向下算法将数据起点终点连接，看做初始的拟合线段。然后遍历时间序列寻找距离初始线段最远的点，度量此距离与规定的阈值的大小，若是大于阈值则把该点归入划分点集合。不断循环迭代，直到达到分段数，或者压缩比超过阈值，或者误差超过范围。自底向上算法首先将相邻的两个数据点相连。然后计算两线段连接拟合误差，如果小于规定的误差阈值则合并两条线段。不断迭代重复此过程直到连接相邻线段的拟合误差都不小于阈值，算法结束。符号化表示方法就是将时间序列用符号序列表示。奇异值分解方法（SVD）[31]利用主成分分子方法作为驱动引擎，将高维时间序列降低为低维。这种方法通常应用于文本挖掘，模式识别等领域。

局部算法中比较常用的有固定窗口算法，用固定大小的窗口划分时间序列进行拟合时间序列。其中窗口的大小可以直接用阈值R1来限定，也可以规定要划分的时间序列的段数，间接获得窗口的大小。滑动窗口法首先将时间序列分为固定大小的基本窗口。因为时间序列数据具有时效性，滑动窗口内的数据表示近期某个时间段内的数据，包含多个基本窗口的数据。利用滑动窗口的滑动动态处理数据。

基于模型的方法是先假定时间序列符合某个模型，如回归模型，隐马尔科夫模型，概率产生器，基于推理文法模型，神经网络等[32,33]，利用模型参数表示数据。基于正交多项式表示法的最小二乘法结合正交多项式。广泛应用于时间序列分割和主题发现领域。然而这种方法由于具有很多缺陷并没有在实际中得到广泛的应用。比如时间序列往往不能由一个模型产生，拟合效果比较差。而且对于训练集的数据量要求量比较大。

## 2.3 数据聚类算法

聚类是人类重要的一种行为，我们从小就开始根据事物的特征不断改进认知模型来区分身边的动物，植物等。这是无监督学习也就是聚类。没有类别标签，将相似性大的样本归入一类，不同类中的差异尽可能的大。聚类在数据挖掘中是十分重要的，广泛应用于科研领域。传统的聚类算法主要可以划分为以下几种。基于划分的方法，基于层次的方法，基于密度的方法，基于网格的方法，基于模型的方法[34]。

（1）K-Means算法[35]是一种基于划分的方法。算法第一次的聚类中心是随机的，后面每一步迭代用均值表示聚类中心。这种聚类算法缺点十分明显，聚类中心点选择影响结果。其中距离的度量方法是距离，距离实际是一系列距离的集合。例如两个时间序列和，二者的距离计算公式为（2-1）：

|  |  |  |
| --- | --- | --- |
|  |  | (2-1) |

公式中的P为2时，距离就是欧式距离。

（2）层次聚类法[36]中常用的是凝聚型聚类算法，通过计算每两个数据之间的距离，不断的选择最小距离的类凝聚为一个。如此迭代，直到满足一定的条件时，算法结束。这类算法不需要事先规定聚类数，但是计算复杂度很高，每次都需要计算类中两两之间的距离。

（3）基于密度的聚类方法就是将距离近并且密度满足一定阈值的区域合并。DBSCAN算法[37]是其中应用最普遍的方法。这种算法的优点是聚类效率高可以聚类空间数据。算法的缺点是，聚类初始参数邻域半径和E邻域最小点数需要提前设定。聚类的结果与这两个参数有很大的关系。

（4）上述聚类算法都是基于距离的。基于距离的聚类算法并不适用于处理大数据。而且聚类的个数也是不确定的，与参数设定有关。其中STING算法[38]是最典型的基于网格的聚类算法，是将样本空间划分为有限数量的元素。

（5）传统的聚类方法中样本所属类别是固定的，模糊聚类是一种新型的聚类方法。模糊聚类中每个数据点都可以属于一个以上的类。每个数据点都会分配一个隶属度等级。这些成员级别表示数据点属于每个群集的程度。因此，在一个聚类的边缘上的点相比聚类中心的数据会有会有较低的隶属度等级。FCM[39]是典型的模糊聚类算法。这种算法的流程是通过不断迭代，将目标函数收敛。根据迭代的结果，确定聚类结果。

## 2.4 关联规则挖掘

关联规则通过分析数据之间的关联关系，对人类生活做出指导。目前，关联规则广泛应用于金融，电子商务领域的推荐算法，预测等。不同的算法针对的数据对象是不同的。传统的关联规则挖掘算法多应用在购物篮分析问题，对于时间序列的关联规则挖掘算法尤其是多元时间序列的深入研究较少。

### 2.4.1 关联规则挖掘的基本概念

假设数据库T是事务集合，关联规则是形如的式子。其中，和分别是关联规则的前件和后件。在有的文献中也被称为先导和后继。首先进行的频繁模式的发现就是从事务集中找到出现次数大于某一规定阈值的模式集合，或是其支持度大于最小阈值。关联规则挖掘则是在频繁模式的基础上发现关联关系。评价关联规则的支持度和置信度，定义如下。

**定义2-3** 对于关联规则，它的支持度（*Support*）表示同时含有和事务的个数占T中事务总个数的比例，如公式（2-2）所示。

|  |  |  |
| --- | --- | --- |
|  |  | (2-2) |

其中，表示在2事物集*中*同时出现在一个事务中的数量，表示中的事务集的总的数量。

**定义2-4** 对于关联规则在数据集*T*中的置信度(*Confidence*)是一个条件概率。表示在发生的时候发生的概率，用公式（2-3）表示。

|  |  |  |
| --- | --- | --- |
|  |  | (2-3) |

其中表示数据集中包含项集的事务的数量。

关联规则由和两个概率指标来体现。因此在挖掘关联规则之前需要提前设置这两个参数的阈值。以下支持度阈值用表示。置信度阈值用表示。如果关联规则满足和，则认为关联规则是用户要寻找的。

### 2.4.2 关联规则挖掘的基本算法

Apriori算法[10]是最经典的传统数据挖掘算法。Apriori算法主要有两步。

首先，找出数据库中的所有一阶频繁项集。这些项集的支持度大于。第二步，利用频繁项集组合出大于的关联规则。由于频繁项集的子集也必须是频繁的。Apriori算法根据这个原则进行剪枝。

下面是算法的具体流程：

假设有项集,有事务集，如表2-1。

表2-1 事务集T

|  |  |
| --- | --- |
| 事务 | 项集 |
| T1 | A,B,C |
| T2 | A,B,D |
| T3 | A,C,D |
| T4 | A,B,C,E |
| T5 | A,C,E |
| T6 | B,D,E |
| T7 | A,B,C,D |

假设规定的是3/7；最小是5/7。则可以得到：

1-频繁项目集:{A},{B},{C},{D},{E}

然后根据1频繁项目集组合成2元素的项目集：

{A,B},{A,C},{A,D},{A,E}

{B,C},{B.D},{B,E}

{C,D},{C,E}

{D,E}

计算这些项目集的支持度得到满足最小支持度阈值的只有{A,B},{A,C},{A,D},{B,C},{B,D},{B,E}。因此得到2-频繁项目集{A,B},{A,C},{A,D},{B,C},{B,D},{B,E}。

首先根据2-频繁项集生成包含3个元素的项目集。

{A,B}{A,C}两个项集合并得到{A,B,C};

{A,B}{A,D}两个项集合并得到{A,B,D};

{A,B}{B,C}两个项集合并得到{A,B,C};

{A,B}{B,D}两个项集合并得到{A,B,D};

{A,B}{B,E}两个项集合并得到{A,B,E}。

计算包含3个元素的项集的支持度之后发现只有{A,B,C}满足条件。因此3-频繁项集是{A,B,C}。

然后根据频繁项集生成关联规则。

 *Confidence* = 3/4

 *Confidence* = 3/5

 *Confidence* = 3/3

置信度不满足要求所以剔除。

因此得到1后件的集合是，然后再以（A,C）作为后件 *Confidence=*3/5不满足最小置信度。

因此对于3频繁项集生成的强关联规则为和。

传统的关联规则挖掘算法Apriori算法，广泛应用在购物篮问题中。在海量数据情况下，Apriori算法需要生成候选项集，会耗费大量的存储空间。因此算法[40]被提出来。

FP-growth算法的伪代码如表2-2和表2-3所示。

表2-2 FP-growth算法建树

|  |
| --- |
| 算法：建造FP树  输入：trans, min\_support  输出：FP树 |
| 1. freq\_set1 = find\_feq\_set(trans, min\_support) // 获取一阶频繁项集 2. tree = new\_tree() // 3. for tran in trans // 逐条从事务集中取出事务 4. new\_tran = sort\_by\_support(freq\_set1, tran) // 根据一阶频繁项集中的支持度，对事务排序 5. insert\_node(tree, new\_tran) // 将事务插入FP树 |

表2-3 FP-growth算法

|  |
| --- |
| 算法：FP-growth算法  输入：tree FP树或条件FP树，item 候选项  输出：模式 |
| 1. freq\_set = [] 2. if count\_branch(tree) == 1 // FP树只有一个分支 3. for b in combine(tree.branch[0]) // 遍历分支中节点的每种组合 4. freq = union(b, item) // 形成模式 b∪item 5. freq\_set.append(freq, support=min\_support(b)) // 使用分支节点b中最小支持度 6. else // 一般情况 7. for a in tree.head\_list // 遍历树的头结点中的每个项 8. b = union(a, item) // 形成模式 9. b.support = a.support 10. cb = condition\_base(b) // 构造条件基 |

表2-3（续表）

|  |
| --- |
| 算法：FP-growth算法  输入：tree FP树或条件FP树，item 候选项  输出：模式 |
| 1. c\_tree = condition\_tree(cb) // 形成条件FP树 2. if c\_tree is not empty 3. FP\_growth(c\_tree, b) // 递归调用 |

假设存在事务集H，构成如表2-4。

表2-4 事务集H

|  |  |
| --- | --- |
| 事务 | 项集 |
| H1 | I1,I2,I5 |
| H2 | I2,I4 |
| H3 | I2,I3 |
| H4 | I1,I2,I4 |
| H5 | I1,I3 |
| H6 | I2,I3 |
| H7 | I1,I3 |
| H8 | I1,I2,I3,I5 |
| H9 | I1,I2,I3 |

第一步根据事务集构造FP-tree

扫描事务集数据库得到频繁1项目集F，如表2-5所示。

表2-5 频繁1项目集F

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| I1 | I2 | I3 | I4 | I5 |
| 6 | 7 | 6 | 2 | 2 |

这里我们定义=0.2, 为2.

对F按照支持度重新排列得到排列后的频繁1项目集如表2-6所示。

表2-6 排序后的频繁1项目集F

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| I2 | I1 | I3 | I4 | I5 |
| 7 | 6 | 6 | 2 | 2 |

根据排序的结果对事务数据库进行排序，结果如表2-7所示：

表2-7 排序后的事务数据库

|  |  |
| --- | --- |
| 事务 | 项集 |
| H1 | I2,I1,I5 |
| H2 | I2,I4 |
| H3 | I2,I3 |
| H4 | I2,I1,I4 |
| H5 | I1,I3 |
| H6 | I2,I3 |
| H7 | I1,I3 |
| H8 | I2,I1,I3,I5 |
| H9 | I2,I1,I3 |

创建FP-tree结构如图2-4。

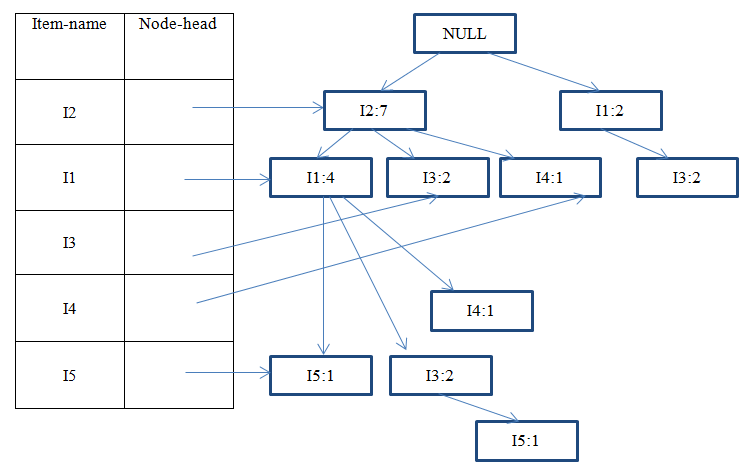


图2-4 创建FP-tree结构

首先根据头节点链表顺序首先从I5开始。得到条件模式基：。冒号后面的数字表示频度。其中条件模式基是指包含树中与后缀模式一起出现的前缀路径的集合。构造条件FP-tree如图2-5.

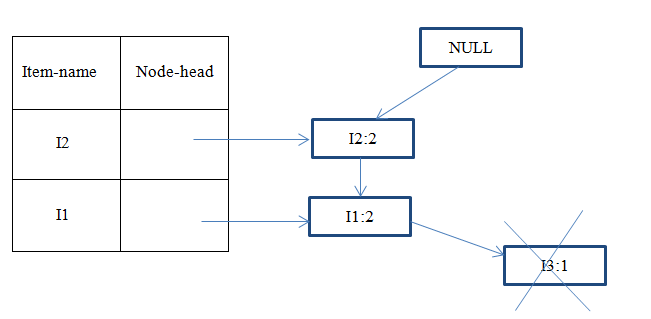


图2-5 I5的条件

得到I5的频繁项集：。.

