学校代码：10246
学　　号：022021142

復旦大學

硕士学位论文

时间序列特征模式挖掘研究

姓　　名：　秦少辉
学　　号：　022021142
系　（所）：　计算机与信息技术系
专　　业：　计算机软件与理论
研究方向：　时间序列分析
攻读学位：　硕士
指导教师：　胡运发　教授
指导小组成员：　于　玉　教授
葛家祥　教授
陆晓鹏　副教授

摘 要

所谓时间序列类型数据就是按照时间先后顺序排列各个观测记录的数据集。时间序列是一类重要的数据对象，在社会生活的各个领域中都大量广泛存在。对时间序列进行分析，识别时间序列所代表的物理现象，预测时间序列的发展趋势，在实际应用中有着重要的意义。

分段是时间序列研究中的一个重要领域，对时间序列进行分段，得到时间序列的高级表示，可以减少存储空间，消除部分噪声，便于进一步对时间序列进行分析。提出了一种边缘算子分段算法，借鉴了数字图象领域中边缘算子的概念，利用边缘算子找出时间序列中的变化点(边缘点)，将这些点依次按顺序用线段连接进来，即可得到时间序列的一个 **PLR**(分段线性表示)。算法时间复杂度为 O(n)，实验表明，在大部分数据集中，边缘算子分段都有着较小的拟合误差。文中采用统计的方法，将时间序列的 **PLR** 逐段提取长度和相对斜率两个特征进行符号化，在这基础上进行了特征模式挖掘的研究。

利用数据挖掘的方法发现时间序列中的特征模式，可以帮助人们认识事物运动、变化和发展的内在规律，为人们正确认识事物和科学决策提供依据。在基于互关联后继树的时间序列特征模式挖掘方法的基础上，加入时间窗口的概念，以弥补互关联后继树这种本来应用于文本检索中的索引模型在时间序列应用中的不足之处。对互关联后继树以及挖掘算法做出了改进，使得它能挖掘出在物理上不紧密衔接特征模式，并且挖掘出的模式有着明确的意义，便于在实践中使用。实验结果表明，文中的方法可以挖掘出更多、更广泛、更具有应用价值的特征模式。

关键词：时间序列，分段，数据挖掘，特征模式，互关联后继树
中图法分类号 **TP311**

Abstract

Time series is a kind of important data existing in a lot of fields, such as stock, weather, medical, etc. With time moving, this data of time series will explode increasing. So it is important and challenging subject to research how to reduce the storage of the time series, and how to discovery valuable knowledge in large-scale time series database. These researches will help us to discover changing or developing principle of things, support to decision-making, etc.

Segmentation is an important subject in time series analysis. We can gain a **PLR(piecewise line representation)** of the original time series by segmentation. Segmentation can reduce the storage of large time series and is the foundation of many other time series researches, such as feature pattern mining, novelty detection, etc. Proposed a new segmentation algorithm: **TEOSegmentation (Temporal Edge Operator Segmentation)**. This algorithm introduced a temporal edge operator, which is similar to the edge operator in digital graphical area, to find the edge points of time series. By connect all edge points one by one with lines, we can gain a PLR of time series. We compared the reconstruction error of TEOSegmentation to important point segmentation. Experiments proved that TEOSegmentation is better than important point segmentation in most data sets. After segmentation, we symbolized the PLR to a symbol string by statistic means and researched how to mining feature patterns from this string.

Mining feature patterns from time series can help us to identifying the nature of the phenomenon represented by the sequence of observations and forecasting. Inter-relevant successive tree (IRST) is a novel mathematical model for full text database. Indexed the symbolized sequence of time series by IRST can accelerate the process of mining frequent patterns. But only tightly continued patterns can be found by this index structure. A new index model based on IRST is designed by using a time window. In this model, more non-continually successive information can be indexed. New algorithms are proposed to discover frequent patterns from this index model. Compared to IRST, this method can find more generalized patterns and have more practical value.

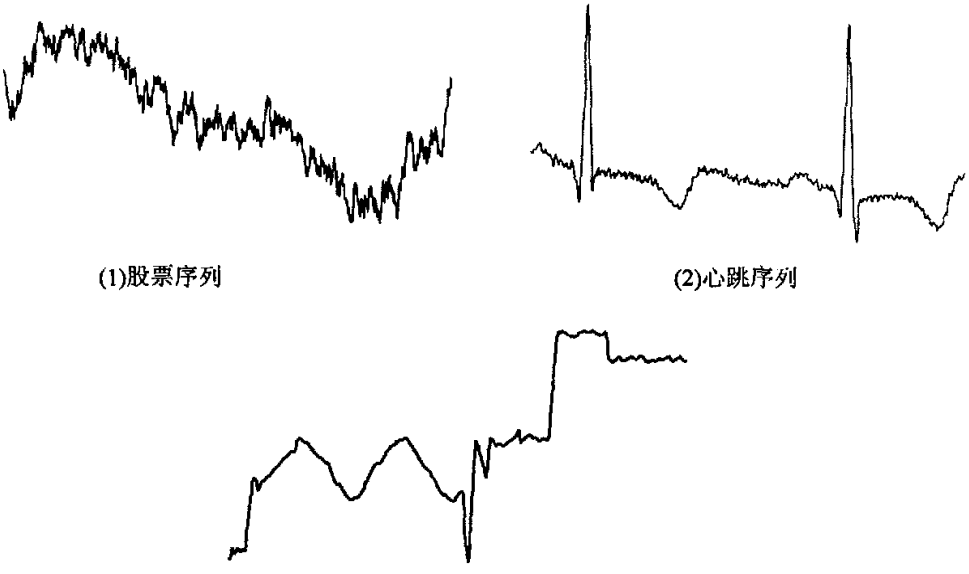
Keywords: Time series, Segmentation, Data mining, Feature Patterns, Inter-Related Successive Trees

中图法分类号 **TP311**

第一章 绪论

1.1 时间序列分析简介

所谓时间序列类型数据就是按照时间先后顺序排列各个观测记录的数据集[1]。时间序列是一类重要的数据对象，在社会生活的各个领域中都大量广泛存在，如金融证券市场中每天的股票价格变化；商业零售行业中，某项商品每天的销售额；气象预报研究中，某一地区的每天气温与气压的读数；以及在生物医学中，某一症状病人在每个时刻的心跳变化等等（如图 1-1）。不仅如此，时间序列也是反映事物运动、发展、变化的一种最常见的图形化描述方式。例如在 1974 年到 1989 年中对 15 种具有国际影响的报纸中，对其所包含的各种图形进行采样统计，结果发现其中至少 75%是采用时间序列的图形方式进行描述的[2]。通过曲线打点的方式，非常有利于人们在高级层次上来展现和理解事物的变化，而且人类早在 10 世纪前就知道通过可视化时间序列来展现事物的变化发展[3]。



(1)股票序列　　(2)心跳序列

(3)气压序列

图 1-1 几种不同领域的时间序列

然而我们应该注意到时间序列不仅仅是对历史事件的记录，展现事物变化的显示方式。随着时间推移和时间序列数据的大规模增长，如何对这些海量的时间序列进行分析处理，挖掘其背后蕴涵的价值信息。这对于我们揭示事物发展变化的内部规律，发现不同的事物之间的相互作用关系，为人们正确认识事物和科学决策提供依据等等具有重要的实际意义。因此有关时间序列分析的研究一直以来

就受到了许多研究人员的广泛重视，成为一个具有重要理论和实用价值的热点研究课题。

经典的时间序列分析的主要目标有两个 [4]:

(1) 识别模式(identifying pattern)

识别模式包括:长期或趋势移动、周期移动或变化、季节性移动或变化、不规则或随机移动。

长期移动就是分析时间序列在一个长的时间间隔内的总的移动方向。典型的方法有加权移动平均方法和最小平方方法。

周期移动或变化，这涉及到周期，也就是关于一个趋势线的长期振荡。它不需要精确地遵从一个相似模式。

季节性移动是指一个时间序列在连续几年的对应的月份出现的遵从同样的或接近同样的模式。

不规则运动，表现由于随机或偶然性事件出现的零星的的运动的特征。

(2)预测(forecasting)

根据时间序列先前的运动情况，分析序列将来可能的移动情况。

传统的分析方法主要有自回归(AR)模型、自回归移动平均模型(ARMA)、指数平滑(Exponential Smoothening),谱分解(Spectral Decomposition)等。传统的时间序列分析方法，主要集中于时间序列数据的建模、滤波和预测等问题，并取得了许多重要的成果，在实际应用中发挥了重要的作用。然而在实际分析中，往往需要对时间序列局部特征进行分析，如发现经常出现变化模式，比较不同序列在某段时间内，其运动变化是否相似等等。这些分析方法在许多应用领域中同样具有重要的意义。例如下面一些典型应用：

- 在整个证券市场中，找出股价与（1）（见图 1-1）相似变化的股票序列；再者说，给定一个特定时间序列变化模式，其长度小于序列（1）的长度，要求从（1）中找出与之相匹配的子序列；
- 在气象预报分析中，对于气压序列（2）（见图 1-1），需要找出在一段时间内那些是经常出现变化模式；再者说，给定一个气压序列和气温序列，要求找出这两个序列之间有那些是经常出现的相互关联的变化模式等。
- 在实时股票交易系统中，对于给出一些特定的、有不同含义的变化模式，如何对股票交易进行上述模式的在线检测与识别，从而提示投资者迅速、及时地进行投资交易等，如图 1-2 所示。