然后分析I4，得到条件模式基：，如图2-6所示。

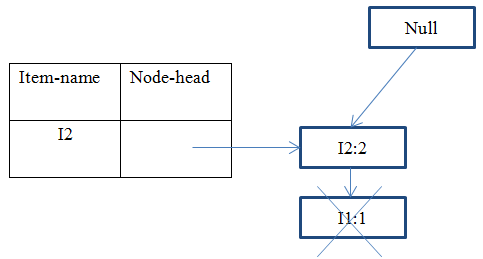


图2-6 I4的条件

得到I4频繁项集：。

然后得到I3的频繁项集 , I1的频繁项集。

## 2.5 关联规则应用在时间序列预测

关联规则预测多数采用置信度大的规则来预测模式。关联规则的置信度是条件概率，描述出现A模式之后会有多大的概率出现B模式。用这种方法来验证关联规则的预测准确性是一种合理的方法，关联规则的预测具有良好的解释性，可以理解为某个事件的发生导致另一个事件的发生。

修正置信度算法由Rudin于2011年提出[41]，使用关联规则来预测未来可能出现的模式，用于预测未来用户可能购买的商品。传统的关联规则挖掘步骤使用最小支持度进行剪枝，修正置信度算法并不采用最小支持度对关联规则进行过滤。因为有的关联规则的支持度和置信度的大小并不是成正比的。如果使用严格的支持度会过滤掉出现次数稀少但置信度非常高的规则，这些关联规则在预测中往往非常重要。修正置信度算法在查找到所有事务的规则之后，使用修改的置信度公式如公式（2-4）排序。在预测步骤可以采用选择最大的修正置信度的方法对下一个要出现的事务进行预测。

|  |  |  |
| --- | --- | --- |
|  |  | (2-4) |

使用修正的置信度算法提供一个变量K，这样可以选择到出现次数少但是非常准确的规则，可以在一定程度上增强了预测的准确度。算法的伪代码如表2-8所示。

表2-8 Adjust-Confidence算法

|  |
| --- |
| 算法：Adjust-Confidence算法  输入：关联规则库Rule、修正参数K、当前购物车中数据Basket  输出：下一个模式 |
| 1. for item in Basket: 2. y += find\_rules(item, rules) // 找到前项为item的关联规则 3. y = sortByConf(y, K) // 根据修正置信度排序 4. y\_list = y[0…k]// 提取前k个规则 5. return y\_list |

传统的关联规则算法多应用在购物篮分析问题，忽略时间因素的存在，对于时间序列的关联规则的挖掘算法尤其是多元时间序列的关联规则的挖掘时间因素往往是十分重要的。如在购物数据的推荐算法中，用户的喜好往往随着时间变化的。

## 2.6 本章小结

本章首先对时间序列的数据进行了介绍。然后根据关联规则的挖掘，先对数据进行预处理，主要包括线性化，聚类，形成模式序列。预处理之后是关联规则的挖掘。本章介绍了相关的基本知识，采用和为指标来衡量。本章介绍了两种经典的算法的基本流程。并且介绍了一种使用关联规则预测模式的修正置信度算法。由于经典的算法主要针对购物篮等离散数据问题，对于时间序列事务间的关联规则，则需要考虑时间先后顺序等因素，传统的关联规则挖掘算法不能实现此目的。

# 第3章 多元时间序列数据的预处理

在实际的生产和生活中我们获得的时间序列数据具有实时产生，海量性，随机波动性，数据密集性等特征，而我们要获取的关联规则隐藏在海量数据中，只有通过数据挖掘才能得到。基于时间序列数据的这些特征，直接在原始的时间序列上挖掘是不现实的，因此在挖掘关联规则之前需要对它进行预处理，转化成模式的形式，便于后续时间序列关联规则的挖掘，以及对于关联规则的解释与评价。时间序列的预处理过程包括时间序列压缩，聚类线段化后的时间序列，符号化表示形成模式序列。

## 3.1 时间序列数据的压缩

时间序列关联规则的挖掘就是在时间序列的模式序列中发现其中的关联关系，从而对人们的生产和生活决策起到指导作用。首先需要将连续的时间序列转化为模式的形式。时间序列的压缩是将原始序列划分为子序列，然后使用模式来表示每个子序列。这里的模式在不同的研究中有不同的定义。模式可能是用模型表示时间序列时的参数，平均值，方差等。有的研究中用增减趋势表示模式，或者线段的斜率。

在以往的时间序列压缩中，很多研究采用模型压缩的方法，运用单一的模型来表示时间序列，但是时间序列数据往往具有随机波动性，采用单一的数学模型来描述整个时间序列很难反应时间序列的变化趋势。如图3-1所示，如果采用最小二乘法拟合成直线，与原始的时间序列的变化趋势有很大的不同，与实际情况不符合。这种直接用最小二乘法的线段拟合会损失很多原始时间序列的特征信息。

图3-1 最小二乘法拟合时间序列

设时间序列为,模式化后的包含若干个模式。通过时间序列压缩可以把原始的时间序列分割表示成公式（3-1）的形式。

|  |  |  |
| --- | --- | --- |
|  |  | （3-1） |

其中指的是第i个模式与第i段时间序列的拟合误差，这种把时间序列分割之后，表示成模式的形式叫做时间序列的压缩。当然，上述的表示形式中有多种形式，可能是直线，二次多项式，n次多项式，根据实际需要选择。根据时间序列的压缩的定义以及意义我们很容易看到，在时间序列的压缩过程中分割点的选择，模式的表示形式是两个至关重要的点。在本文献中模式用线段表示。后面聚类符号化模式时详细介绍线段模式的特征表示。通过时间序列的压缩过程，将海量数据的时间序列划分为模式序列，方便后续关联规则挖掘。

时间序列压缩的方法中常用的有均值法，误差法，特征点法，多项式法等方法[42]。均值法是指将整个时间序列平均分成若干子序列，用每段时间序列的均值来表示这段子序列，从而达到特征降维的目的。这种方法比较简单，在时间序列波动变化幅度比较小的时候也可以反映出序列的变化趋势等特点。但是对于分割的段数，需要人为规定，参数很难确定。并且在时间序列数据波动幅度很大的情况下，分段平均值压缩法会损失很多原始时间序列的特征信息。误差法以误差阈值对时间进行分段和拟合。其中又分为滑动窗口法，自顶向下法，自底向上法。而误差的度量有三种不同的形式累计误差,平均误差，最大误差。假设数据被分成n段，每段的误差表示为，则三种误差的计算公式如公式（3-2），（3-3），（3-4）所示。根据实际情况选择不同的误差计算方式。

|  |  |  |
| --- | --- | --- |
|  |  | (3-2) |
|  |  | (3-3) |
|  |  | (3-4) |

特征点法都是根据不同的标准选取时间序列中的部分点作为分割点，连接分割点进行时间序列的压缩和线性化。不同的研究其重要点和关键点的定义不同。通常选择时间序列的起点终点，极值点，变化幅度比较大的点。而有些研究从局部极值点中去掉一些不重要的局部极值点作为特征点。不重要的局部极值点指的是两个相邻的极值点时间间隔很小并且数值相差不大。这其中涉及到时间间隔的阈值，数值差距的阈值，都需要人为设置。分段多项式压缩法就是用多项式来拟合子序列，用多项式的系数表示每段子序列。但是这种算法的多项式的阶数很难确定，如果用相同的阶数的多项式拟合不同的子序列，则会损失太多时间序列的信息。

## 3.2 改进的边缘算子法压缩时间序列

通过分析已有的时间序列压缩算法的优劣，根据数字图像中边缘识别[43]的启发提出了改进的边缘算子（Improved Temporal Edge Operator, ITEO）。

### 3.2.1 图像处理中的边缘算子

在数字图像处理中，可以根据边界的灰度变化检测边缘。，，，是四种经典的边缘检测算子，普遍应用于图像处理的研究中。

边缘算子用差分计算寻找边缘。它由公式（3-5）给出：

|  |  |  |
| --- | --- | --- |
|  |  | (3-5) |

其中f(x,y)是输入图像。

Sobel算子由两个卷积核对原图像f(x,y)进行卷积计算获得。如图3-2所示，Sobel边缘算子是通过要检测的图像模板先分别与边缘检测算子模板卷积相乘。然后将得到的结果进行微分计算。计算公式如公式（3-6）-（3-8）。

|  |  |  |
| --- | --- | --- |
|  |  | (3-6) |
|  |  | (3-7) |
|  |  | (3-8) |

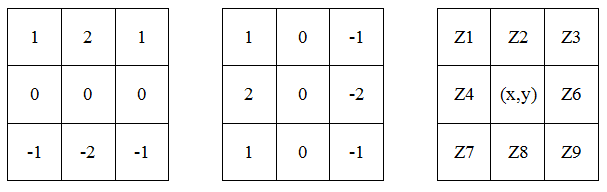


图3-2 Sobel边缘算子

使用边缘检测算子进行线检测，通过比较像素值与模板的卷积计算值确定要检测的点的方向。

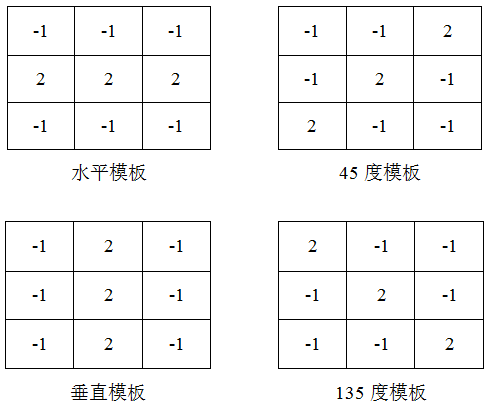


图3-3 四个方向的边缘算子

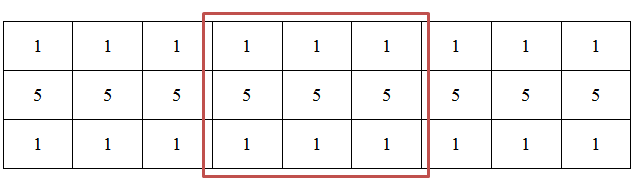


图3-4 要检测的图像

用图3-3中的4个方向的中模板分别与图3-4中方框中要检测的图像像素进行卷积运算得到4个方向上的边缘值。

R水平方向=-6+30=24；

R45度方向=-14+14=0；

R135度方向=-14+14=0；

R垂直方向=-14+14=0

比较得到除了水平方向上的边缘值非0，其余方向上的边缘值都为0。因此图像是水平方向上的边缘。

### 3.2.2 时间序列的ITEO表示

分析时间序列的数据特点，在分段点的两边往往会有不同的变化趋势类比图像处理中图像的边缘的灰度的变化。在图像的边缘处，图像点的灰度变化率会发生变化。若时间序列上某点之前的数据有增长的趋势，该点之后的数据有减小的趋势，则在一定程度上可以认为这个点是分段点，也就是时间序列的边缘点。因此将图像处理中的边缘检测算子与时间序列数据特点结合提出了ITEO表示的方法。

时间序列的ITEO表示是一种分段线性化表示，根据设计的时间序列边缘算子与原始的时间序列数据进行卷积的计算结果。然后根据确定的选择原理从计算的边缘度结果中选择分段点，连结分段点来表示时间序列。时间序列的表示形式是，定义如公式（3-9）。

|  |  |  |
| --- | --- | --- |
|  |  | (3-9) |

其中2u+1表示检测窗口的长度。表示的权重函数，选择的依据是数据的特征。本课题采用的权重设置方法是越靠近检测窗口中心权重设置的越高。ITEO（Improved TEO）设计权重函数为公式（3-10）表示的形式。

|  |  |  |
| --- | --- | --- |
|  |  | (3-10) |

时间序列,其分段点为，那么时间序列的分段表示的形式如（3-11）。

|  |  |  |
| --- | --- | --- |
|  |  | (3-11) |

其中L(x,y)表示边缘点x和y之间的连线，用线性函数表示。函数的阶数不确定。采用ITEO和序列的数值做卷积运算，得到各个点的边缘幅度。每个点的边缘幅度代表这个点是要寻找的分段点的程度。边缘幅度的值越大说明这个点是边缘点的可能性越大。计算边缘幅度值是将ITEO与原始时间序列的数值点进行卷积运算。算法伪代码如表3-1所示。

表3-1 图像边缘算子与时间序列做卷积

|  |
| --- |
| 输入：时间序列curve, 卷积窗口大小win\_size  输出：卷积后的时间序列faltung\_curve |
| * 1. for (x（时间）,y（值）) in curve: // 选择序列中每一个点   2. for (x\_dist=1; x\_dist<win\_size（窗口）+1;x\_dist++) // 选择窗口内的点   3. x1, y1 = curve[x+x\_dist] // 窗口中的点一   4. x2, y2 = curve[x-x\_dist] // 点二，两个点具有相同权重   5. weight = math.cos(math.pi / 2 - x\_dist) // 计算正弦权重   6. faltung\_curve[x] += (y1 \* (y1-y) + y2 \* (y2-y)) \* weight // 叠加窗口内点权重到序列点上   7. return faltung\_curve |

表3-2 根据ITEO计算的边缘幅度值选取分段点算法

|  |
| --- |
| 算法：根据ITEO计算的边缘幅度选取分段点算法  输入：边缘幅度序列 faltung\_curve  输出：选取的分段点连接成线段结果 |
| * 1. line =[curve[0]] // 时间序列第一个点作为分段点   2. flag = x\_max = x\_min = 0   3. for (x（时间）, y（值）) in faltung\_curve: // 选择序列中每一个点   4. if y > faltung\_curve[x\_max]:   5. x\_max = x // 记录极大值下标   6. elif y < faltung\_curve[x\_max] and x-x\_max > win\_size and flag !=1:   7. line.append(curve[x]) // 超出卷积窗口，添加为分段点   8. x\_max = x\_min = x; flag =1 // flag保证极大值、极小值交替作为分段点   9. if y < faltung\_curve[x\_min]:   10. x\_min = x // 记录极小值下标   11. elif y > faltung\_curve[x\_min] and x-x\_min > win\_size and flag !=-1:   12. line.append(curve[x]) // 超出卷积窗口，添加为分段点   13. x\_max = x\_min = x; flag=-1   14. line.append(curve[-1]) // 时间序列最后的点作为分段点   15. return line |

如何确定时间序列的分段点是一个很关键的问题。一种简单的方法是选择前K个大的边缘幅度点作为分割点。虽然这种方法很简单，但是效果并不好。存在的问题也十分明显。如果直接选择前K大的边缘点，可能会使得分段点距离很近。时间序列的数据实时发生变化，不同阶段的边缘幅度值直接相比也是不合理的。

因此本课题在选取分段点时，选取计算结果的局部极大和极小点作为分段点。首先在边缘幅度值中选取极大极小值点。在所选取的点中根据规定的最小模式的长度阈值d，选取最终的分段点。假设序列点分别是时间序列的边缘幅度的极值点。设是极大（小）边缘幅度值点，只有当是极小（大）边缘幅度点，并且的情况下，是分段点。其中d是正整数，由用户设定的最小模式长度。根据以上算法流程可以看到极大极小边缘幅度点包含分段点。根据ITEO计算的边缘幅度值选取分段点的算法伪代码如表3-2。

## 3.3 聚类并符号化生成模式序列

在对时间序列数据进行分段线性化处理之后，原始的数值点转化为线段的形式。因为线段的个数很多，直接进行符号化会导致问题复杂度变高，需要大量的资源，而且用户更关注的是某一类（或是某几类）线段的关联关系。下一步是对这些线段进行聚类符号化。原始的K-Means聚类方法的算法的流程：

输入聚类个数K,数据集

1.首先随机选择K个聚类中心点，用表示，假设选择的K个中心点是，;

2.比较数据集S中的点距离的距离，选择最小值的归入类别。

3.同一个的数值，用均值代表新的类中心。

4.迭代步骤2,3，直到聚类中心变换的范围小于给定的阈值。

这种聚类算法简单直观，但缺点也很明显，初始的聚类个数K事先规定，K值很难确定。并且选择K个的时候是随机选择的，这就决定聚类的结果依赖于初始的选择。根据聚类的定义，类间的间隔越大意味着聚类效果越好。因此在之前的算法流程中的第一步选择初始聚类中心的时候第一个聚类中心点随机选择，下一个聚类中心点的选择策略，采用每次选择距离之前选取的点的均值最远的点作为下一个初始聚类中心点，直到选够K个。这样确保了初始的选择尽量远。使得类间距尽可能的大，聚类效果会更好。在聚类时需要度量线段之间的相似度。

由于时间序列数据的数据量大实时产生等特点，直接进行关联规则的挖掘不仅会浪费过量的存储空间，而且还会影响后续关联规则的挖掘算法的准确性和可靠性。因此将原始的离散的时间序列表示成模式序列的形式大大降低算法的复杂度，并且更容易解释和评价关联规则。不同的研究对于模式有不同的定义，本文献中的模式指的是在对时间序列进行压缩线性化之后，聚类符号化表示的一类线段。

上文已经描述了具体的聚类方法，而符号化聚类结果需要明确我们用来表示线段的特征的参数。之前的研究中通常采用两个参数来表示一条线段，斜率和长度。这种方式虽然能够表示线段的上升下降的趋势，但是在实际应用中存在一定的问题。比如两条线段A，B按照上述的表示方法具有相同的参数，但是Y轴的最低点不同，A最低点的值是10，而B最低点的值是100，则两条线段在实际数据中表示的意义有很大的不同。因此我们在原有的两个参数基础上加入一个参数表示y轴上面的增长率，数学意义就是y轴的增长值占y轴最低点的比值。假设一条线段是L,则L的表示形式如公式。

|  |  |  |
| --- | --- | --- |
|  |  | (3-12) |

公式中的指的是斜率，指的是线段在横轴的截距（在时间t轴的长度模式持续的时间），是y轴的增长值占y轴最低点的比值。在进行聚类时需要通过计算线段的距离比较相似度，决定样本归于那个聚类中心。计算相似度采用欧式距离[44]。

|  |  |  |
| --- | --- | --- |
|  |  | (3-13) |

公式（3-13）是利用三个参数计算相似度。考虑到三个参数的单位都是不同的，本文献加入归一化的过程。如公式（3-14）所示。

|  |  |  |
| --- | --- | --- |
|  |  | (3-14) |

在单条线段化后的时间序列上分别聚类，对线段聚类之后，为每一个类分配一个符号，就是时间序列的符号化过程。符号化时间序列是因为原始的时间序列数据海量并且实时产生，直接挖掘不仅会降低挖掘效率浪费大量的存储空间，并且可能会影响算法的准确度和可靠性。另一方面，研究过程中用户可能并不需要获得太多的时间序列的细节，更多的关注某一时间段数据的增减趋势或者空间形状。比如在研究股票数据时，我们更多的关注股票的涨幅和跌幅。在模式的符号前加上时间序列的序列号作为标记。图3-5中包含的模式为S1\_a, S1\_c, S1\_b, S2\_a, S2\_e, S3\_c, S3\_d, S3\_f。

T

S1

S2

S3

a

c

b

c

d

f

a

e

Value

图3-5 多元时间序列模式序列

到此为止，时间序列预处理的过程已经完成。整个时间序列的预处理阶段的流程如图3-6所示。

采用ITEO算法对数据进行线性化

将相似线段聚类

符号化聚类的线段

结束

生成模式序列

读入一个窗口数据

开始

图3-6 时间序列预处理流程图

## 3.5 本章小结

本章主要讲述时间序列的预处理过程。由于时间序列数据是连续产生的，并且具有实时性，直接在时间序列上进行挖掘关联规则并不现实，不仅会耗费过多的资源而且对于后续关联规则的挖掘增加复杂度。因此在进行挖掘关联规则之前需要对时间序列进行预处理。预处理的过程主要包括时间序列的压缩，聚类符号化，形成模式序列。时间序列的压缩方法有很多，本课题通过与现有的时间序列压缩方法进行对比之后，提出改进的边缘算子算法ITEO算法，选择时间序列中的分段点，连接分割点，线性化。然后对于已经线性化的时间序列，进行聚类，采用了IK-means方法进行聚类并符号化，形成模式序列。这样就可以在模式序列上进行挖掘关联规则了。

第4章 基于IAMTL的多元时间序列跨事务关联规则挖掘

经过第3章的预处理，时间序列已经转化为模式序列的形式。本章主要介绍在模式序列上的关联规则的挖掘。本课题针对多元时间序列数据的跨事务的关联规则，数据量大，处理流程也比较复杂，传统的算法多针对多元时间序列事务内关联规则挖掘，对于跨事务关联规则挖掘大多忽略关联规则前后件的顺序性，以及数据增量挖掘的需求。本文设计了新型的双头结点的链表表示方法，表示固定的时间T内的多时间序列的跨事务关联规则。在内存中维护一个全局的基于多时间序列的跨事务关联规则链表IAMTL(Inter-Transaction Association Rules from Multiple Time-Series List)，通过对IAMTL的更新和剪枝来挖掘关联规则，提高算法的效率。I-IAMTL算法是基于IAMTL的增量实现算法，不需要保存历史数据，利用修正支持度计数，实现增量的挖掘关联规则。

## 4.1 多时间序列跨事务关联规则的相关概念

关于时间序列中关联规则的挖掘最先研究的是单时间序列。假设一条时间序列为S，它经过预处理后的模式序列形式是S={A,B,C,D,E….}，其中A,B,C,D,E指的是聚类之后的符号，表示不同的模式。对于单时间序列挖掘出的关联规则形式是, 表示的意义是如果模式发生，在某个时间T内或者时间T之后模式也会发生。如图4-1所示。

Value

图4-1 单时间序列关联规则挖掘

对于与上例相同的关联规则AB，多元时间序列事务内关联规则的挖掘是指A，B属于用一个事务，A，B可能来自不同的时间序列。而跨事务的关联规则挖掘则表示A，B可能来自不同的事务，属于不同的时间序列。这里A，B表示的意义与单时间序列相同，不同问题中定义的事务是不同的。本课题的事务指一个时间T限制内的模式。多元时间序列事务内关联规则的挖掘如图4-2所示。

图4-2多元时间序列事务内关联规则的挖掘

多元时间序列跨事务关联规则的挖掘示意图如图4-3所示。

图4-3 多元时间序列跨事务关联规则的挖掘

通过分析，多时间序列事务内关联规则的挖掘，只能挖掘出同一时间段内多个时间序列之间的关联关系。对于时间序列挖掘的研究，更广泛的应用于预测，而事务内关联规则很难达到预测的目的。跨事务挖掘的则是多条时间序列在不同的时间段内的关联关系。关联规则的前件和后件的发生有时间先后顺序。通过挖掘到的关联规则，可以推测在前件发生的前提下有多大的概率后件会发生，从而达到预测的目的。

对于多时间序列跨事务关联规则的挖掘，假设有n条时间序列数据，而我们规定的跨事务的跨度是x，即可以跨x个事务，则在x跨度内可以产生的关联规则的数目是nx，因此我们发现随着时间序列条数和事务跨度的增加，挖掘到的关联规则的数目呈指数增加。增加了挖掘跨事务关联规则的难度。其中事务的跨度的设置也是至关重要的。通常我们更多的关注一定时间段内的关联规则，因为关联规则具有时效性。

## 4.2 基于时间序列的关联规则的挖掘算法

由于时间序列数据海量性，实时性的特点，传统的关联规则挖掘算法不能够对时间序列关联规则进行有效的挖掘，不能反映时间序列的先后顺序。因此大多数的算法在以往的基础上进行了改进，采用滑动窗口的方式，限定当前要挖掘的事务集，将连续的时间序列离散为事务集。通过对预处理后的事务进行合并得到事务间数据集，在改进的存储结构基础上进行挖掘频繁项集。

这类基于FP-growth算法改进的算法虽然在一些问题上可以挖掘到部分关联规则，但是存在的问题也非常明显。在对原始的时间序列预处理结束之后，为了挖掘事务间的关联规则，需要将事务内数据集合并，这会导致部分重复统计的事务项。合并的跨度的选择也是十分困难的，对于后面关联规则的挖掘影响较大。时间序列关联规则与普通的关联规则关键的区别时间跨度T也并没有体现，前后件的先后顺序的严格性在之前的算法中并不重要。因此本课题提出了更适用的算法。

## 4.3 时间序列频繁模式的定义

在经典的购物篮问题和离散性问题中，模式的频繁性是根据模式出现的次数确定的。在挖掘频繁模式之前会提前设定一个固定的阈值，如果出现次数小于阈值则应该剔除，大于则是频繁模式。然而在时间序列中，以模式出现的次数决定频繁程度是不现实的，因为在上一章中描述过，在预处理时间序列时，采用线性化的方式压缩时间序列，时间序列中的模式是以时间轴的长度，线段的斜率，线段在y轴的增长幅度占y轴最低点的比率来决定的，因此在时间序列的模式表示中如果出现两个相同的模式的可能性会很小，因此我们通过比较模式的相似度，进行了聚类，根据在预处理过程中对时间序列的聚类，用特定的字符表示某个模式。某个模式的频繁程度由在一条时间序列中与特定的模式相似的模式的频繁程度决定。

一个时间序列模式可以生成一些时间序列关联规则，序列模式，序列模式S结束的时间是,结束的时间是,并且满足 (是限定的时间窗口)，如图4-4所示。所以在时间段T内有三个事件EF，E，F。因此由频繁模式序列A能够推出的时间序列模式有三个，对应产生的关联规则是,, 。然而需要验证这三个关联规则是否满足和，如果满足则是用户感兴趣的关联规则。目标是挖掘出保证A在B之前出现的规则，这样的规则才是有意义的。

一个频繁模式A能够推出的关联规则的数量的决定因素有，规定的时间长短T的限制，，。

Value

图4-4 序列模式产生关联规则

在时间序列的序列模式中生成关联规则的步骤如下：

首先需要用户提前规定要挖掘的关联规则的时间长度，，。其中表示最小支持度阈值。表示最小置信度阈值。

（1）确定由频繁模式A决定的时间序列AB的最小支持度，统计子时间序列S中模式A出现的次数，用P(A)表示，N表示序列中模式的总个数，计算公式如（4-1）。

|  |  |  |
| --- | --- | --- |
|  |  | (4-1) |

（2）确定由频繁模式A决定的关联规则的后件B。在时间序列S中搜索A模式结束后时间T内发生的模式集合,S中每个模式都是以A模式作为前件的关联规则的后件。二元的关联规则有k个，如公式（4-2）。

|  |  |  |
| --- | --- | --- |
|  |  | (4-2) |

然后根据和确定关联规则是否是满足条件关联规则。

（3）确定置信度, 搜索在模式后T时间段内发生的次数，用表示，公式如（4-3）所示。

|  |  |  |
| --- | --- | --- |
|  |  | (4-3) |

（4）确定是否是强时间序列关联规则，如果关联规则规则同时满足公式（4-4），（4-5），则关联规则就是满足要求的关联规则。

|  |  |  |
| --- | --- | --- |
|  |  | (4-4) |
|  |  | (4-5) |

## 4.4 多元时间序列跨事务关联规则的挖掘

### 4.4.1 基于IAMTL的时间序列跨事务关联规则挖掘算法

定义了频繁模式之后，下一步是挖掘多元时间序列中的关联规则。基于时间序列数据的特点，数据量大，随机波动等特点，本课题以参数T作为时间限定，设计了一个基于多时间序列的跨事务关联规则链表IAMTL（Inter-Transaction Association Rules from Multiple Time-Series List），将预处理后的多元时间序列模式映射到这个概要存储结构上，然后挖掘相邻的两个时间T上的关联规则。假设有3条时间序列，分别是S1,S2,S3,经过第三章的预处理过程之后三条时间序列的模式序列表示分别是：