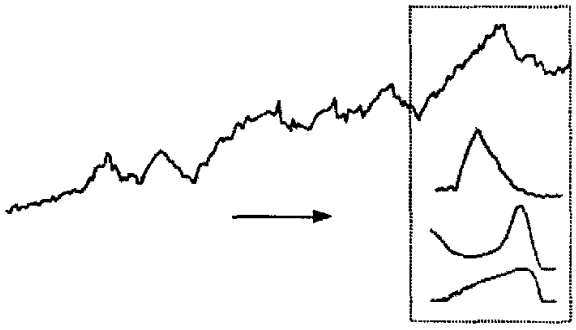


图 1-2 动态相似性检测查找

由于传统的时间序列分析技术不能完全满足新的数据库应用如数据仓库以及知识发现等领域的需求，因此从 1990 年代早期开始，时间序列数据挖掘(Time Series Data Mining, TSDM)作为一个新的研究领域应运而生，成为数据库知识发现(KDD)与数据挖掘领域的一个重要的分支[6]。

时间序列数据通常属于高维、非结构化数据，并可能受到噪声污染。复杂的海量时间序列数据已经远远超出了人类的直观理解能力。因此，研究如何有效地从这些海量时间序列中自动抽取隐含的和潜在有用的重要信息，即 TSDM，具有重要的理论价值和现实意义。在本文中，我们把数据挖掘的思想和方法引入到时间序列分析中，围绕时间序列的特征模式挖掘展开研究，重点研究了时间序列的分段、符号化以及特征模式挖掘等一系列重要问题。

1.2 相关的研究工作

在介绍时间序列挖掘之前，我们首先对知识发现与数据挖掘做一个简单的回顾。

1.2.1 知识发现与数据挖掘

1) 背景与意义

数据库中的知识发现（KDD Knowledge Discovery in Database）是数据库技术和机器学习等人工智能技术相结合的产物，是最近新兴诞生的一门数据分析的智能技术[7]。

20 世纪 80 年代末,随着数据库、互联网技术的迅速发展以及管理信息系统(MIS)及网络数据中心(IDC)的推广应用，数据的存取、查询、描述统计等技术已日臻完善，但高层次的决策分析、知识发现等实用技术还很不成熟，导致了“信息爆炸”但“知识贫乏”的现象。到了 90 年代，人们提出在数据库基础上建立

数据仓库，应用机器学习和统计分析相结合的方法处理数据，这两者的结合促成知识发现与数据挖掘技术的诞生。

所谓 **KDD** 就是从数据集中识别有效的、新颖的、潜在有用的以及最终可理解模式的处理过程。它是一门交叉性学科，涉及到机器学习，统计学，数据库，神经网络，数据可视化，并行计算等多个领域。从数据库中挖掘出的知识可以用语信息管理，过程控制，科学研究，决策支持等许多方面[8]。

数据挖掘（DM Data Mining）只是 **KDD** 中的一个步骤，它主要是利用某些特定的知识发现算法，在一定的运算效率的限制下，从数据中挖掘出有价值的知识。数据挖掘也是 **KDD** 最核心的部分，数据挖掘算法的好坏直接影响到所发现知识的好坏。目前大多数的研究都集中在数据挖掘算法及其应用上。

2) 挖掘算法研究

经过近几年的研究，人们对不同的挖掘任务，包括关联规则，序惯模式，分类，聚类，预测等提出了多种挖掘算法[6]。其中对于关联规则模式和序惯模式挖掘算法一直是数据挖掘的重点，所做的工作与取得的成果也比较多的。

- **关联规则挖掘算法**

关联规则的发现是 R.Agrawal 等人[9]首先提出的。它可以做如下形式化定义：令 $I=\{I_1, I_2, \ldots I_m\}$为项目集，$D$ 为事务数据库，其中每个事务 T 是一个项目子集（$T\subseteq I$），并有一个唯一的标识符 TID。我们说事务 T 包含项目集 X，如果 $T\subseteq X$。关联规则是形如 $X\Rightarrow Y$ 的逻辑蕴涵式，其中 $X\subset T$，$Y\subset T$ 且 $X\cap Y=\Phi$，如果事务数据库中有 s%的事务包含 XY，那么我们关联规则 $X\Rightarrow Y$ 的支持度为 s%；如果事务数据库中包含 X 的事务中有 c%的事务同时也包含 Y，那么我们说关联规则的信任度为 c%。

关联规则挖掘算法分为两步走，第一步是识别所有的频繁项目集，即其支持度不低于用户要求最低支持度的项目集。第二步是从频繁集中构造其信任度不低于用户要求最低信任度的规则。识别或发现所有的频繁集是关联规则发现算法的核心，也是计算量最大的部分。如果有 m 个项目，那么就有 m^2 个可能的频繁集。这构成了 I 上的可能解空间。然而，事实上，其中只有一小部分是确实频繁的。关联规则的构造相对较容易。尽管如此，在这一步仍有重要的问题有待进一步研究，如怎样在已发现的大量关联规则中选取真正令人感兴趣的规则，即所谓规则的兴趣度评价问题。这里，我们组要考虑怎样发现所有的频繁项目集。目前已有不少方法，其中最著名的仍然是 R.Agrawal 本人在他自己的 AIS 算法基础上提出的 Apriori 算法。

Apriori 算法核心是利用这样一个性质：频繁项集的所有非空子集都必须也是频繁的。该算法基于这样的事实，算法使用频繁项集的先验知识，Apriori 使

用一种成做逐层搜索的迭代方法，k-项集用于探索(k+1)-项集。首先找出频繁 1-项集的集合。该集合记做 L_1。用 L_1 找频繁 2-项集的集合 L_2，而 L_2 用于找 L_3，如此下去，直到不能找到频繁 k-项集。找每个 L_k 需要依次数据库扫描。

虽然 Apriori 算法能最终挖掘出所有的关联规则，但是由于处理的数据库的量非常大，算法的效率显得十分重要。后来有一些研究人员对算法的连接和剪支过程进行各种优化。

如[10]提出了称为 AprioriTid 的改进算法，该算法提出了在每一步（第一步除外）计算候选频繁项目集的支持度时不需要浏览整个事务数据库。它认为如果一个事务不包含任何一个 k-候选模式频繁项目集，则在扫描时不必在扫描不包含任何有关这个事务的元素。

而[11]提出的 AprioriPro 算法，其与 AprioriTid 算法的基本思想是一致的，也是减少对数据集的频繁项的扫描，不同的是 AprioriPro 算法是通过在原有的数据集上增加一个属性，通过这个属性的取值来减少对某些事务的频繁扫描。

[12]是结合 Apriori 和 AprioriTid 两种算法，提出了一种混合挖掘算法 AprioriHybrid 算法，其基本思想是在扫描的早期使用 Apriori 算法，当候选模式集中记录条数小到可以知己放进内存时就转向 AprioriTid 算法。

这些算法虽然比 Apriori 算法在挖掘效率有一些提高，但是它们本质上没有什么变化，都要在挖掘过程中生成大量的候选模式项。

- **序惯模式挖掘**

序惯模式挖掘最早也是有 R.Agrawal 首先提出的[13]，序惯模式的问题定义与关联规则很相似，它们之间的区别可以用下列例子描述，关联规则描述的是在一次购物中所购买物品之间的关联关系，而序惯模式则是描述同一顾客在多次购物所购物之间可能存在的某种关联关系。

序惯模式的发现方法与关联规则的发现方法大致相同，Agrawal 在 Apriori 算法的基础上提出了三个序惯模式的发现算法 AprioriAll,AprioriSome 和 DynamicSome，序惯模式的发现可以视为在含时间属性的数据库中发现关联。但应注意两者的序列包含判断是不一样的，而且关联规则仅仅是发现事物内部(intra-transaction)的模式，而序惯模式则是发现事务之间（inter-transaction）的模式。显然，所有频繁序列的集合是所有频繁项目集的超集。

- **事件序列中的频繁情节发现研究**

Agrawal 的关联规则发现算法的基本思想具有广泛的适用性，其在事件序列中的一个典型应用就是 Heikki. Mannlia 等首先提出的事件序列中频繁情节(Frequent Episodes)的发现研究[14]。所谓情节就是在给定长度的时间区间内出现

的事件的有序集合。而频繁情节是指在事件序列中具有一定出现频率的情节。如果在事件序列中发现频繁情节，我们就可以生成描述或预测该序列的行为。

考察一个事件序列（e,t），其中 e 为事件类型，t 为该类型事件的发生时间。如果我们记所有事件的类型为 E，则一个情节 ω 是一个元素取自 E 的偏序集合。给定一事件序列 s={（e1,t1），…（en,tn）}，s 中宽度为 w 的一个片段，s_t 由事件序列 s 中的事件（ei,ti）构成，其中 $t \leqslant ti \leqslant t+w$，我们称情节 ω 出现在 s_t 中，如果在 s_t 中，存在与 ω 中事件类型一致且事件发生的顺序也与 w 的偏序关系一致的事件。如果某情节 ω 在足够数量的原始事件序列 s 的片段中出现，则它是频繁的。