S1={1a,1b,1c,1b,1a,1b,1a};

S2={2a,2b,2c,2b,2a,2b,2a};

S3={3a,3b,3c,3b,3c,3b,3c}。

三条时间序列的序列模式表示形式如图4-5所示。

图4-5 时间T限制的多元时间序列模式表示

如图所示，三条时间序列分别进行预处理，线性化和聚类之后符号化表示，前面的数字表示时间序列标号，后面字母代表类别。在第三章节中我们提出了用三个特征表示线段（x轴的截距，线段与x轴的夹角，y轴的截距占最低点y的比例）。

然后根据多元时间序列的模式表示形式，建立多时间序列的跨事务关联规则链表IAMTL。将数据映射到链表中，算法的过程如表4-1所示。

表4-1 建立关联规则链表IAMTL

|  |
| --- |
| 算法：建立关联规则链表IAMTL  输入：序列化后的时间序列ts，时间段长度time\_span  输出：IAMTL链表 |
| * 1. time\_span\_num = 0   2. do   3. for (i = 0; i <len(ts); i++) // 取每个时间序列的第一个元素是当前元素   4. if (empty(ele\_arr[i]) == True) // 该时间序列在当前元素数组中没有元素   5. ele\_arr[i] = ts[i].deque() // 出队操作   6. ele = min(ele\_arr) // 取这些当前元素数组的中的最小元素   7. If (matrix.in\_head\_list(ele) == False) // 判断是否在元素头链表中   8. matrix.add\_to\_item\_list(ele) // 加入对应元素链表的Next域   9. else   10. matrix.add\_to\_head\_list(ele) // 加入元素头链表 |

表4-1（续表）

|  |
| --- |
| 算法：建立关联规则链表IAMTL  输入：序列化后的时间序列ts，时间段长度time\_span  输出：IAMTL链表 |
| * 1. if ele.time < time\_span \* time\_span\_num // 判断元素是否属于当前时间段链表   2. matrix.add\_to\_time\_link(time\_span\_num, ele)// 加入当前时间段链表   3. else // 当前最小元素都大于改时间段   4. time\_span\_num += 1 // 改变当前时间段   5. matrix.add\_to\_time\_link(time\_span\_num, ele) // 加入当前时间段链表   6. del(ele\_arr, ele) //   7. while (empty(ts) == False) // 判断是否完成所有元素   8. return line |

建立多时间序列跨事务关联规则链表的过程，扫描多时间序列的序列模式表示，每次取多条时间序列的第一个元素，也就是发生时间最早的模式。每次在将多元时间序列映射到链表中时，选取规则是按照时间发生的先后顺序，这个原则保证了我们在生成关联规则时，前件发生的时间一定在后件之前。判断这个模式在链表的头结点中是否存在，如果存在，只需要将它加入到对应的节点域的next域中，如果不存在，则在头结点中新建该节点，next域置为空。同时需要判断当前的模式所在的时间段是否是当前的时间段链表，如果是，则加入当前时间段链表，如果不是则在时间段加一后加入下一个时间T窗口内。图4-5中的多元时间序列模式形式，按照表4-1的算法建立双头节点的链表，得到4-6所示的结构。在完成链表的建立之后，左边的头结点统计了模式出现的次数，根据用户提前规定的模式的最小支持度阈值可以进行剪枝。因为一阶项集，若模式出现的次数如果小于最小支持度阈值，则二阶频繁项集中不可能出现这些模式的组合，因此可以直接不需要扫描。本课题通过标记那些不满足最小支持度阈值的模式，在第二步统计二阶频繁项集时就不需要再扫描被标记的模式。即在图中头结点里用斜线标记的模式。

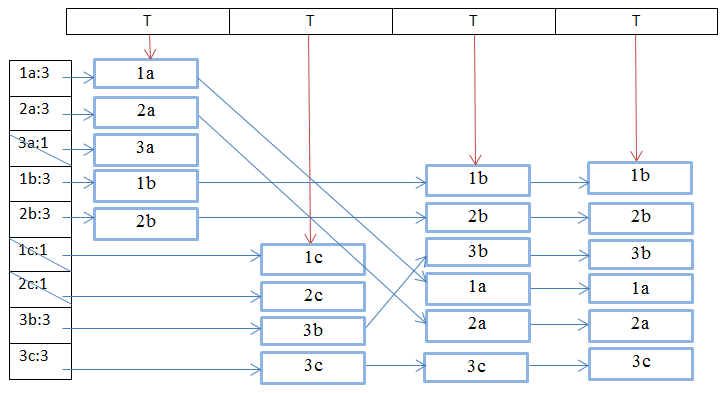


图4-6 基于多时间序列的跨事务关联规则链表（IAMTL）

在建立IAMTL链表时，头结点有一个统计模式个数的过程。在全部扫描完时间序列数据时，头结点链表中保存了每个模式出现的次数。根据模式的支持度公式，分母是模式的总个数，只需要比较每个模式的个数就可以得到模式是否满足最小支持度阈值。这实际是一个剪枝的过程，小于的模式则不参与下一步二阶项集的统计。在算法中本课题采取标记策略，标记那些不满足条件的模式。如果在一条时间序列中在时间T时间段内出现两次同一个模式则从头节点的指针沿着相同的节点指向它。在建立IAMTL链表的同时需要统计模式的数目，如果模式数目大于规定的支持度阈值，则属于下一步需要建立二阶项集的模式，如果小于则在下一步扫描中可以忽略。建立这个概要存储结构的过程，同时也是为多元时间序列划分事务集的过程。同时可以减去不满足最小支持度一阶项集，可以让下一步搜索步骤可以少搜素一些项集，更方便剪枝。

根据建立多元时间序列跨事务关联规则链表IAMTL，下一步是统计多元时间序列中出现的频繁项集，统计二阶项集的支持度，建立二阶项集的支持度矩阵。算法建立好概要存储结构之后，扫描链表，在链表横向的两个T时间限制内前一个中的模式发生时间一定在后面的模式之后，而竖直方向同样按照时间顺序排列，因此二阶频繁项集的建立不需要再次比较模式的发生时间。通过扫描二阶项集的支持度计数矩阵，检测关联规则是否满足最小的和的阈值。假设矩阵的元素是，并且则生成关联规则。算法的伪代码如表4-2所示。

表4-2 根据IAMTL链表查找频繁项集

|  |
| --- |
| 算法：查找频繁项集  输入：IAMTL链表  输出：频繁项集matrix |
| |  | | --- | | 1. //获取二阶频繁项集    1. for item in matrix.item\_list    2. //查找同时间跨度的项    3. while (item.next != NULL)    4. if item.next.time > item.time    5. freq2.add\_one(item, item.next)    6. //查找下一时间跨度的项    7. next\_span = matrix.get\_time\_span(item.time).next    8. next\_item = next\_span.next    9. while (next\_item.next != NULL)    10. if next\_item.time > item.time    11. freq2.add\_one(item, next\_item)    12. //统计一阶频繁项    13. freq1.add\_one(item) | |

按照算法流程，如图4-6所示的IAMTL数据集建立二阶项集支持度矩阵时，按照头结点链表中的顺序，扫描到第一个模式1a时，沿着竖直方向的链表找到（1a,2a），（1a,3a），（1a,1b），（1a,2b）括号中第一项是关联规则的前件，第二项是后件。1a所在的T窗口的下一个窗口中的模式都发生在1a模式的后面，并在T范围内，因此得到以下关联规则：（1a,1c），（1a,2a），（1a,3b），（1a,3c）。下一步沿着1a模式的横向指针找到下一个1a模式在的T窗口，找到所有以1a作为前件的关联规则。而在头结点表中被标记的3a,1c,2c模式的支持度计数是1，假设模式的是2，则三个模式不满足阈值，不需要进行下一步二阶项集的建立。重复迭代。依照图4-9的IAMTL结构，得到的二阶项集矩阵如表4-3.

表4-3 二阶项集矩阵

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1a | 1b | 1c | 2a | 2b | 2c | 3a | 3b | 3c |
| 1a | 0 | 2 | 1 | 4 | 2 | 1 | 1 | 2 | 4 |
| 1b | 3 | 0 | 1 | 3 | 4 | 1 | 0 | 4 | 4 |
| 1c | - | - | - | - | - | - | - | - | - |
| 2a | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 4 |
| 2b | 3 | 1 | 1 | 3 | 0 | 1 | 0 | 4 | 4 |

表4-3（续表）

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1a | 1b | 1c | 2a | 2b | 2c | 3a | 3b | 3c |
| 2c | - | - | - | - | - | - | - | - | - |
| 3a | - | - | - | - | - | - | - | - | - |
| 3b | 3 | 2 | 0 | 3 | 2 | 0 | 0 | 0 | 4 |
| 3c | 2 | 2 | 0 | 2 | 2 | 0 | 0 | 2 | 0 |

假设规定的是4，为4/14。则得到的满足条件的关联规则是（1b,2b），（1b,3b），（1b,3c），（2a,3c），（2b,3b），（2b,3c），（3b,3c）。满足的关联规则是（2a,3c），（3b,3c）。也就是和。矩阵可以用二阶哈希表加快查询速度，这里为了方便表示采用矩阵方式。

### 4.4.2 基于IAMTL的时间序列跨事务关联规则的增量挖掘算法

在实际生产中产生的时间序列流数据是随着时间顺序源源不断的产生的，而过去很长时间内产生的数据不可能一直保存下来，但也不能全部删除掉，历史数据对于关联规则的挖掘有一定的影响。因此在挖掘多元时间序列关联规则时需要设计一种增量的挖掘算法，过去一段时间内的数据与当前数据应该占有不同的权重。实际生产中，用户更关心的是新到来的数据，因此距离当前数据窗口越远的数据占的权重应该越小。在以往的时间衰减窗口[45]中，我们认为近期发生的时间窗口的数据对于当前挖掘的关联规则影响更大，而对于历史数据兴趣度更低一些。计数衰减的方法是针对此问题提出的。每个基本窗口中模式都会有一个计数，假设一个模式A在第i个窗口中的计数是,每个基本窗口有不同的衰减权值大小用表示，权值按照距离当前窗口由近及远递减。则模式A的计数用公式（4-6）计算。

|  |  |  |
| --- | --- | --- |
|  |  | (4-6) |

受到时间衰减窗口的启发，基于本课题用时间跨度T限制的关联规则挖掘，将每个时间T看作一个窗口，当前到达的数据在统计模式的计数时，用权值衰减的计算方法。假设将历史数据分为三个部分，P1是1个月内的数据(某个模式A在1个月内出现的频次)，P2是3个月内的数据，P3是P2前的历史数据。当前的时间T内数据在统计计数时应该与不同衰减权值的数据合并。模式A的支持度技术用公式（4-7）计算。

|  |  |  |
| --- | --- | --- |
|  |  | (4-7) |

采用衰减因子的计数方式之后，原来的多元时间序列跨事务关联规则挖掘算法就可以实现增量挖掘，不需要全部记录历史的数据。基于IAMTL算法的时间序列跨事务关联规则的增量挖掘算法（I-IAMTL）算法的算法思想主要是在新到时间序列数据时，根据动态维护的两个时间T的事务集，将出现的模式和关联规则的修正支持度存入哈希表中，不需要保存整个时间序列历史数据。

算法伪代码如表4-4所示。

表4-4 增量挖掘多时间序列跨事务关联规则

|  |
| --- |
| 算法：增量挖掘多时间序列跨事务关联规则  输入：原有的关联规则库rules，时间段长度time\_span  输出：频繁项集 |
| |  | | --- | | * 1. rules.update() // 更新rules的各个时间段   2. time\_span\_num = 0   3. trans = list() // 存放当前时间段事务集   4. last\_trans = list() // 存放上一个时间段事务集   5. do   6. for (i = 0; i <len(ts); i++) // 第一个元素是当前元素   7. if (empty(ele\_arr[i]) == True) // 该时间序列在当前元素数组中没有元素   8. ele\_arr[i] = ts[i].deque() // 出队操作   9. ele = min(ele\_arr) // 取这些当前元素数组的中的最小元素   10. if (ele.time < time\_span \* time\_span\_num) // 元素是否属于当前时间段   11. trans.append (ele) // 加入当前时间段事务集   12. else // 当前最小元素都大于改时间段   13. for (j = 0; j < len(trans); j++)   14. for (k = j + 1; k < len(trans); k++) // 查找同时间跨度的项   15. key = trans[j].key + trans[k].key   16. rules.add\_recent(key, 1) // 在规则最近时间段加一   17. for (k = 0; k < len(last\_trans); k++) // 查找上一个时间跨度的   18. key = trans[k].key + trans[k].key   19. rules.add\_recent(key, 1) // 在规则最近时间段加一   20. last\_trans = trans   21. time\_span\_num += 1 // 改变当前时间段   22. del(ele\_arr, ele) //   23. while (empty(ts) == False) // 判断是否完成所有元素   24. return line | |

假设如图4-5所示的时间序列数据，在进行关联规则挖掘时，首先扫描经过同样预处理的模式序列，然后在扫描的同时将模式和形成的候选关联规则集用哈希表保存并统计计数，其中计数相当于模式或者候选关联规则的支持度计数，计算方式采用上节介绍的衰减权值与原始计数相乘的修正支持度计数。维持一个如图4-7所示的动态数据结构，两个固定的时间T限制内的事务集，当有新的数据到达时，之前一个T时间限制内的数据将会被删除，同时当前的模式和关联规则的修正支持度会保存在哈希表中，达到增量挖掘关联规则的目的。