可以运用关联规则的发现算法的基本思想，首先寻找长度为 1 的频繁情节集，记为频繁 1-情节序列集 L_1，然后用 L_1 生成候选 2-情节序列集 L_2，接着根据实际事件序列计算各个候选的出现频率，形成频繁 2-情节序列集 L_2，如此下去，一直反复执行到没有新的候选模式产生为止。

1.2.2 时间序列挖掘

随着数据挖掘研究领域的拓展，针对时间序列数据的数据挖掘研究日益受到人们的关注。

Heikki Mannila 等将事物数据库中关联模式和序惯模式的算法思想引入到事件序列中，提出了从事件序列中发现频繁情节的算法[14]。该文献首次给出频繁情节的定义及搜索算法，所谓频繁情节就是指频繁出现的具有一定时间跨度的事件结构模式，包括并行频繁情节和串行频繁情节，采用自动机搜索算法实现了多种频繁情节的发现。

Claudio Bettini 等也对时间序列中静态结构的发现问题进行了研究[15][16][17]，与 Heikki Mannila 不同的是，Bettini 允许用户指定感兴趣的事件结构，并在其算法中增加了对多时间粒度关系的支持。为实现多粒度挖掘，Bettini 定义了带时间粒度的静态约束和带时间粒度的事件结构，在计时自动机基础上提出了一种称为带时间粒度的计时自动机搜索算法，实现了多粒度复杂结构的发现。

Gautam Das 等对从时间序列中发现关联规则进行了研究[18]，这里规则是对时间序列中不同模式间关系的一种描述，[18]的主要贡献在于给出了一个将原始时间序列转换成一个有各个模式表示符组成的符号序列的一般性方案，该方案由

三部分组成，即分割，聚类和符号替换。然后采用序惯模式发现算法实现了符号序列中规则的发现。

Richard J.Povineli 提出了一个从时间序列中发现具有预测功能的静态模式的挖掘算法[19][20]，该算法将相空间重构法与数据挖掘中静态模式的发现思想相结合，利用遗传算法实现了时间序列的模式聚类与模式提取。

这些算法基本上是对原始时间序列的数据进行挖掘，挖掘的结果形式在每个时间间隔上的模式，但是由于时间序列由于其自身波动性与取值的连续性，因此这种模式结果在实际运用上的价值仍然值得商榷。虽然[18]提出了对时间序列进行分割，具有一定的改进，但是这种分割是通过一个时间窗口，窗口的大小作为参数输入，因此这种输入的随意性，可能直接影响了挖掘的效果。此外，它们大部分还是利用 Apriori 算法作为其挖掘算法，由于该算法需要生成大量的候选模式，因此这也影响了时间序列的挖掘效率。

Eamonn Keogh 等提出了一种从时间序列中发现异常模式的算法[21]。他将异常模式定义为出现频率远大于(或远小于)欺望值的模式。所用的方法称为 Tarzan 算法，其基本原理为：先使用一个时间窗口来提取分段的特征，通过聚类将时间序列转化为一个符号序列，然后利用后继树来计算各个模式的出现频率。

曾海泉[22]等人提出了一种新颖的时间序列特征模式挖掘方法：根据重要点来分段，采用互关联后继树索引模型[23]来挖掘时间序列中的特征模式。根据重要点来分段的方法与其他方法中常用的利用时间窗口来分段的方法相比，有着较高的效率。而且应用在股票交易市场中，分段后的线段代表着一段时间的平均涨幅，更能描绘股票市场中大家关心的涨跃特性。利用互关联后继树模型的挖掘算法也有着较高的效率，但互关联后继树模型本是一种应用在文本索引中的模型，将其应用到模式挖掘中，存在着一个重大缺陷：即它只能挖掘出紧密衔接的特征模式。

1.3 本文工作

1.3.1 研究目标

根据前面对国内外时间序列挖掘方面研究现状的分析，不难发现目前的研究对以下几方面的问题还没有得到很好的解决：

1) 序列的分段多采用时间窗口抽取特征来分段，但这种分段方法的准确性较差，因为其只关注了序列局部的特征，而忽略整个序列的变换特征。

2) 时间序列挖掘大部分的研究成果是基于 Apriori 算法，该算法最大的缺陷就是在挖掘过程中必须生成大量的候选模式，这大大影响了挖掘效率。
3) 曾海泉等人[22]提出的根据重要点分段、基于互关联后继树的挖掘算法非常新颖，而且挖掘是无须产生候选模式，效率较高。但在符号化、规则的发现等方面仍有很多值得改进的地方。

基于以上观点，我们对时间序列特征模式挖掘的研究目标是：丰富目前的时间序列分段及符号化的方法，使得符号化序列更能描述原时间序列的特征；进一步改进基于互关联后继树的挖掘算法，挖掘出更多的潜在规则。

1.3.2 研究内容

针对上述研究目标，我们在时间序列分段算法、符号化方法以及特征模式挖掘等方面展开研究：

- **分段及符号化**

由于时间序列取值的连续性和变化的波动性，造就了它与一般的数据挖掘不同。因此需要对它进行分段，将以实数形式表达的时间序列转换成以相对抽象的符号形式表达的符号序列。在大量的时间序列方面的文献中，或者将分段算法作为其主要贡献，或者利用了一种或多种分段算法作为其进一步研究的基础。因此分段及符号化算法是时间序列分析中的一个重要研究领域。

目前的时间序列分段算法主要分为三类：移动窗口分段(Sliding-Windows)、自顶向下分段(Top-Down)和自底向上(Bottom-Up)分段。各种分段法各有各的特点，在不同的数据领域中，各个算法的表现也各不相同。

重要点分段算法是一种新型的分段算法，其关注的斜率特征也正好是股价序列中最重要的特征（涨和跌），在股市时间序列分析中有着很好的应用前景。目前的重要点分段算法的符号化通常是用长度天数和一个斜率符号来表征一条线段，存在着两个缺点：直接使用长度天数来表征线段过于刚性，不利于特征模式的挖掘；斜率的符号化多采用经验分类，根据不足。

我们对重要点算法进行了改进，对长度也进行了符号化，并对大量的股价序列进行统计得出了各个斜率符号的阈值。另外还讨论了一种边缘点分段算法，并与重要点分段算法进行了比较。

- **时间序列特征模式挖掘**

时间序列不仅是数据记录的形式，而且也反映了事物发展变化规律。因此从时间序列发现知识，我们需要对每个时间序列变化的特征模式进行挖掘。时间序列特征模式是时间序列变化的高层含义概括，因此它的挖掘不仅要有视觉上的图形描述，而且还应该有明确的实际意义。目前大部分的时间序列特征模式挖掘算

法都是基于 Apriori 算法，由于挖掘过程中要产生大量的候选模式，因此挖掘效率较低。

曾海泉等人[22]提出的基于互关联后继树的挖掘算法不需要产生候选模式，效率较高。但互关联后继树模型本是一种应用在文本索引中的模型，将其应用到模式挖掘中，存在着一个重大缺陷：即它只能挖掘出紧密衔接的特征模式。我们对这种算法做出了改过，加入一个时间窗口，使它可以挖掘出更广义的特征模式。

1.3.3 本文结构

本文围绕着时间序列的分段以及时间序列特征模式挖掘方法与实现技术展开，共分为四章，具体内容安排如下：

第一章 绪论

介绍时间序列分析的任务，引出研究相关问题，并概述了国内外最近有关的研究动态，介绍了本文的主要研究内容。

第二章 分段及符号化

该章主要讨论了分段及符号化问题。首选介绍了目前流行的各种时间序列分段及符号化方法，分析其存在的不足之处。重点讨论了基于重要点的分段及符号化方法并做出了改进，另外还讨论了基于边缘点的分段算法并与重要点算法进行了比较。

第三章 时间序列特征模式挖掘

该章主要解决了时间序列特征模式的挖掘，首先介绍了传统的时间序列特征模式挖掘方法，指出了它的一些不足之处。重点讨论了基于互关联后继树的挖掘算法及其缺点，并做出了改进。

第四章 原型系统

介绍了由我独立开发的一个时间序列特征模式挖掘原型系统，实现了本文中提出的算法，包括从原始数据集的处理到挖掘出特征模式的全部功能。

第五章 结论

该章对全文的工作进行了总结，并对今后的工作提出了新的研究方向。

1.4 几个术语的说明

术语名	英文名	解释
时间序列	Time Series	由按时间有序的实数组成的集合
分段	Segmentation	将时间序列分为一个个片段，通常为线段
符号序列	Symbol Sequence	将时间序列分段符号化后得到的序列
模式	Pattern	指用户感兴趣的特定序列部分，常常是时间序列的子序列
特征模式	Feature Pattern	时间序列中出现频率大于一定阈值的模式，即频繁模式

第二章 分段及符号化

目前研究的时间序列特征模式挖掘方法通常分为两步，分别是时间序列特征描述和挖掘算法设计。现在用得比较多的特征描述方法是线段化的描述方法：先将时间序列分段，再对一个个分段进行符号化，把时间序列转化为一个字符串序列；然后再利用丰富的字符串挖掘算法来进行特征模式挖掘。为 E. Keogh 等人[24]将时间序列数据挖掘研究任务分为四个方面:

1)索引(基于内容的查询)。给定一个时间查询序列 Q 以及某个相似性/不相似性度量 D(Q,C)，发现时间序列数据库 DB 中与 Q 最匹配的那些时间序列。

2)聚类。在某种相似性/不相似性度量 D(Q,C)下，将时间序列数据库 DB 中的时间序列自然的分组。

3)分类。给定一个未加标签的时间序列 Q，将其划分到某些预定义类中的一个类。

4)分段。给定一个包含 n 个时间点的时间序列 Q，构建一个模型 $\overline{Q}$，将 Q 分为 K 个部分〔K<<n〕使得 $\overline{Q}$ 近似 Q。分段有两个主要的应用，它可以被用来检测生成时间序列的系统的变化，即变化点检测；它也可以用来创建时间序列的高级表示以便索引、聚类和分类。

可见分段是时间序列分析中的一个重要研究领域，在大量的时间序列方面的文献中，或者将分段算法作为其主要贡献，或者利用了一种或多种分段算法作为其进一步研究的基础。尽管分段后的片段可以用任意次的多项式来表示，但通常人们均用线性的函数来表征这些分段。很直观的，我们用由 K 条直线组成的 **PLR(Piecewise Linear Representation 分段线性表示)**来模拟一个长度为 n 的时间序列 Q。图 2-1 是一个例子：

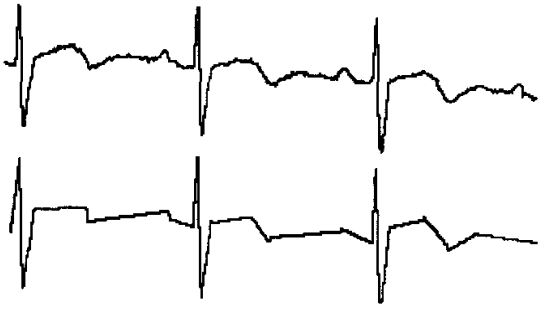


图 2-1 一个时间序列及其 PLR 的例子

通常 K<<n，所以使用 PLR 来模拟时间序列可以提高数据存储、转换以及计算的效率。特别是对于目前的时间序列，动辄上万条甚至百万条数据，使用 PLR 就显得更加意义重大。

下面的内容安排如下：第一部分介绍了传统的时间序列分段方法；第二部分详细讨论了重要点分段及符号化方法；第三部分详细讨论了边缘算子分段；第四部分对重要点分段和边缘算子分段进行了比较。

2.1 传统的分段算法

传统的时间序列分段算法主要分为三类：移动窗口分段(Sliding-Windows)、自顶向下分段(Top-Down)和自底向上分段(Bottom-Up)。各种分段法各有各的特点，在不同的数据领域中，各个算法的表现也各不相同。

2.1.1 移动时间窗口(SW)

移动窗口分段（Sliding Window）的原理是从时间序列起始点开始利用最小二乘法进行拟合，当拟合误差超过给定的最大误差时，在该处进行分段，然后在下一个点上重新进行拟合，直到把整个序列拟合完毕。具体算法由算法 2-1 所示。

这种分段方法最简单，最直观，计算量也较小为（n 为序列的长度，L 为各子段的平均长度）[25][26] [27]，但是它分段拟合的准确性比较差，因为其只关注了序列局部的最小误差，而忽略了序列整体的变化特征。

```
算法2-1   Seg_TS = Sliding_Window(T , max_error)
    {   anchor = 1;      // 设置起始点;
        while not finished segmenting time series
          i = 2;
            while calculate_error(T[anchor: anchor + i ]) < max_error
          //从起始点后面开始扫描，直到出现分段误差超过与阈值;
            i = i + 1;
            end;
          Seg_TS =concat(Seg_TS,create_segment(T[anchor:anchor+(i-1)]);
         //在i点分段序列，然后从i+1点后继续开始分段;
          anchor = anchor + i;
         //设置新的起始点;
        end;
        }
```

2.1.2 自顶向下分段(TD)

自顶向下分段（Top-Down）其寻求的是时间序列最佳分段，每一次分段都要测试分段后全部子段的拟合误差是否超过用户给定的最大误差，如果没有超过，则继续分段其左、右子序列，直到序列分段拟合完毕。下面算法 2-2 给出了其具体分段过程。

```
算法2-2   Seg_TS = Top_Down(T , max_error)
  {     best_so_far = inf;    //开始设置分段效果，最差极限；
        for i = 2 to length(T)- 2   // 在整个序列中寻求第一个最佳分段点.
          improvement_in_approximation = improvement_splitting_here(T,i);
          if improvement_in_approximation < best_so_far
         //获取最佳的分段位置
          breakpoint = i;
         //目前最佳的分段误差；
          best_so_far = improvement_in_approximation;
          endif;
        end;
       // 对左边的线段，递归调用分段算法；.
        if calculate_error(T[1:breakpoint]) > max_error
          Seg_TS = Top_Down(T[1: breakpoint]);
        endif;
       // 对右边的线段，递归调用分段算法；
        if calculate_error( T[breakpoint + 1:length(T)] ) > max_error
          Seg_TS = Top_Down(T[breakpoint + 1: length(T)]);
        endif;
        }
```

从算法可以看出，从上到下的分段方法注重了序列的整体分割误差，因此它比前面的移动窗口分段在分段效果上要好[28] [29]，然而该算法却比移动分段方法有更高的计算时间复杂度为 $O(n^2K)$（K 为序列的分段数目，$K=n/L$），此外它在出入参数数量上要多，不仅要给出整体上的分段误差阈值，而且还要给出整个期望的分段数目以及每段分割误差阈值等。

2.1.3 自底向上分段(BU)

自底向上分段（Bottom-Up）是从上向下分段的反过程，它是通过合并各个连接点来达到分段的目的，其过程是：首先分别计算序列左右两相连子段合并后拟合误差，然后选取最小误差的相连子段进行合并直到超过给定的最大误差值。

在合并两相连子段时，拟合合并后的长子段要重新放入序列中，并计算它与左右相连子段的拟合合并误差。文献[30][31][32]中运用的分段方法也就是这种分段算法。

下面我们给出其具体分段过程描述算法，见算法 2-3。

```
算法2-3   Seg_TS = Bottom_Up(T , max_error)
    {  for i = 1 :  2 :  length(T)
        // 在整个序列中创建需要合的各个子段.
          Seg_TS = concat(Seg_TS, create_segment(T[i: i + 1 ]));
       end;
       for i = 1 : length(Seg_TS) – 1
      // 找出各子段合并误差.
          merge_cost(i)=calculate_error([merge(Seg_TS(i), Seg_TS(i+1))]);
       end;
       while min(merge_cost) < max_error // 设置结束条件
          i = min(merge_cost);    //找出最小合并拟合误差的子段.
          Seg_TS(i) = merge(Seg_TS(i), Seg_TS(i+1)); //合并.
        //重新设置各子段记录.
          delete(Seg_TS(i+1));
          merge_cost(i)= calculate_error(merge(Seg_TS(i), Seg_TS(i+1)));
          merge_cost(i-1)= calculate_error(merge(Seg_TS(i-1), Seg_TS(i)));
      end; }
```

由底自上的合并分段方法既注重了时间序列在各个子段合并后的拟合误差，又通过选用最小合并代价来把握时间序列在整体上分段特征，因此其分段的效果比移动窗口的方法要好，而且比自顶向下分段方法要有较小的计算时间复杂度，为 $O(Ln)$。但是这种方法仍然采用了最小二乘法作为其拟合方法，在计算时间上仍然也是比较耗时，而且在输入的参数上，也要有各个子段间的合并误差阈值和合并后的期望线段数量等。

2.1.4 几种分段算法的比较

衡量分段算法好坏的有很多种，其中最明显的方法是比较固定数目分段的拟合误差（**Reconstruction Error**）。拟合误差是指原始数据与分段表示之间的欧氏距离。表 2-1 给出了三种算法在不同的数据集中的表现：

Dataset	Best Algorithm	Second-Best Algorithm	Third-Best Algorithm
Soiltemp	TD (522.6)	SW (538.0)	BU (538.1)
Darwin	TD (575.2)	BU (821.0)	SW (833.9)
pHdata	SW (3.590)	TD (4.013)	BU (4.037)
Winding	SW (6.883)	BU (113.0)	TD (117.6)
Balloon	BU (168.1)	TD (224.5)	SW (234.1)
Network	BU (11.02)	SW (13.62)	TD (891.4)

表 2-1 三种分段算法在不同的数据集中的表现

从表 2-1 中可以看出，在不同的数据集中，最好的算法各不相同。说明算法的好坏对数据集有着很强的依赖。不过在大部分情况下，自底向上算法要优于其他两种算法。在对 69 个数据集的比较中，自底向上算法在 47 个数据集中优于自顶向下算法，在 58 个数据集中优于移动窗口分段算法[24]。