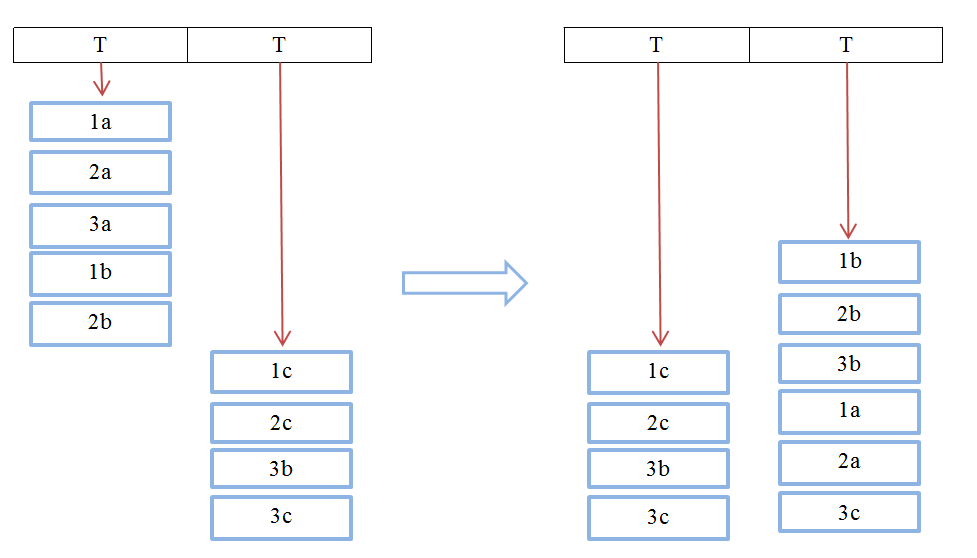


图4-7 I-IAMTL算法动态结构

## 4.5 多元时间序列关联规则预测序列

在时间序列关联规则挖掘中，输入时间序列*X*，，和时间间隔*T*，根据算法可以挖掘到很多关联规则。但是对于其评价标准则研究很少。

时间序列关联规则广泛应用于预测。传统的时间序列预测方法是基于挖掘过程，进行统计分析时间序列的趋势，以预测未来一段时间内出现连续的数值。一般来说，时间序列的数据量是相对较大的，使用上述方法处理起来速度较慢。而关联规则用于时间序列的预测，具有良好的解释性，预测模型以条件概率作为预测条件。

本课题采用关联规则应用于预测序列模式的性质，验证挖掘到的关联规则的价值。先把数据集分为训练集与测试集。训练集数据用于运用本文献提出的关联规则的挖掘算法挖掘关联规则，测试集则用于验证挖掘到的关联规则的优劣以及预测方法的准确程度。本课题受到关联规则对序列事件的预测算法[46]的启发，提出的测试方法是前后件滑动窗口式的方法，使用第一个时间段作为滑动窗口的前件窗口，第一个时间段后的第二个时间段作为滑动窗口的后件窗口。前后件窗口PCW (Prereqiusite and the Consequent Window)的大小与之前挖掘关联规则时设置的时间T长度保持一致。滑动窗口结构如图4-8所示,箭头指向窗口的滑动方向。

后件窗口

下一个窗口

前件窗口

图4-8 PCW前件后件窗口

在利用测试集测试关联规则在时间序列预测方面的有效性时，使用前一个时间段也就是前件窗口中出现的模式预测下一个时间段的模式并计算预测的准确度，然后把整个滑动窗口向后移动一个T时间段，并重复上一步计算预测结果。评价预测后件的优劣，可以采用准确率，召回率等，还可以采用符合标准如F1值，G-Mean等。本课题采用预测准确率来评价挖掘的关联规则。计算公式如公式（4-8）所示。其中TP代表查找正确的后件数量，FP代表查找错误的后件的数量。

|  |  |  |
| --- | --- | --- |
|  |  | (4-8) |

对于时间序列的预测方法还可以使用最长匹配策略，由于本课题使用的是前件长度相同的规则，因此无法使用这种方法进行预测，基于马尔科夫链的预测方法，通常用于单一模式的预测，在预测多个模式的场景中表现效果并不好。运用关联规则进行时间序列关联规则预测问题是指使用之前出现的模式来推测之后时间序列可能出现的模式。例如的关联规则中前件A和后件B，在前件窗口中搜索到模式A之后，如果在后件窗口中找到了模式B，则证明此条关联规则预测的准确度在这个滑动窗口中是1。而本课题通过选取部分关联规则然后验证，最后求预测准确度的平均值的方法来度量算法的准确度。

本课题采用的是基于置信度的关联规则的筛选方法，因为置信度高的关联规则在时间序列的下一个时间段内出现的频率更高，并且在关联规则挖掘之前已经选择了支持度大于最小支持度阈值的模式，避免了选择非频繁模式。预测算法的第一步先在前件窗口中使用每一个出现的模式去关联规则库中去查找与模式相同的规则，这些关联规则是在预测中可能会使用到的关联规则。第二步根据置信度排序从而选出K个置信度最高的关联规则。如果采用增量算法进行预测，要使用该算法记录各个时间段的支持度与置信度，用上一章中介绍的衰减算法进行修正最小支持度和最小置信度，首先利用最小支持度进行过滤，然后选择K个置信度最高的关联规则。这些关联规则的后件作为预测结果。K是根据不同的训练数据做出的不同选择，训练数据中每个时间段中出现的模式的数量不同，可以根据预测准确率进行交叉验证，选择一个最好的K来预测。通过按照在选出的K个置信度的关联规则中，先在IAMTL中搜索前件，然后在后件窗口中搜索后件。通过关联规则预测下一时间段的伪代码如表4-5所示。

表4-5 预测策略伪代码

|  |
| --- |
| 输入：关联规则库rules, 每个后件选择数量b\_k，前件模式序列a\_list, 最小支持度min\_support，衰减因子u  输出：预测模式 |
| 1. b\_list ={} // 返回的后件，字典类型 2. for a in a\_list: // 循环每个前件 3. for b in rule[a]: 4. for i in len(rules[x].support) //循环各个时间段的支持度 5. support += rules[x].support \* u[i] // 计算修正支持度 6. for i in len(rules[x].confident) //循环各个时间段的置信度 7. confident += rules[x]. confident \* u[i] // 8. if support > min\_support: 9. b\_list.add(b, confident) // 叠加每一种类的后件 10. return b\_list[0…k] |

## 4.6 本章小结

本章描述新型的双头结点的链表（IAMTL）表示方法，表示固定的时间T内的多时间序列的跨事务关联规则。在内存中维护一个基于多时间序列的跨事务关联规则链表IAMTL，利用IAMTL结构挖掘多元时间序列跨事务关联规则。引入衰减策略计算支持度计数，提出基于IAMTL算法的增量挖掘关联股则算法（I-IAMTL）。然后介绍将挖掘到的时间序列应用在序列模式的预测方面的算法。采用最小支持度阈值挖掘关联规则，将得到的关联规则按照排序，选取部分在测试集中进行验证并计算预测准确度。

# 第5章 实验结果与分析

本章首先介绍了实验的实验数据，然后分成3个部分介绍课题的实验过程。第一部分对时间序列数据进行预处理，采用ITEO算法选取时间序列中的分段点，实现分段线性化。通过与已有算法对比，验证ITEO算法的效果。第二部分是在预处理之后的模式序列上进行关联规则的挖掘。通过比较tsapriori算法和本文提出的IAMTL算法以及其算法的增量实现I-IAMTL算法在工业数据的时间效率，内存占用，预测准确率三个方面进行比较。同时又比较了两种算法在股票数据上的效果，证明了算法在不同数据集场景上的适用程度。

## 5.1 实验数据的描述

本课题第一部分实验采用的数据集是工业数据集，是广东汕尾热电厂从2013年7月至2013年10月3个月间采集的工业数据，采集数据的时间间隔是5分钟，一共有17570条数据记录。本实验选取了5条典型的时间序列数据作为数据训练集。分别是主蒸汽温度（数显表），主汽门前压力，磨煤机进口一次风压，主蒸汽压力，再热汽温度。

## 5.2 时间序列数据的预处理

第一部分是对时间序列数据的预处理，采用ITEO时间序列边缘算子检测时间序列的关键点进行线性化，运用IK-Means聚类算法聚类线性化后的线段模式，在聚类过程中度量模式相似度时采用线段在时间轴的长度即持续时间t，线段的斜率k，以及线段在y方向的增长占y轴最低点的比率。基于三个参量单位不同采用了归一化处理。

### 5.2.1 采用ITEO算法对时间序列进行压缩线性化

时间序列的ITEO表示是一种分段线性化表示方法，根据设计的时间序列边缘算子与原始的时间序列数据进行卷积的计算结果，得到每个时间序列数据点的边缘幅度，然后根据一定的选择原则检测时间序列的分段点，连接分段点来表示时间序列。采取选择局部范围内的极值点作为选取分段点的策略。

本实验采用了两条不同的时间序列，其中一条是波动幅度不太大的时间序列数据，另一条时间序列则波动幅度明显。本实验选择了另外两种常用的时间序列线性化表示方法与ITEO方法做比较。在压缩率相差不大的情况下，线性化的结果对比图如图5-1至图5-3所示。

Method=PLR\_EP

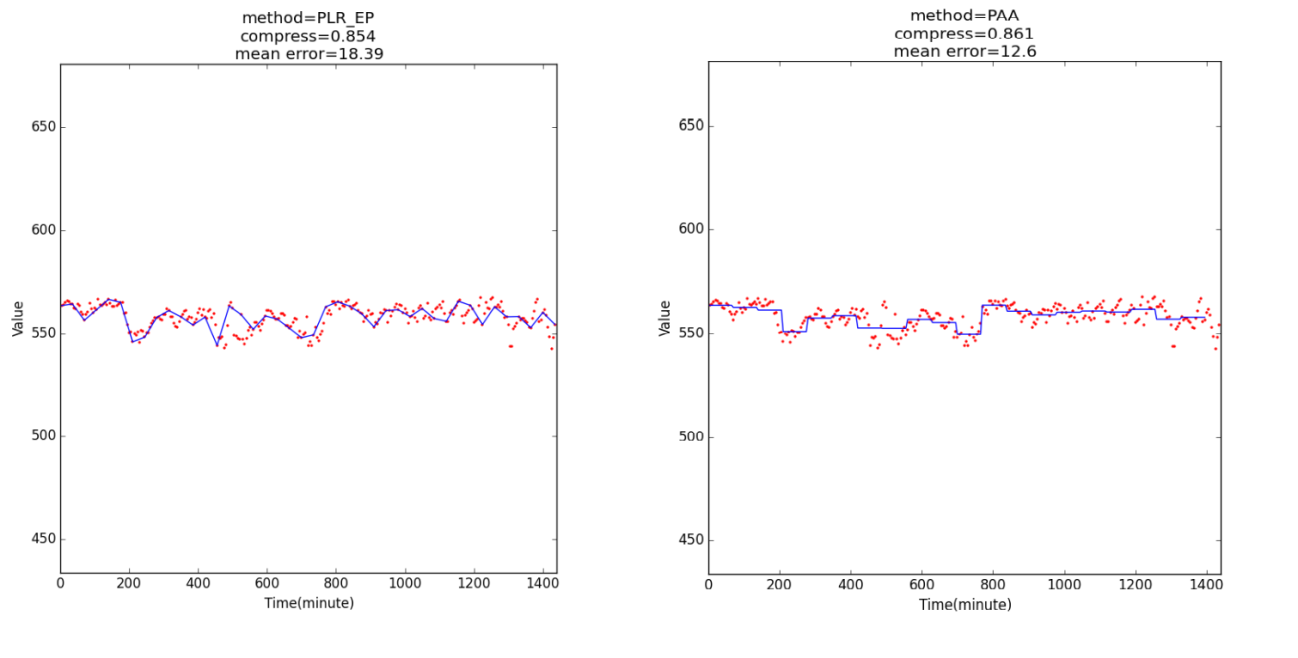
Compress=0.853

Mean error=18.015

Method=PAA

Compress=0.864

Mean error=12.278



Time(min)

Time(min)

Value(℃)

Value(℃)

图5-1 PLR\_EP和PAA算法在再热汽温度上线性化对比

Method=PLR\_EP

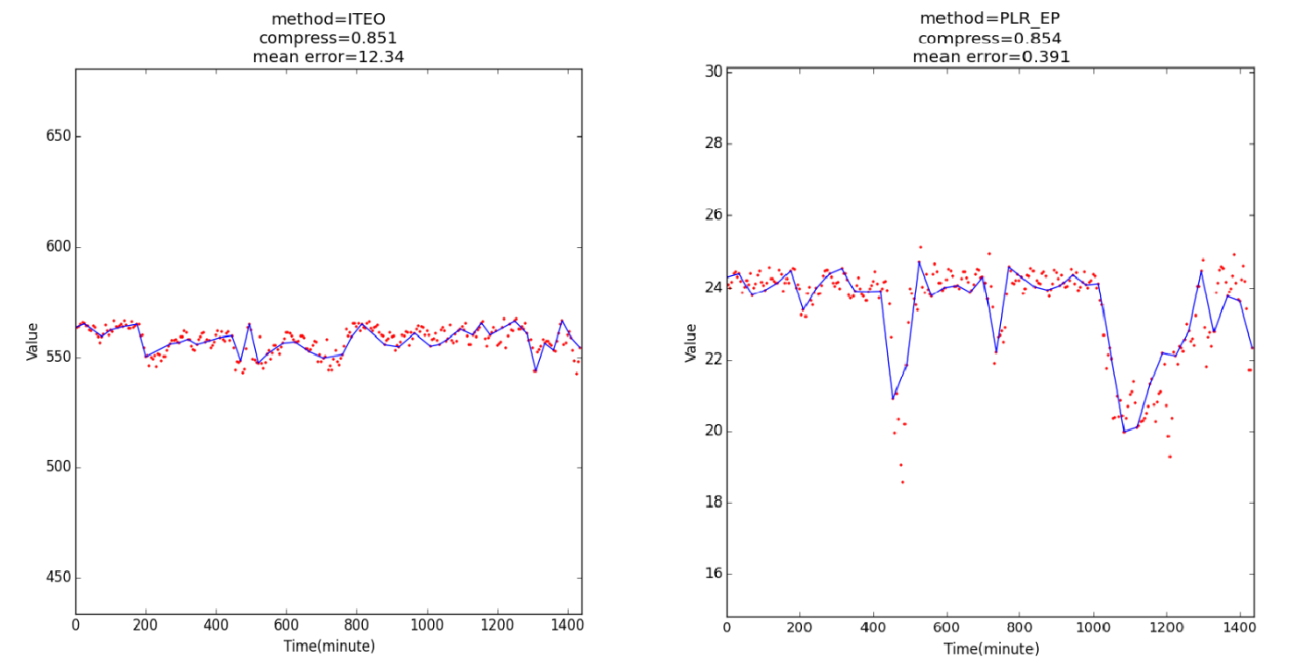
Compress=0.854

Mean error=0.391

Method=ITEO

Compress=0.851

Mean error=12.34



Time(min)