2.2 重要点分段

本节介绍一种与传统分段方法不同的分段方法：基于重要点的分段方法。

2.2.1 重要点分段介绍

所谓的重要点是在序列变化中视觉上有着相对重要影响的观测点，实际上就是序列中的某些局部极小、极大点。如图 2-2 所示。通过选取重要点进行分段能有效地消除噪声对序列的影响，保存序列变化的主要特征模式。

为了描述方便，我们在本节以中，设 S=<x1=(v1,t1),…,xn=(vn,tn)>为一时间序列，其中 vi 是在时间 ti 上的观测值，且 ti+1-ti=Δ（i=1,…,n-1），并且假定Δ=1（表示一个时间间隔单位），且 t1=0。

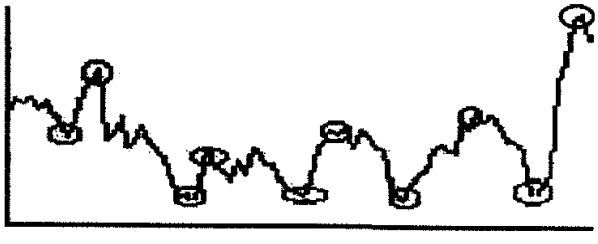


图 2-2 序列的重要点

定义 2-1（重要极小点）给定常量 R>1 和时间序列{$<x_1=(v_1,t_1),\ldots,x_n=(v_n,t_n)>$}，如果称数据点 $x_m(1\le m\le n)$是一个重要的极小点，则应满足如下条件之一：

1） 当1<m<n时，存在下标i和j且$1\le i<m<j\le n$，使得（i）v_m是$v_i,\ldots,v_j$中的最小

值，（ii）$v_i/v_m \geq R$且$v_j/v_m \geq R$成立；

2） 当m=1即x_m为时间序列的起始数据点时，存在j且m<j≤n，使得（i）v_m是$v_m,\ldots,v_j$中的最小值，（ii）$v_j/v_m \geq R$成立；

3） 当m=n即x_m为时间序列的终止数据点时，存在下标i且1≤i<m，使得（i）v_m是$v_i,\ldots, v_m$中的最小值，（ii）$v_i/v_m \geq R$成立。

定义 2-1 的直观含义是：x_m 是子序列段 $x_i,\cdots,x_j$ 的最小值，并且此段上的两个端点的值大于 x_mR。R 是可控制选取的参数，R 值越大则被选中的相对重要点就越少，时间序列线段化描述就越粗；反之，R 值越小，则被选的重要点就越多，线段化描述越细。因此，通过选择 R，可以在不同的精细程度上进行数据挖掘。对应地，我们给出重要极大点的定义如下。

定义2-2（重要极大点）给定常量R>1和时间序列$\{<x_1=(v_1,t_1),\ldots,x_n=(v_n,t_n)>\}$，如果说数据点 $x_m(1\leq m \leq n)$是一个重要的极大点，则应满足如下条件之一：

4） 当1 <m< n时，存在下标i和j且1≤i<m<j≤n，使得（i）v_m是$v_i,\ldots,v_j$中的最大值，（ii）$v_m/v_i \geq R$且$v_m/v_j \geq R$成立；

5） 当m=1即x_m为时间序列的起始数据点时，存在j且m<j≤n，使得（i）v_m是$v_m,\ldots,v_j$中的最大值，（ii）$v_m/v_j \geq R$成立；

6） 当m= n即x_m为时间序列的终止数据点时，存在下标i且1≤i<m，使得（i）v_m是$v_i,\ldots, v_m$中的最大值，（ii）$v_m/v_i \geq R$成立。

根据上述定义，我们不难给出选取序列重要点的算法框架：

算法 2-4：Select_Important_Points(S, R, S^C);
输入：时间序列S (其长度为N)，选取参数R；
输出：由重要点组成的序列SC；
步骤：

1) i=find_first_important_point(S^C); *//依据定义找出S中的第一个重要点, i是其时间位置。*
2) if i≤N and $v_i \geq v_1$ then i=find_minimum(i)；*//若 xi是极大重要点，则依据定义找i后面的极小重要点位置 I。*
3) while i≤N do
4) { i= find_maximum (i)；*//找i后面的极大重要点位置i。*
5) if i≤N then i= find_minimum(i)； } *//找i后面的极小重要点位置 i。*

从算法 2-4 可看出，该算法具有如下几个特点：第一，整个算法只要对序列扫描一次，分段过程只需一些简单的比较计算，而无需复杂的最小二乘法计算，其时间复杂度是 O(N)；第二，该算法支持序列的在线分段；第三，算法在输入的参数上只需一个 R。

根据算法 2-4 获得的重要点集，将这些重要点依次用线段连接起来，便可得到一个由线段表示的趋势变化的模式组成的序列集合 S^C，即时间序列的 **PLR(分段线性表示)**。

2.2.2 符号化

在找出时间序列的重要点，将时间序列 S 转换为 **PLR** 后，还需要将这些线段模式进行符号化，将时间序列转换为一个字符串序列，这样才能更方便的利用多种多样的字符串处理方法来挖掘特征模式。

将时间序列离散为一个字符串序列在很多学科中已经有了广泛的应用，如航空、医学、化学等[42]。在数据挖掘领域，这种方法也被应用到相似性查询[43]以及变化点检测[44]等问题中。

对于重要点分段，文献[22]中对每一个线段模式，提取了两个特征：长度和斜率。这里的斜率是指相对斜率，余文中提到的斜率，若不做特殊声明，皆指相对斜率。事实上，对于线段$<(t_i,v_i)$ ，$(t_{i+1},v_{i+1})>$而言，用其长度 l_i 和相对斜率 ρ_i $=((v_{i+1}-v_i)/v_i)$ / $(t_{i+1}-t_i)$组成的特征集合$<l_i,\ \rho_i>$来表征这一线段。然后再根据经验将斜率 ρ_i 进行符号化，这样可以把一个时间序列 S 转变成形为$<l_i,\ A_j>$, i=1,..,n; j=1,…,k｝的由长度和符号描述的有序线段数字加符号组成的混合序列。

这种符号化方式特别适合于应用在股票市场的时间序列中，因为相对斜率刚好反应了我们最关心的股票涨跌特性。但是它没有对长度特征进行符号化，这样两个线段$<10, A_i>$，$<11, A_i>$会被当做两个完全不同的模式，这将不利于挖掘后面的特征模式挖掘。而对斜率的符号化，它也只是利用了经验的方法，将斜率的取值分为一些范围，每个范围对应一个符号，这种分类方法的科学性不佳。

本节后面的内容分为两部分，分别对长度和斜率的符号化方法进行讨论。

2.2.1.1 长度的符号化

假设 Σ 是一个有限的字符集，共有 a 个字符，对于一个任一给定的长度 l，要将其映射到 Σ 中的某一个字符，关键就是要确定这个字符集中的每个字符对就的长度范围，即阈值数组 *boundaries*[*a*-1]。这样，如果 $l <=$ *boundaries*[1]，则将 l 映射为第一个字符；如果 *boundaries*[*i*-1]$<l <=$ *boundaries*[*i*]($i > 1$)，则将 l 映射为第 i 个字符；如果 $l >$ *boundaries*[*a*-1]，则将 l 映射为第 a 字符。

本文中采用统计学的方法来确定每个字符的阈值，认为每个字符所代表的长度在时间序列中出现的次数是相等的，下面给出具体算法：

```
算法 2-5: Proc get_Length_Boundaries(length_array L, int a, boundaries[a-1])
输入：长度数组 L, 字符集大小 a
输出：阈值数组 boundaries
步骤：
    Sort(L);
    For i=1, a-1
        let pointer = I * |L| / a;
        let boundaryes[i] = L[pointer];
    return boundaryes;
```

该方法的关键是需要对大量的数据进行统计，这样才能得到合理的阈值范围。对于不同领域的数据，也应采用不同的阈值，应分别进行统计。

2.2.1.2 斜率的符号化

对于斜率 ρ 的符号化，文献[22]中采用了一种经验的方法，结合领域知识对斜率进行分类符号化，如对股票时间序列，当 ρ 为 1%-2%时记为小涨； ρ 为 5%-10%时为大涨等等。这种方法虽然简单，容易理解，但其合理性值得怀疑。

本文中对斜率的符号化采用了类似于前一节长度符号化的方法，首先统过对大量数据的统计确定各个符号的阈值，然后再将斜率映射为符号，算法如下：

```
算法 2-6: Proc get_Slope_Boundaries(slope_array  ρ , int a, boundaries[a-1])
输入：斜率数组 ρ , 字符集大小 a
输出：阈值数组 boundaries
步骤：
    Sort( ρ );
    For i=1, a-1
        let pointer = I * | ρ | / a;
        let boundaryes[i] =  ρ [pointer];
    return boundaryes;
```

采用这种方法，可以使得每个符号代表的斜率在时间序列中的出现几率相同。在实际应用中，也可以结合特定领域的知识，对阈值进行适当调整。

下面举一个对线段化的时间序列进行分段的简单例子，如图 2-3 所示，其中共有三个长度符号：a, b, c；四个相对斜率符号：A, B, C, D。长度符号 a 代表长度 1，长度符号 b 代表长度 2，长度符号 c 代表长度 4。斜率符号 A 代表斜率 0.33，斜率符号 B 代表斜率-0.33，斜率符号 C 代表斜率-0.3，斜率符号 D 代表斜率 0.75。这样，原时间序列符号化后的结果为 aAaBbAbCcD。