Time(min)

Value(MPa)

Value(℃)

图5-2 ITEO和PLR\_EP算法在两个数据上线性化对比

第一种对比算法是PAA的分段线性化算法，这是最常用的分段线性表示方法，需要用户提前规定阈值R,表示要将时间序列分为多少段。通过平均时间序列，使用每个段的平均值的序列来表示原来的时间序列。其中分段的数目由人为规定，每个时间序列的分段长度都是相等的。因此这种方法的表示效果依赖于分段数参数，显然如果分段数过小，会损失原时间序列过多的信息，时间序列的表示会过于粗糙。相反如果分段数过大，虽然能避免损失原来时间序列的信息，但是降维幅度会变小。

Method=PAA

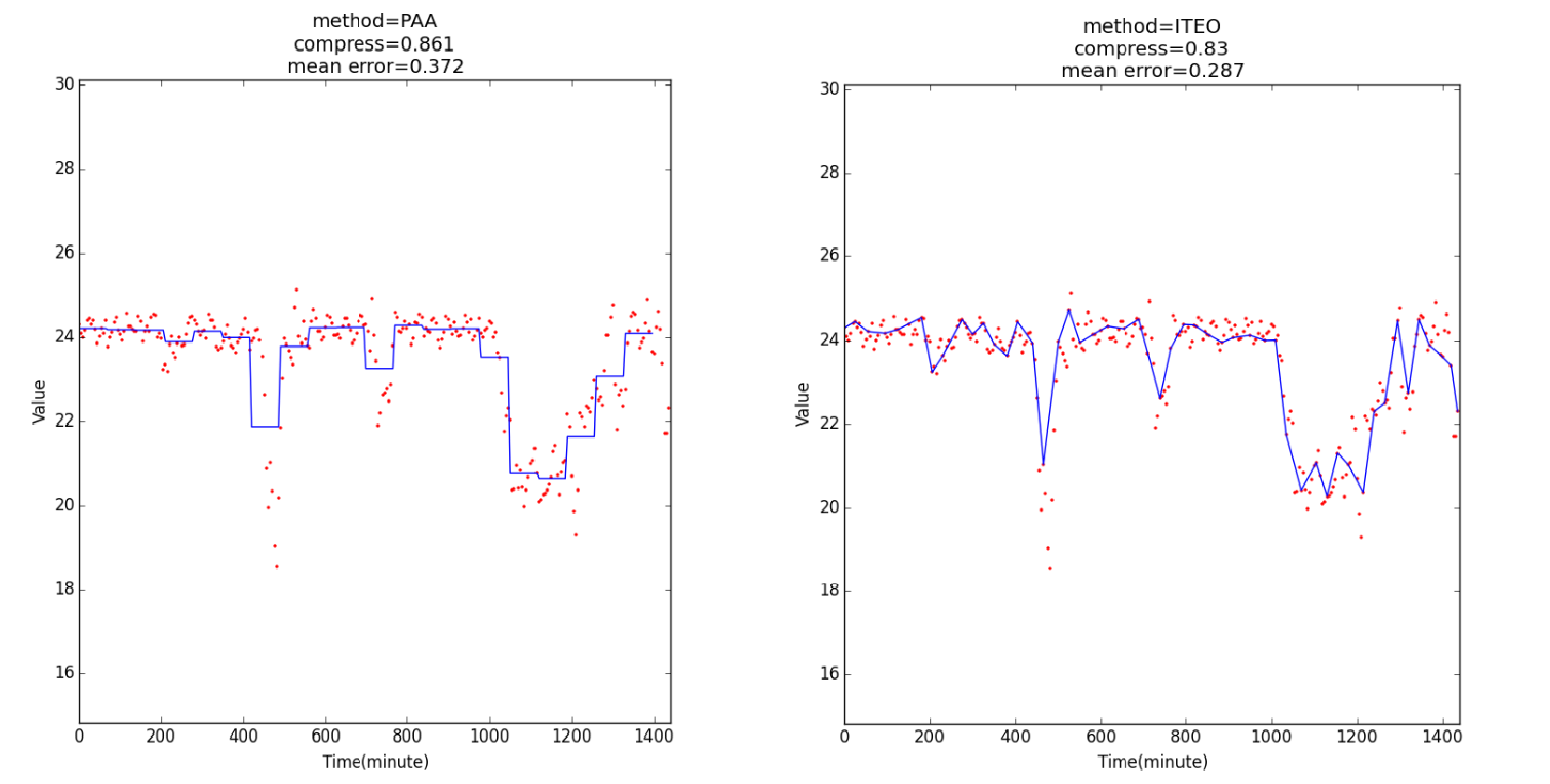
Compress=0.861

Mean error=0.372

Method=ITEO

Compress=0.83

Mean error=0.287



Time(min)

Time(min)

Value(MPa)

Value(MPa)

图5-3 PAA和ITEO在主蒸汽压力数据上线性化对比图

第二种对比算法是PLR\_EP方法，用时间序列的极值点做分段点，达到对时间序列压缩线性化的目的。查找时间序列中的极值点集，这就是分段点。这种算法最明显的优点是不需要像PAA算法一样提前设置要分段的分段数，性能比较好，支持在线划分，并且容易理解。算法基于人类的关注点，但是存在的问题也很明显。该算法是一种局部算法，只考虑两个相邻点，在划分的时候很容易过于精确而忽略了趋势，而极值点的识别方法只考虑了逆问题的趋势。只能从上升到下降和从下降到上升，而对于那些相同趋势但是大的转折点则忽略了，比如一条线段的斜率夹角是10度上升趋势，与之相邻的线段的斜率夹角是80度上升趋势。按照极值点的识别策略，这两条线段的交点就会忽略，而这种转折比较大的点实际有可能是用户比较感兴趣的点。

在计算拟合误差时，假设一条时间序列初始的数据表示形式是，线性化后的表示形式是,其中指的是连接两个分段点，的线段。初始的时间序列在线性化后的线段上映射的点是，则这条时间序列上拟合误差计算公式是如公式（5-1）所示。

|  |  |  |
| --- | --- | --- |
|  |  | (5-1) |

几种时间序列的分段线性化表示算法需要输入的参数是不同的。因此我们可以比较在相同的压缩率大小的情况下，分别采用3种不同的线性化方法压缩好的时间序列与原始的时间序列之间的拟合误差的大小。

计算线性化的压缩率公式如公式（5-2）。

|  |  |  |
| --- | --- | --- |
|  |  | (5-2) |

表5-1是热电厂数据中选取的不同数据集在不同算法下拟合误差对比。

表5-1 不同数据集不同算法的对比结果

|  |  |  |  |
| --- | --- | --- | --- |
|  | 算法 | | |
| PAA | PLR\_EP | ITEO |
| 数据集1 | 12.278 | 18.015 | 11.014 |
| 数据集2 | 0.381 | 0.382 | 0.278 |

采用ITEO算法选取的四条数据集的部分分段点结果如图5-4所示，括号中第一个参数代表时间轴的数值，另一个参数是对应的数据点的数值。连接分段点则完成压缩线性化时间序列。中括号区分不同时间序列数据。

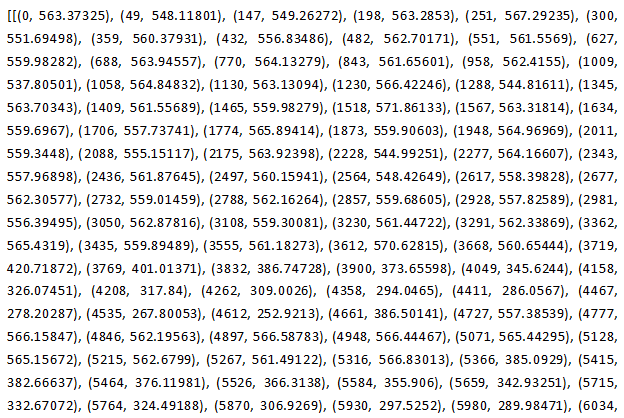


图5-4 采用ITEO算法选取4条时间序列的分段点结果

通过实验对比可以发现，在第一条时间序列中，再热汽温度是变化不大的时间序列，在三种算法的压缩率设定在0.85左右的条件下，PAA算法的平均误差是12.278，极值点线性化方法PLR\_EP的平均误差是18.015，ITEO算法的误差率是11.014。由于第一个时间序列数值比较大，因此平均误差的数值比较大，但与时间序列数值相比是相对较小的。在第二条时间序列主蒸汽压力中，数据的变化幅度相对第一条大的多。在三种算法的压缩率设定在0.85左右的条件下，PAA算法的平均误差是0.381，PLR\_EP算法的平均误差是0.382，ITEO算法的平均误差是0.278。两种不同的时间序列数据，三个算法进行对比，ITEO算法都具有与原始时间序列更小的误差，具有明显的优势。

### 5.2.2 对相似的线段进行聚类并符号化

在本课题中采用ITEO线性化的方式压缩时间序列时，时间序列中的模式是以时间轴的长度，斜率，在y轴的增长幅度占y轴最低点的比率，三个参数来决定的。对线段聚类之前对三个参量进行均一化处理，使用IK-Means聚类。聚类的结果如图5-5所示。本课题采用英文字符代表聚类后的时间序列模式。

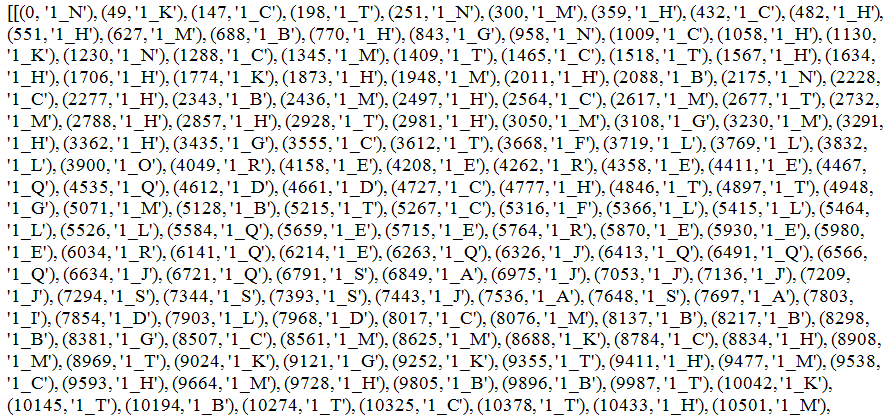


图5-5 聚类符号化后的时间序列表示

## 5.3 算法性能上的比较

经过上述时间序列的预处理之后，得到线段化并符号化聚类之后的事务集。下一步是在这些事务集上进行跨事务关联规则的挖掘。本章从算法运行的时间效率，内存占用，算法应用在时间序列预测方面的准确率这三个方面来比较tsapriori，IAMTL，I-IAMTL三个算法。在以往的研究多元时间序列跨事务关联规则的算法中，通常采用合并事务将事务内数据转化为跨事务的关联规则挖掘，然后将改进的Apriori和FP-tree算法应用到关联规则的挖掘中。这些研究一方面可能由于跨事务关联规则的数据量巨大导致概要存储结构的前缀树变得非常大，另一方面在原本应用在离散数据上的算法往往不能很好的反映时间序列关联规则中的时间属性已经前后件严格的先后顺序，并且在将事务合并大事务集时会出现重复统计模式的问题。因此在综合考虑之后，根据划分事务集的条件不同，挖掘到的关联规则的表示形式及意义不同，本课题选择了其中具有代表性的算法tsapriori算法作为一类对比算法。tsapriori算法是将传统的apriori算法应用在时间序列数据上，针对时间序列的数据特征进行了一定的改进，采用与本课题中所提算法中相同的划分事务集的方式，并同步相同的T。

### 5.3.1 时间效率上的比较

本节主要对tsapriori，IAMTL，I-IAMTL三个算法在时间效率上的比较。在取相同的时间T限制的情况，tsapriori算法运行的时间远大于IAMTL算法，和I-IAMTL算法，其中I-IAMTL算法在相同的时间T相同的最小支持度阈值情况下是最小的。因为tsapriori将每个T时间窗口内的模式看做一个事务，但在挖掘时需要不断扫描事务集。I-IAMTL能够增量的挖掘时间序列上的关联规则，不需要保存之前的数据。假设数据集具有m个时间段（也就是本课题中规定的T限制,下文的事务集也是每个时间T限制内的事务），有n个模式种类。假设有u条时间序列，每条时间序列含有的模式数目是。tsapriori算法首先进行划分事务集，将模式序列组合成事务集，一个T时间限制内的模式是一个事务。时间复杂度是。然后进行模式组合生成候选集，此操作产生C(n,2)次组合，时间复杂度O(n2)。逐个去查看候选集是否为频繁项集，需要时间复杂度就为O(n2\*m)。IAMTL算法利用首先生成一个链表，此步操作时间复杂度与tsapriori算法划分事务集的复杂度相同是。所以生成频繁项集需要O(n2\*m)（O(n\*x\*m)，n是头链表中的模式数目，也就是总的模式种类，x是一个时间T中的模式数，m是时间T限制的间隔数。）。但是这里的n虽然最大值是模式数，但一个事务集中出现的模式一般均远小于n，所以IAMTL时间效率高于tsapriori。I-IAMTL算法操作步骤类似与IAMTL，不用保存过去的数据，通过保存修正的支持度计数，能够增量的挖掘跨事务关联规则。

首先观测在固定的窗口大小，本实验选取24小时作为一个窗口，也就是时间界限T的长度是24小时。三个算法在工业数据集下，分别取最小支持度阈值为0.06，0.08，0.10，0.12，0.14，0.16时，其运行所用的时间对比如图5-6。

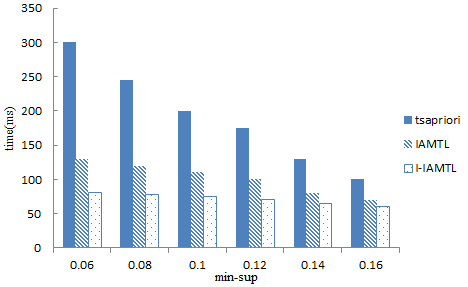


图5-6 不同支持度阈值下三种算法的运行时间对比

通过实验结果可知，随着的增大，三个算法的运行时间在不同程度上都是减小的。其中tsapriori算法和IAMTL算法变化比较明显，I-ITAMTL变化不是很明显。并且I-IAMTL算法的运行时间远小于另外两种算法。这是由于最小支持度阀值变大，tsapriori算法与IAMTL算法由于具有剪枝操作，导致剪枝剩下的模式数目减少，因此所需要的时间都会减小。I-ITAMTL因为需要记录所有出现的事务集，不会进行剪枝操作，因此变化略小。

当选取固定的最小支持度阈值0.1，时间窗口大小分别为12，24，36，48，60，72小时时，tsapriori和IAMTL算法，I-IAMTL算法的运行时间作对比。算法对比如图5-7所示。

从图5-7所示的实验结果对比可看到，在相同的下，随着的增大，单个算法所用的运行时间在增加。这是因为时间限制的窗口增大之后，一个时间限制T内的模式增多，而要挖掘的关联规则的数目也随之增多。而三种算法tsapriori算法运行的时间远大于IAMTL算法，和I-IAMTL算法，其中I-IAMTL算法在相同的时间T相同的情况下是最小的。由于变大，每个窗口中的模式n增多，窗口个数m减小，三种算法需要比较次数都随n增长，但是由于都是时间复杂度O(n2\*m)，所以时间也变长。

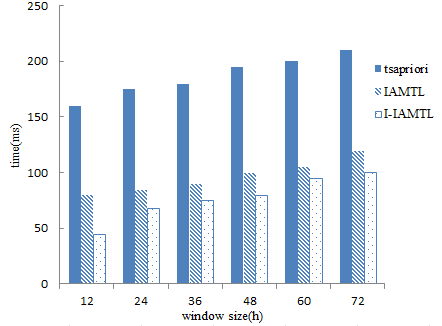


图5-7 不同窗口大小下三种算法的运行时间对比

### 5.3.2 内存占用上的比较

本节主要对tsapriori，IAMTL，I-IAMTL三个算法在内存占用上的比较。tsapriori算法由于每次都需要扫描原始事务集，所以事务集要一直保存在内存中，还需要保存每一步所产生的候选项集以用于生成频繁项集。而IAMTL的事务集采用IAMTL链表保存事务集和频繁项集，生成频繁项集时只需要扫描IAMTL链表即可，从而节省了内存。I-IAMTL由于是增量算法不需要保存IAMTL链表，但需要加载之前保存的规则库。但规则库通常小于IAMTL链表的大小，所以通常所需内存小于IAMTL算法。

首先观测在固定的窗口大小，本实验选取24小时作为一个窗口，也就是时间界限T的长度是24小时。三个算法在工业数据集下，分别取最小支持度阈值为0.06，0.08，0.10，0.12，0.14，0.16时，其运行所占的内存大小对比如图5-7。

从图5-8所示的实验结果对比可看到，在相同的大小下，随着的增大，单个算法所用的内存在减小。这是因为由于变大后，tsapriori与IAMTL由于有剪枝操作，所以需要挖掘的模式减少了，需要的内存也会减小。但I-ITAMTL因为需要记录所有出现的事务集，是不会进行剪枝操作，所以随着支持度阈值的变化算法的占用内存基本不会发生变化。

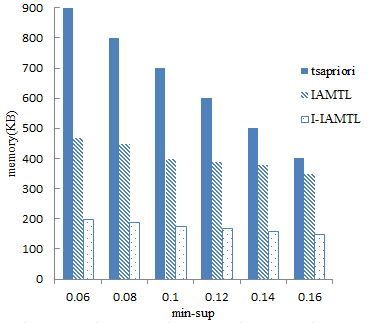


图5-8 不同支持度阈值下三种算法的占用内存对比

当选取固定的为0.1，时间窗口大小分别为12，24，36，48，60，72小时时，tsapriori和IAMTL算法，I-IAMTL算法的占用内存作对比。算法对比如图5-9所示。

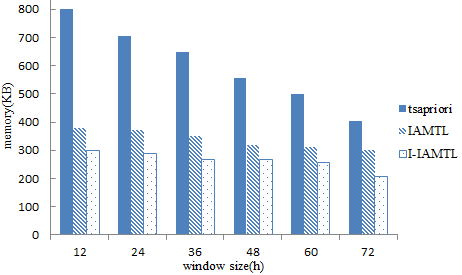


图5-9 不同窗口大小下三种算法的占用内存对比

从图5-9所示的实验结果是因为在相同的下，随着的增大，一个窗口（事务）的模式增多，窗口个数减小，但是窗口小的时候一条线段会被分解成多个模式，因而模式的总数目会增多。所以三种算法内存随着窗口增大都不同程度的减小。

### 5.3.3 预测时间序列准确率上的比较

通过训练集已经挖掘到了一系列多元时间序列跨事务的关联规则，要验证关联规则在时间序列预测方面的准确性，采用部分数据集作为测试集进行测试。本课题通过选取部分关联规则在验证集进行验证，最后求预测准确度的平均值的方法来度量算法的准确度。本课题采用的是基于置信度的关联规则的筛选方法，因为置信度高的关联规则在时间序列的下一个时间段内出现的频率更高，并且在关联规则挖掘之前已经选择了支持度大于的模式，避免了选择非频繁模式。预测算法的第一步先在前件窗口中使用每一个出现的模式去关联规则库中去查找与模式相同的规则。第二步根据置信度排序选出K个置信度最高的关联规则，这些关联规则的后件作为预测结果。由于算法预测的准确率与数据集和算法参数相关，所以要找到合适的参数才能使得算法的预测准确率得到最高，本课题实验验证支持度阈值和时间窗口两个参数和预测准确率的关系。

IAMTL算法挖掘出来的规则是具有顺序的。I-IAMTL具有顺序的同时还考虑了规则支持度、置信度随着时间衰减，所以在预测准确率上I-IAMTL通常更准确一些。由于tsapriori挖掘出来规则是没有顺序的，所以在数据时间相对敏感的数据集上（如股票数据），前两种算法要明显优于tsapriori算法。

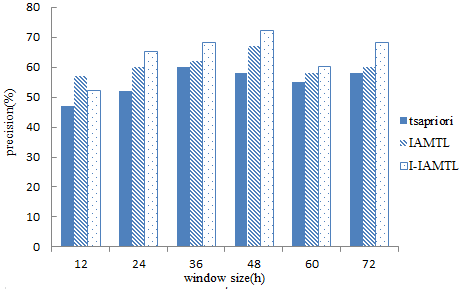


图5-10 不同窗口大小下三种算法预测准确率对比

当选取固定的最小支持度阈值0.1，时间窗口大小分别为12，24，36，48，60，72小时时，tsapriori和IAMTL算法，I-IAMTL算法的预测准确率作对比。算法对比如图5-10所示。

从实验结果可以看到，在相同的最小支持度阈值下，当窗口大小到48小时左右，三个算法所用的准确达到最高。所以在工业数据集上面选取36小时到48小时比较作为挖掘窗口比较合适。

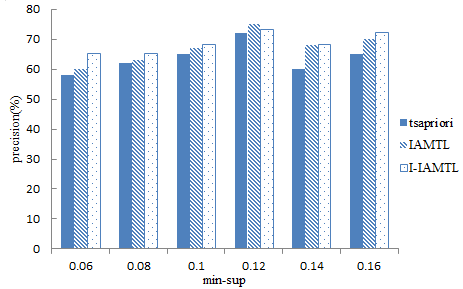


图5-11 不同支持度阈值下三种算法的预测准确率对比

选取24小时作为一个窗口，也就是时间界限T的长度是24小时。三个算法在工业数据集下，分别取最小支持度阈值为0.06，0.08，0.10，0.12，0.14，0.16时，其算法预测准确率对比如图5-11。

从实验结果可以看到，在相同的时间窗口下，当最小支持度阈值达到0.12左右，三个算法所用的准确率最高。所以在工业数据集上面选取0.10到0.12比较作为挖掘的最小支持度阈值比较合适。

通过在不同窗口大小和不同的支持度阈值下对比三种算法对于时间序列数据的预测准确率，可以看出I-IAMTL在多数情况下优于IAMTL，tsapriori算法的预测准确率最低。因此I-IAMTL算法可以有效适用于时间序列的预测方面。随着选参数不同，预测准确率也不同。从图5-10和图5-11可以看到在工业数据集上，当最小支持度取值为0.12左右，窗口大小取48左右时三种算法预测准确率都较高。

### 5.3.4 算法在不同数据集上的性能比较

为了验证算法在不同数据集上的适用性，本课题还比较了两种种算法在股票数据集上的表现，数据源自标普永华网站。包含99年到08年沪深两市的股票数据，数据间隔是5分钟，每支股票数据包括109152条数据。选取SH1A0001（沪市上证指数），SH1A0002（A股指数），SH1A0003（B股指数），SH1B0001（沪市工业指数），SH1B0002（商业指数），SH1B0004（地产指数）这几支股票数据进行试验，在算法的时间效率，内存占用，预测的准确率上进行对比。验证IAMTL和I-IAMTL在股票数据场景下的算法性能对比。

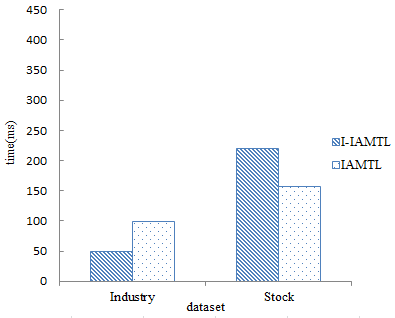


图5-12 算法在工业数据和股票数据上运行时间对比

两个算法IAMTL，I-IAMTL在算法的运行时间上的对比，设置算法的支持度阈值为0.1，窗口的大小时间T限制长度是24小时。算法对比试验结果如图5-12所示。

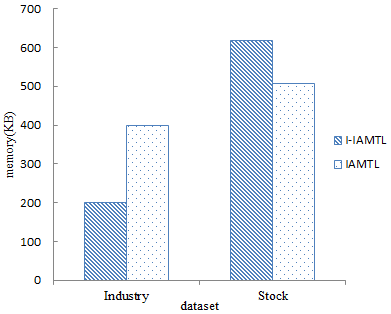


图5-13 算法在工业数据和股票数据上占用内存对比

IAMTL，I-IAMTL在算法的内存占用上的对比，同样设置算法的支持度阈值为0.1，窗口的大小时间T限制长度是24小时。算法对比试验结果如图5-13所示。

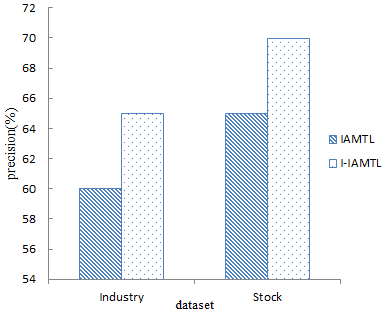


图5-14 算法在工业数据和股票数据上预测准确率对比

IAMTL，I-IAMTL在算法的预测准确率上的对比，设置算法的支持度阈值为0.1，窗口的大小时间T限制长度是24小时。算法对比结果如图5-14所示。

本课题在股票数据上分别对IAMTL和I-IAMTL两个算法在运行时间上，算法的内存占用，算法在预测时间序列的准确率上做了对比。以上实验结果显示在基于股票数据上的运行时间和占用内存方面IAMTL因为通过剪枝将数据量大大缩减因此相比I-IAMTL算法有一定的优势，而在预测准确度方面由于I-IAMTL考虑历史数据对于当前支持度的影响，因此具有更高的准确度。IAMTL算法更适宜于数据海量，但更新比较慢，或者对占用内存要求更高的场景。I-IAMTL算法是IAMTL算法的增量实现，更加适用于时间序列流数据这类实时产生的数据，对于时间序列的预测具有更高的准确度。在数据上进行增量的挖掘关联规则，当新的数据到来时不需要重新考虑之前的历史数据，从而实现增量挖掘。

## 5.4 本章小结

本课题是多元时间序列数据上的跨事务关联规则的挖掘，关联规则的挖掘分为两个步骤，首先是对时间序列的预处理，然后是关联规则的挖掘。

首先是时间序列预处理，采用ITEO的方式对时间序列数据进行线性化分段。然后应用IK-Means算法对线性化的时间序列模式进行聚类符号化，形成事务集。本课题通过将ITEO，PAA，PLR\_EP三种算法在相同的压缩率下的对比证明在相同的压缩率情况下ITEO对时间序列数据的线性化拟合误差更小，效果更好。