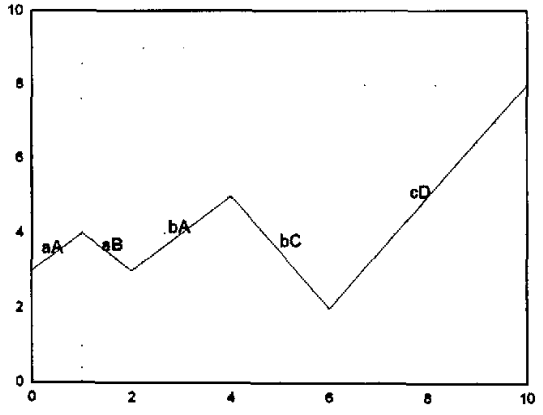


图 2-3 时间序列符号化的简单示例

2.3 边缘算子分段

本节提出了一种基于时态边缘算子的时间序列分段方法。边缘算子分段的基本思想与重要点分段相似，也是先找到一些特殊的点，然后将这些点依次连接起来，就得到了时间序列的一个 **PLR**(分段线性表示)。不同的是，在重要点分段方法中，这些特殊的点是“重要点”；面在边缘算子分段中，这些点是“边缘点”。

边缘算子本是应用在数字图象领域中，用来发现图象中物体的边界像素点。本节借鉴了数字图像研究领域中边缘算子的基本思想，将边缘算子与时间序列的特点结合起来，提出了时态边缘算子(Temporal Edge Operator, TEO)，根据时态边缘算子计算时间序列上各点的边缘幅度，选取部分边缘幅度的局部极值点作为模式的边缘点(端点)，将这些边缘点依次用线段连接，就得到了时间序列的一种分段线性表示，称为时间序列的 TEO 表示。

2.3.1 算法

时间序列的 TEO 表示方法最重要的问题是如何寻找模式的边缘点。如果时间序列中的某个点是边缘点，那么在局部范围内，位于该点两端的时间序列将呈现不同的变化趋势。在数字图像研究中，边缘算子通常用于检测图像的边缘，我们借鉴了边缘算子的基本思想，结合时间序列的自身特点，提出时态边缘算子来寻找模式的边缘点。

定义 2-3 时态边缘算子

设时间序列 $X=\langle x_1,x_2,\cdots,x_n\rangle$，时态边缘算子定义为：

$$Teo(t,u)=\{w(i)*(x_{t+i}-x_t)| \quad i=-1,-2,\cdots,-w,0,w,\cdots,2,1\} \qquad \text{(公式 2.1)}$$

其中 u 表示时态边缘算子的检测窗口长度，$w(i)$ 表示检测窗口的位置 i 上的权重，可以根据不同数据特征，选择不同的权重函数。以下为了简便，如无特殊说明，本文一律采用 $w(i)=abs(i)$，即越靠近检测窗口中心的点权重越高，对 t 时刻的序列点是否模式边缘点的影响越大。

定义 2-4 时间序列的 TEO 表示

设时间序列 $X=\langle x_1,x_2,\cdots,x_n\rangle$，其边缘点集合为 $\{x_{i_1},x_{i_2},\cdots x_{i_K}\}$，其中 $1\le i_1<i_2<\cdots<i_K\le n$。那么该时间序列的 TEO 表示为

$$X^T=\begin{cases}L(x_{i_1},x_{i_2}), & t\in[i_1,i_2]\\ L(x_{i_2},x_{i_3}), & t\in[i_2,i_3]\\ \cdots\cdots & \\ L(x_{i_{K-1}},x_{i_K}), & t\in[i_{K-1},i_K]\end{cases} \qquad \text{(公式 2.2)}$$

其中 $L(x,y)$ 表示连接边缘点 x 和 y 之间的的线段函数。在不引起混淆的情况下，公式(2.2)也可以简单表示为

$$X^T=\langle L(x_{i_1},x_{i_2}),L(x_{i_2},x_{i_3}),\cdots,L(x_{i_{K-1}},x_{i_K})\rangle \qquad \text{(公式 2.3)}$$

采用时态边缘算子与时间序列作卷积运算，就能得到时间序列上各点的边缘幅度。根据边缘幅度的大小及其分布，选择模式的边缘点，依次连接这些边缘点就构成了时间序列的 TEO 表示。

那么如何寻找模式的边缘点呢？简单的粗鲁算法就是选出边缘幅度最大的 K 个点作为边缘点。该方法看起来非常直观，而且体现了“边缘幅度最大的序列点最有可能是边缘点”的特性。但是实际上，这种方法得到的 TEO 表示与原时间序列的拟合效果并不令人满意。主要有两方面的原因：一方面，时间序列一般具有时变非稳态的特点，随着时间的推移，时间序列的数据特征已经发生改变，不同时段的边缘幅度没有可比性；另一方面，时间序列的数据之间相关性很强，相邻的序列点的边缘幅度很接近，粗鲁算法会选择许多邻近的序列点作为边缘点，在选择相同数目的边缘点情况下，将会降低时间序列表示的质量。我们提出了一种边缘点选择算法，充分考虑了时间段对边缘幅度的影响，选择局部范围内的边缘幅度极值点作为边缘点。

定义 2-5 极大极小边缘点

设时间序列 $X=\langle x_1,x_2,\cdots,x_n\rangle$，那么有

(1) 序列点 x_k 是极大边缘点，如果存在 $i<k<j$，使得 x_k 是 $X_{i:j}$ 中的边缘幅度的极大点；

(2) 序列点 x_k 是极小边缘点，如果存在 $i<k<j$，使得 x_k 是 $X_{i:j}$ 中的边缘幅度的极小点。

算法 2-7：基于时态边缘算子的线性分段算法 TEOSegmentaion

算法输入：时间序列 $X=\langle x_1,x_2,\cdots,x_n\rangle$，参数 u,d

算法输出：时间序列的 TEO 表示

具体步骤：

1. 利用时态边缘算子 $Teo(t,u)$ 与时间序列 $X=\langle x_1,x_2,\cdots,x_n\rangle$ 作卷积运算，计算时间序列中各点的边缘幅度。设边缘幅度序列为 $E(X)=\langle e_1,e_2,\cdots,e_n\rangle$；
2. iMax = 1; iMin=1; iFlag = 0;
3. SP={(x_1,1)}; //第一个序列点是边缘点
4. For (i=1; i<n+1; i++)
5. If $e_i > e_{iMax}$ then iMax = i ; If $e_i < e_{iMin}$ then iMin = i ;
6. if $e_i > e_{iMin}$ and i-iMin>=d and iFlag<>-1 then
   SP = SP + {(x_i,i)}; //序列点 x_i 加入边缘点集合 SP
   iMax = iMin = i; iFlag = -1;
7. if $e_i < e_{iMax}$ and i-iMax>=d and iFlag<>1 then
   SP = SP + {(x_i,i)}; //序列点 x_i 加入边缘点集合 SP
   iMax = iMin = i; iFlag = 1;
8. SP = SP + {(x_n,n)}; //最后一个序列点是边缘点
9. output $L(X)=\{L(x_{i_1},x_{i_2}),\cdots,L(x_{i_{K-1}},x_{i_K})\,|\,(x_{i_m},m)\in SP\}$；

定义 2-6 边缘点

设时间序列 $X=\langle x_1,x_2,\cdots,x_n\rangle$，那么有

(1) 序列点 x_1,x_n 分别是时间序列 X 的边缘点；

(2) 设 x_i 是边缘点，且是极大(小)边缘点，那么 x_j 是边缘点当且仅当 x_j 是极小(大)边缘点，并且 $j-i\geq d$；

其中d是正整数，表示最小模式长度。注意极大(小)边缘点不一定是边缘点，但是边缘点一定是极大或极小边缘点。

具体的边缘点计算和选择算法如算法 2-7 所示。

2.3.2 算法分析

TEOSegmentation 算法共需扫描序列两次，第一次扫描序列作卷积运算，第二次扫描序列得到边缘点，算法的时间复杂度为$O(n)$，其中n是时间序列的长度。经过对程序优化，可以在作卷积运算的同时选择边缘点，这样就只需要对时间序列扫描一次。TEOSegmentaion 算法的一个优点是可以直接用于时间序列的在线分段。当时间序列动态增长时，以前的边缘点结果集依然可以保留，只需对新的时间序列数据应用 TEOSegmentation 算法即可。

TEOSegmentaion 算法的一个与众不同的特点是采用了时态边缘算子来选择分段点。对于不同的应用领域，时间序列的数据特征有着明显的不同，通过设计不同的时态边缘算子，TEOSegmentaion 算法可以灵活地应用于各种不同的时间序列。

基于边缘算子分段的**PLR**表示其结构与基于重要点分段的**PLR**表示结构相同，也可以使用 2.2.2 中的介绍的符号化方法来进行符号化。

2.4 边缘算子分段与重要点分段的比较

在本节中，我们通过详尽的实验来对边缘算子分段和重要点分段进行比较，采用的数据集是 keogh 等人[24]提供的不同领域的 10 条时间序列，本文简称为 KData。各序列如表 2.1 所示。

表 2.1 KData 数据集描述

序列名称	序列长度	序列名称	序列长度
Burst	**9382**	**Ocean**	**4096**
Chaotic	**1800**	**Powerplant**	**2400**
Darwin	**1400**	**Speech**	**1021**
Earthquake	**4097**	**Tide**	**8746**
Leleccum	**4320**	**Sunspot**	**2899**

由于数据集中各序列来自不同领域，序列值相差很大，为了便于对比，在应用分段算法之前首先对时间序列做规范化处理，将序列值规范化到(0, 1)之间。

规范化公式如下：

$$norm(x_i)=\frac{x_i-\min(X)}{\max(X)-\min(X)} \qquad \text{(公式 2.4)}$$

对于算法性能的评价指标，我们主要考察两种 PLR 表示方法与原时间序列之间的拟合误差。由于两种算法的输入参数各自不同，为了公平起见，我们比较选择相同数量的分段点即相同的压缩率下，通过调节各自参数后得到的时间序列表示与原时间序列之间的拟合误差。拟合误差越小，表示算法性能越好。

定义 2-7 分段线性表示的拟合误差

设时间序列 $X=\langle x_1,x_2,\cdots,x_n\rangle$，通过线性分段算法得到 PLR 表示为 $L(X)=\{L(x_{i_1},x_{i_2}),L(x_{i_2},x_{i_3}),\cdots,L(x_{i_{K-1}},x_{i_K})\}$，其中 $L(\bullet,\bullet)$ 表示连接两点的直线段。将 $L(X)$ 经过线性插值后得到的时间序列记为 $X^c=\langle x_1^c,x_2^c,\cdots,x_n^c\rangle$，那么该 PLR 表示与原始时间序列之间的拟合误差定义为：

$$E=\sqrt{\sum_{i=1}^{n}(x_i-x_i^c)^2} \qquad \text{(公式 2.5)}$$

我们取压缩率为 80%来进行实验，实验结果如表 2.2 所示：

表 2.2 压缩率为 80%时两种 PLR 表示的拟合误差

算法 / 数据集	重要点分段	边缘算子分段
Burst	0.89	**0.16**
Chaotic	1.63	**1.22**
Darwin	**2.41**	2.45
Earthquake	**2.10**	2.17
Leleccum	0.88	**0.54**
Ocean	0.38	**0.19**
Powerplant	2.23	**1.03**
Speech	1.52	**1.42**
Tide	**2.29**	2.37
Sunspot	3.52	**2.68**

从上表可以看出，在 10 条时间序列的性能对比中，基于重要点的分段方法在其中的 3 条时间序列上拟合误差最小；基于边缘算子的分段方法在其中的 7

条时间序列上拟合误差最小，在另外 3 条序列中，其结果也与重要点分段方法的结果非常接近。实验结果表明：基于边缘算子的分段方法在所有的 10 条时间序列上的拟合误差都很小，具有很好的适应性。基于重要点分段方法也表现出了一定的竞争力。

2.5 小结

由于时间序列数据的海量和高维特性，直接在时间序列上的数据挖掘不但在储存和计算上要花费高昂代价而且准确性和可靠性也不能得到保证。通过对时间序列进行分段和符号化，有利于对时间序列进行进一步的分析处理。

通过重要点分段或边缘算子分段方法，找到时间序列中的特征点，再将这些特征点依次用线段连接起来，就可以得到时间序列的一个 **PLR**(分段线性表示)。进一步对它提取特征进行符号化，可以便于后续的分析研究工作。

第三章 时间序列特征模式挖掘

时间序列挖掘是数据挖掘在时间序列分析中一种典型运用。数据挖掘（Data Mining DM）是90年代中后期兴起的一门跨学科的综合研究领域，它集中计算机机器学习，统计学，数据库管理，数据仓库，可视化，并行计算，决策支持为一体，利用数据库，数据仓库技术存储和管理数据，利用机器学习和统计学方法分析数据，旨在发现大量复杂数据中蕴涵的有价值的知识和信息[33]。DM是在数据集中进行知识发现（Knowledge Discovery in Database KDD）的关键步骤。数据挖掘因其具有深刻的科学理论知识和巨大的商业前景，现在已经成为国际上数据库和信息决策领域最前沿、最热门的研究方向之一，并引起了学术界和工业界的广泛关注[34]。

时间序列是一种常见而又重要的数据类型，在海量的时间序列中发现其背后隐藏的知识对于我们分析时间序列变化规律，科学地做出决策具有重要的意义。因此在数据挖掘概念提出不久，就有不少研究人员把数据挖掘的思想运用到时间序列分析中来。

Szladow & Ziarko 等人利用 Rough 集理论通过一个移动的窗口将把带时标的数据经过计算转化成各种新的条件与决策属性标记[35]，然后利用一般意义上的挖掘方法进行挖掘。但是这种方法主要缺点是只有对落入窗口内的时间依赖性才被列入考察范围，这样造成挖掘结果不完全。类似的文献还有[36],[37][38]等。

与此同时，Mannila 等人[14]则将 Agrawal 等人[9]的关联规则发现算法的核心思想推广到时间序列，提出了事件序列中频繁事件的发现算法。这里，事件序列是一种离散的时间序列。

G. Das 等人[18]通过一个移动窗口将序列分割成若干子序列，利用聚类方法对这些子序列分类为特定的变化模式，然后按照关联规则的方式来发现频繁出现的特征变化模式。这种模式改变了以往是时态逻辑的形式，而是一种图形化表达方式。

此外，Fu 和 Chung 等人则利用自组织映射通过聚类来发现时间序列频繁出现的模式[39]。

Eamonn Keogh 等提出了一种从时间序列中发现异常模式的算法[21]。他将异常模式定义为出现频率远大于(或远小于)欺望值的模式。所用的方法称为 Tarzan 算法，其基本原理为：先使用一个时间窗口来提取分段的特征，通过聚类将时间序列转化为一个符号序列，然后利用后继树来计算各个模式的出现频率。

从上面的研究可看出，时间序列挖掘方法基本上分为两个过程：序列特征描述和挖掘算法设计。目前研究主要是采用了线段化的描述方式和基于传统的关联规则挖掘算法，如 Apriori 算法等。

在本章中，我们先介绍一下传统的时间序列特征模式挖掘算法，然后详细讨论我们的采用重要点或边缘算子分段，基于互关联后继树的挖掘算法。

3.1 传统的时间序列挖掘方法

为了后面的比较，在这部分中，我们简要地介绍一下[18]时间序列模式的挖掘算法。该算法的主要过程分为三步骤：利用移动时间窗口对序列进行分段；然后对各个子段进行聚类后用符号化描述；最后利用关联规则的挖掘思想对上述字符进行模式规则发现。

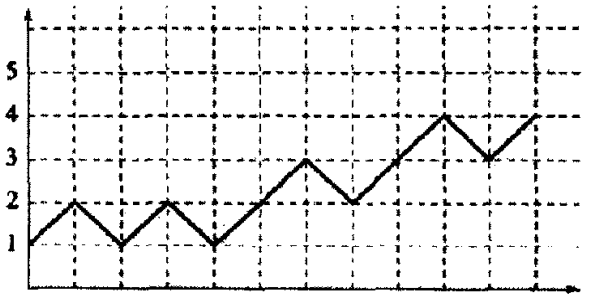
一． 序列分段

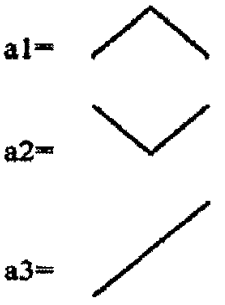
假设 s 是一时间序列，$s=(x_1, x_2, \ldots, x_n)$，$w$ 为一时间窗口，其长度为 w。在时间序列 s 上移动时间窗口 w，则可以得到一些长度为 w 的子序列段$(x_i, x_{i+1},\ldots, x_{i+w-1})$，也就说整个序列 s 分为了子序列 $s_1, s_2, \ldots, s_{n-w+1}$，其中 $s_i = (x_i, x_{i+1},\ldots, x_{i+w-1})$。记 $W(s) = \{s_i \mid i=1, 2, \ldots, n-w+1\}$ 。

二． 子段聚类

假设子序列 s_i 与 s_j 的距离为 $d(s_i, s_j)$。利用这种距离作为全部子序列的聚类依据对全部子序列进行聚类，把 $W(s)$分类成 k 类型 $C_1, C_2, \ldots, C_k$。对于每个类型，我们用一个符号 a_h 来描述，然后根据时间序列 s_i 中属于某个类 $C_{j(i)}$，就用该类所对应的符号 $a_{j(i)}$来替换。这样通过 k 个字符集把整个时间序列 s 转变为离散形式的字符序列 $D(s) = a_{j(1)}, a_{j(2)}, \ldots, a_{j(n-w+1)}$。