第二步是关联规则的挖掘，对本课题提出的IAMTL算法和I-IAMTL算法的有效性进行验证。首先采用热电厂的工业数据对tsapriori和IAMTL和I-IAMTL算法在运行时间，内存占用，预测时间序列的准确率上进行比较，然后为了证明算法在不同数据集场景上的适用性，本课题又选用了股票数据集进行相应方面的对比实验。通过对比实验证明在股票数据场景下IAMTL算法在运行时间，占用内存相对于I-IAMTL算法有一定的优势，而在预测准确度方面I-IAMTL算法更优。IAMTL能够挖掘出固定的时间T限制下的关联规则，划分事务集，反映关联规则的前后件的严格时间顺序，能够通过对数据的剪枝提高算法的效率。I-IAMTL是其增量实现更适合实际情况下的海量时间序列数据实时产生的增量的挖掘关联规则。总体而言，算法达到了预期的目的。

# 结 论

随着经济的发展，科技的进步，在各个领域都会产生大量的时间序列数据。获得隐藏在这些数据中的有用的信息对人们的生产和生活做出一定的决策指导作用是非常有意义的。因此本课题通过分析以往的关联规则挖掘算法的缺陷和不足，结合时间序列数据的海量性实时性等特点，提出挖掘多元时间序列流跨事务关联规则算法。其中主要的创新点如下：

（1）采用ITEO算法对原始时间序列进行压缩预处理，线性化表示。相比于以往其他线性化的方式拟合误差更小。采用IK-Means聚类算法对线性化的时间序列进行聚类并符号化表示。增强了类间间距，聚类效果更好。聚类时相似度的度量采用三个参数的归一化表示。

（2）设计基于IAMTL结构的多元时间序列流跨事务关联规则的挖掘算法。基于这种概要存储结构的IAMTL算法能够表示多条时间序列在固定的时间T内事务的关联，在时间序列关联规则中加入固定时间T限制，划分事务集，在挖掘关联规则过程中通过剪枝提高算法的效率。I-IAMTL算法能够增量挖掘多元时间序列跨事务关联规则，更加符合时间序列数据动态产生的实际数据场景。

（3）提出采用PCM前后件窗口的方式验证时间序列关联规则在预测方面的准确性。

本课题通过对比IAMTL，I-IAMTL，tsapriori三种算法在时间效率，内存占用，预测准确率三个方面，并且在工业热电厂数据和股票数据两个数据集上进行了对比。实验结果表明本课题提出的算法在时间效率，内存占用和预测准确度上更优。本课题存在其他可以改进和创新的地方。

（1）可以利用挖掘出的关联规则通过建模的方式，或者构造分类器，提高预测的准确率。

（2）在时间序列压缩线性化表示时可以用其他的方式表示减小拟合误差。

（3）挖掘关联规则时最小支持度阈值的确定可以采用其他方式结合实际需求动态确定。若设置的太小可能会导致挖掘到很多无用的关联规则，而设置的过大则会导致无法挖掘到有效的关联规则。

# 参考文献

1. Tao F, Murtagh F, Farid M. Weighted Association Rule Mining Using Weighted Support and Significance Framework[C]//Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2003: 661-666.
2. 邱凤翔, 司风琪, 徐治皋. 电站关联规则的主元分析挖掘方法及传感器故障检测[J]. 中国电机工程学报, 2009, (5): 97-102.
3. Sang Y F, Wang Z, Liu C. Period Identification in Hydrologic Time Series Using Empirical Mode Decomposition and Maximum Entropy Spectral Analysis[J]. Journal of Hydrology, 2012, 424-425(10): 154-164.
4. 袁曾琼. 关于水文时间序列分析方法的研究[J]. 科技展望, 2016, 26(28): 321.
5. Chu H H, Chen T L, Cheng C H, et al. Fuzzy Dual-factor Dime-series for Stock Index Forecasting[J]. Expert Systems with Applications, 2009, 36(1): 165-171.
6. Janacek G. Time Series Analysis Forecasting and Control[J]. Journal of Time Series Analysis, 2010, 31(4): 303.
7. Koijen R S J, Lustig H N, Van Nieuwerburgh S. The Cross-Section and Time-Series of Stock and Bond Returns[J]. Ssrn Electronic Journal, 2012, 12(1): 98-104
8. Wang N, Ruan J. Principal Component Cluster Analysis of ECG Time Series Based on Lyapunov Exponent Spectrum[J]. Chinese Science Bulletin, 2004, 49(18): 1980-1985.
9. Zhang X S, Zhu Y S, Zhang X J. New Approach to Studies on ECG Dynamics: Extraction and Analyses of QRS Complex Irregularity Time Series[J]. Medical and Biological Engineering and Computing, 1997, 35(5): 467-473.
10. Usharani P. Fast Algorithms For Mining Association Rules in Datamining[C]// Proc of International Conference on Very Large Data Bases. 1994: 21-30.
11. Zaki M J. Scalable Algorithms for Association Mining[J]. IEEE Transactions on Knowledge & Data Engineering, 2000, 12(3): 372-390.
12. Bhalodiya D, Patel K M, Patel C. An Efficient Way to Find Frequent Pattern With Dynamic Programming Approach[C]//Nirma University International Conference on Engineering. 2013: 1-5.
13. 吴聪聪, 李忠哗, 何丕廉. 基于矩阵的最大频繁模式挖掘及其更新算法[J]. 微电子学与计算机, 2007, 24(12): 29-31.
14. Das G, Lin K I, Mannila H, et al. Rule Discovery From Time Series[C]//Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 1998: 16-22.
15. Han J, Pei J, Yin Y. Mining Frequent Patterns without Candidate Generation[J]. Acm Sigmod Record, 2000, 29(2): 1-12.
16. Sarker B K, Hirata T, Uehara K, et al. Mining Association Rules from Multi-stream Time Series Data on Multiprocessor Systems.[J]. Lecture Notes in Computer Science, 2005, 3758: 662-667.
17. Qin L X, Luo P, Shi Z Z. Efficiently Mining Frequent Itemsets with Compact FP-tree[C]//International Conference on Intelligent Information Processing. Springer US, 2004: 397-406.
18. 秦亮曦, 史忠植. 多时间序列跨事务关联分析研究[J]. 计算机工程与应用, 2005, 41(27): 10-12.
19. 王中义. 基于动态支持度的流数据关联规则挖掘[D]. 哈尔滨工业大学, 2014.
20. Velumani B, Umajothy P. Mining Temporal Association Rules from Time Series Microarray Using Apriori Algorithm[J]. Review of Bioinformatics and Biometrics, 2013, 2(2): 29-36.
21. Nair B B, Mohandas V P, Nayanar N, et al. A Stock Trading Recommender System Based on Temporal Association Rule Mining[J]. Sage Open, 2015, 5(2): 1-10.
22. Seema P M, Makkar M P. An Approach to Improve the Web Performance by Prefetching the Frequently Access Pages[J]. International Journal of Advanced Research in Computer Engineering and Technology, 2012, 1(4).
23. Tseng V S, Lin K W. Efficient Mining and Prediction of User Behavior Patterns in Mobile Web Systems[J]. Information & Software Technology, 2006, 48(6): 357-369.
24. Rudin C, Letham B, Kogan E, et al. A Learning Theory Framework for Sequential Event Prediction and Association Rules[J]. Massachusetts Institute of Technology Operations Research Center, 2012, 43(5): A294.
25. 李海林, 郭崇慧. 时间序列数据挖掘中特征表示与相似性度量研究综述[J]. 计算机应用研究, 2013, 30(5): 1285-1291.
26. Keogh E, Chakrabarti K, Pazzani M, et al. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases[J]. Knowledge & Information Systems, 2001, 3(3): 263-286.
27. 李海林, 郭崇慧. 基于云模型的时间序列分段聚合近似方法[J]. 控制与决策, 2011, 26(10): 1525-1529.
28. Agrawal R, Faloutsos C, Swami A. Efficient Similarity Search in Sequence Databases[J]. Proc.of Intl Conf.on Data Organazation, 1993, 730: 69--84.
29. Struzik Z R, Siebes A. Wavelet Transform in Similarity Paradigm[C]//Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer Berlin Heidelberg, 1998: 295-309.
30. Keogh E, Chu S, Pazzani M. An Online Algorithm for Segmenting Time Series[J]. IEEE International Conference on Data Mining series, 2001: 289-296.
31. Weng X, Shen J. Classification of Multivariate Time Series Using Two-dimensional Singular Value Decomposition[J]. Knowledge-Based Systems, 2008, 21(7): 535-539.
32. Fuchs E, Gruber T, Nitschke J, et al. Online Segmentation of Time Series Based on Polynomial Least-Squares Approximations[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2010, 32(12): 2232-2245.
33. Fuchs E, Gruber T, Pree H, et al. Temporal Data Mining Using Shape Space Representations of Time Series[J]. Neurocomputing, 2010, 74(1–3): 379-393.
34. Fuchs E, Gruber T, Nitschke J, et al. On-line Motif Detection in Time Series With SwiftMotif[J]. Pattern Recognition, 2009, 42(11): 3015-3031.
35. Hartigan J A, Wong M A. A K-means Clustering Algorithm.[J]. Applied Statistics, 2013, 28(1): 100-108.
36. 张文开. 基于密度的层次聚类算法研究[D]. 中国科学技术大学, 2015.
37. Bi F M, Wang W K, Chen L. DBSCAN: Density-based Spatial Clustering of Applications with Noise[J]. Journal of Nanjing University, 2012, 48(4): 491-498.
38. Wang W, Yang J, Muntz R R. STING: A Statistical Information Grid Approach to Spatial Data Mining[C]//International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc. 1997: 186-195.
39. Wang J, Chung F, Wang S, et al. Double Indices-induced FCM Clustering and its Integration with Fuzzy Subspace Clustering[C]//WCCI 2012 IEEE World Congress on Computational Intelligence. 2012: 1-8.
40. Borgelt C. An Implementation of the FP-growth Algorithm[J]. Osdm Proceedings of International Workshop on Open Source Data Mining Frequent Pattern, 2010: 1-5.
41. Rudin C, Letham B, Salleb-Aouissi A, et al. Sequential Event Prediction with Association Rules.[J]. Omnipress, 2011, 19: 615-634.
42. 周勇. 时间序列时序关联规则挖掘研究[D]. 西南财经大学, 2008.
43. 肖辉. 时间序列的相似性查询与异常检测[D]. 复旦大学, 2005.
44. Kel’manov A V, Pyatkin V. On the Complexity of Some Quadratic Euclidean 2-clustering Problems[J]. Computational Mathematics and Mathematical Physics, 2016, 56(3): 498–504.
45. 赖军, 李双庆. 挖掘滑动时间衰减窗口中网络流频繁项集[J]. 计算机应用研究, 2011, 28(3): 895-898.
46. Rudin C, Letham B, Madigan D. Learning Theory Analysis for Association Rules and Sequential Event Prediction[J]. Journal of Machine Learning Research, 2013, 14(1): 3441-3492.

# 攻读硕士学位期间发表的论文及其它成果

1. Chunkai Zhang, Yaqi Sun, Jianwei Guo,Tengfei Xiong. Mining Dynamic Association Rules from Multiple Time-series Data Based on Data of Power Plant[C]//IEEE Information Management,Electronic and Automation Control Conference,2016.

# 哈尔滨工业大学学位论文原创性声明和使用权限

学位论文原创性声明

本人郑重声明：此处所提交的学位论文《多元时间序列流跨事务关联规则挖掘》，是本人在导师指导下，在哈尔滨工业大学攻读学位期间独立进行研究工作所取得的成果，且学位论文中除已标注引用文献的部分外不包含他人完成或已发表的研究成果。对本学位论文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明。

作者签名： 日期： 年 月 日

学位论文使用权限

学位论文是研究生在哈尔滨工业大学攻读学位期间完成的成果，知识产权归属哈尔滨工业大学。学位论文的使用权限如下：

（1）学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文，并向国家图书馆报送学位论文；（2）学校可以将学位论文部分或全部内容编入有关数据库进行检索和提供相应阅览服务；（3）研究生毕业后发表与此学位论文研究成果相关的学术论文和其他成果时，应征得导师同意，且第一署名单位为哈尔滨工业大学。

保密论文在保密期内遵守有关保密规定，解密后适用于此使用权限规定。

本人知悉学位论文的使用权限，并将遵守有关规定。

作者签名： 日期： 年 月 日

导师签名： 日期： 年 月 日

# 致 谢

不知不觉，两年半的研究生就要过去了。在论文完成之际，在这里要感谢指导鼓励我的导师，同学和亲人。

首先要特别感谢我的导师张春慨老师，整个毕业设计及论文的撰写过程，都是在老师的精心指导下完成的。从论文开题到论文完成期间，通过无数次交流和讨论，才使得论文能够顺利完成。没有导师的指导和监督，我不可能完成论文。张老师不仅在学习上无私的帮助我，给我很多中肯的指导意见，老师严谨的治学之风和兢兢业业的生活态度也让我受益良多，教会我很多做人的道理。研究生时光短暂，却让我受益终生。

感谢我的父母，是你们养育我，给我足够的物质和精神支持。每当我因为遇到困难而沮丧难以前行的时候，是你们给予我前进的力量，让我重新拾起信心。如果没有你们，我是无法完成研究生学业的。

要感谢我的室友和实验室的同学，是你们的陪伴让我感受到友情的温暖，在写论文期间，我们互相帮助互相借鉴，学习上通过不断的讨论才能顺利完成论文。生活中室友们的关心和鼓励让我如沐春风，感谢你们在我找工作灰心丧气的时候给予我的鼓励，感谢你们贴心的照顾，研究生时光短暂，但我们的友谊长存，一生难忘。