事实上每个字符 a 代表了一类序列的基本形状，这些基本形状图形构成我们希望挖掘的模式规则。图 3-1 给出了一些基本形状的例子。





原始序列=（1,2,1,2,1,2,3,2,3,4,3,4）
窗口大小=3
符号化序列=(a1,a2,a1,a2,a3,a1,a2,a3,a1,a2)

聚类后的基本图形

图 3-1 基于基本图形化的模式例子

至于序列之间的距离计算，算法采用基于 Euclidean 距离公式。即

$d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$，其中 $x = (x_1, x_2, ..., x_n)$，$y = (y_1, y_2, ..., y_n)$。

而关于聚类算法，[18]采用一种典型的贪心算法。任取 $W(s)$中某点 q 作为中心，然后计算任意其它点 p 与 q 的 Euclidean 距离 $d(p, q)$。如果 $d(p, q) < d$，则把 p 加到 q 所在的类中，否则创立新的一个类。这样不断下去，则到所有 $W(s)$中的子序列分类完毕为止。此外，同样也可以运用其他聚类算法实现上述符号化聚类[40][41]。

三. 模式规则发现

对上述符号化的时间序列采用一般意义上的序惯模式挖掘算法，如 Apriori 算法[9]来发现时间序列基本形状模式规则。这种模式规则可以定义如下

若基本图形 A 出现，那么在 T 时段内基本图形 B 也会出现，记为 $A \Rightarrow^T B$。

对于给定符号化序列 $D(s) = (a_1, a_2, ..., a_n)$，图形模式 A 的频繁度 $F(A)$为 A 在 $D(s)$中出现的次数，其相对频繁度 $F(A)/n$ 为。规则 $A \Rightarrow^T B$ 的信任度 $c(A \Rightarrow^T B)$ 是在 T 时段内 A 之后接着出现 B 的百分数，即

$$c(A \Rightarrow^T B) = \frac{F(A,B,T)}{F(A)}$$，其中 $F(A, B, T) = \{i \mid a_i = A \wedge B \subset \{a_{i+1}, ..., a_{i+T-1}\}\}$。

计算这样规则的频繁度与信任度比较简单，只要通过扫描整个序列就可以得出。下图 3-2 是算法挖掘出来的一图形模式：$A \Rightarrow^{T=12} B$

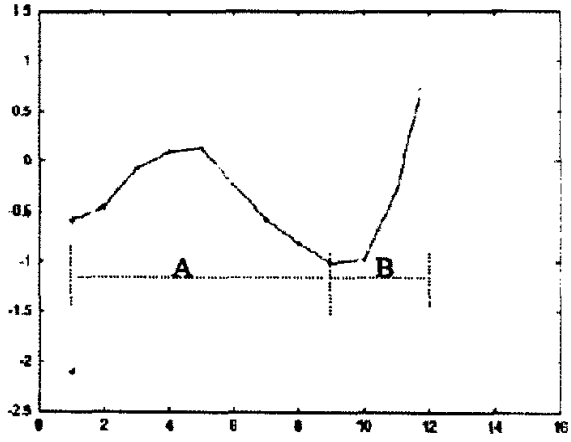


图 3-2 算法发现的频繁模式

方法讨论：该算法首先提出对序列进行符号化转变，通过序惯模式的挖掘思想来发现时间序列图形化的变化模式，这与以往利用 Rough 集理论进行时态模式的挖掘相比，在挖掘结果上更有感性认识，具有新颖性。

然而我们也应该看到该方法仍然一些缺陷：首先，时间序列的基本图形要受到时间窗口大小的影响，这种等长度的子序列划分很容易对时间序列重要特征的分割破坏；其次，其聚类方法的效果与效率也值得商榷，因为简单的欧氏距离并不能刻画时间序列的相似性；再次，基于 Apriori 算法的模式规则挖掘，由于其需要生成大量的候选模式，影响了其在海量时间序列挖掘效率[10]。最后，挖掘出来的模式虽然在感性上有比较好的认识，但是却难于语言描述，这对于模式在实际运用上造成了障碍。

因此，我们认为一个好的时间序列模式挖掘方法应该：首先，能够发现真正代表时间序列变化特征的变化模式，而且这种模式不仅能有图形化的感性认识，而且也利于在实际中描述运用；第二，挖掘算法还应该是高效、简单的，可以适应于海量的时间序列挖掘。

3.2 互关联后继树简介

互关联后继树（**Inter-Relevant Successive Trees IRST**）是胡运发教授根据序列字符的有序性和冗余性而提出的一种新型的海量全文存储、索引模型[23]。它的一些基本概念如下。

定义 3-1 (后继) 对任意文本符号 a_1, a_2，在字符串 a_1a_2 中，a_1 称为 a_2 的前驱；a_2 称为 a_1 的后继。

注记：在一个全文字符串 $a_1, a_2, \cdots, a_N$ 中(为了处理方便，我们在全文的最后字符后面添加了结束标记符*)，若出现了 m 个相同的字符，不妨记为 a，那么 a 的后继共有 m 个(a_N 的后继为*)，记为($a[k], k=1, \cdots, m$)，$a[k]$表示 a 的第 k 个后继。

定义 3-2 (后继表达式与后继树) 设全文是由一字符串 $a_1, a_2, \cdots, a_N$ 组成的，若其中 $a_{i_1}, a_{i_2}, \ldots, a_{i_m}$ 为相同的字符，不妨记作 $a_{i_1}, a_{i_1+1}, a_{i_2+1}, \ldots, a_{i_m+1}$ 分别是其后继，而 $a_{i_1+1}, a_{i_2+1}, \ldots, a_{i_m+1}$ 的后继分别是 $a_{i_1+1}[tag_1], a_{i_2+1}[tag_2], \ldots, a_{i_m+1}[tag_m]$，那么我们称 $a_{i_1}((a_{i_1+1}, tag_1), (a_{i_2+1}, tag_2), \ldots, (a_{i_m+1}, tag_m))$ 为 a_{i_1} 的后继表达式，其可以用一棵后继树来描述此表达式，如图 3-3 所示，

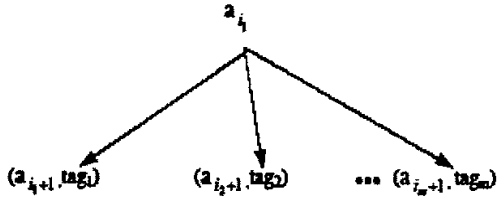


图 3-3 关联后继树

其中 a_{i_j} 为树根，分支叶节点由 (a_{i_j+1}, tag_j) ($j=1, \ldots, m$)组成，其表示 a_{i_j+1} 是 a_{i_j} 的一个后继，标记 tag_j 是指 a_{i_j+1} 的后继将出现以 a_{i_j+1} 为根的后继树中的第 tag_j 个分支叶节点上。

定义 3-3 (互关联后继树模型) 若全文 $a_1, a_2, \cdots, a_N$ (以*为结束标记)中共有 k 个不同的字符，这 k 个字符对应于 k 棵不同的后继树，而这些后继树中的每个叶节点都对应于另外一棵后继树的根节点。换而言之，一个全文所对应的所有后继树是相互关联的，这种关联关系反映了字符在全文中出现的位置前后关系。我们称一个全文对应的所有后继树为全文的**互关联后继树**,记为 **IRST**。

例 2-1 设全文“abcabaabc”，取“*”为串的结束标志符。其中共有 3 个不同的字符：a、b 和 c，它们对应的后继表达式分别为：a (b, b, a, b)、b (c, a, c)和 c (a, *)，相应的后继树如图 2-7 所示。这三棵后继树构成全文“abcabaabc”的互关联后继树。

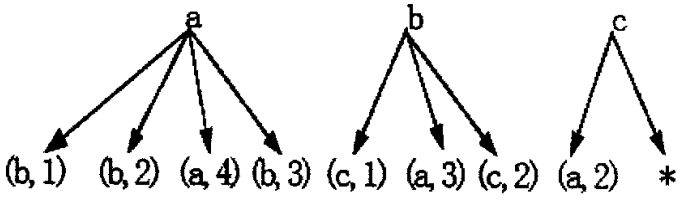


图 3-4 abcabaabc*的互关联后继树(IRST)

进一步，从图 3-4 中的第一棵树的第一个分支 a→(b, 1)开始，按照叶节点的后继符 b 及其后继标记 tag=1 遍历后继树 b 的第 1 个分支，如此下去将遍历整个

IRST，那么遍历路径上的根项所组成的有序字符便还原初始的全文序列。

由此并结合定义 3-2, 3-3，我们不难得出下面关于 IRST 等价的描述性定理。

定理 3-1. IRST 是全文的一种等价的描述模型，即对任意一全文都能构造其 **IRST**，同时对于 **IRST** 也能还原其对应的原文[23]。

3.3 基于互关联后继树的时间序列挖掘算法

在本节中，我们介绍一下[22]中提出的一种基于互关联后继树的时间序列挖掘算法，该算法先对始时间序列进行重要点分段和符号化，然后对得到的符号序列建立互关联后继树模型，最后在这个互关联后继树模型中进行特征模式挖掘。

3.3.1 时间序列的互关联后继树模型

由于经斜率符号化的时间序列与全文字符序列具有相似的性质，因此可以用前面的 IRST 模型表示符号化的时间序列。不过，由于时间序列分段后的每一个模式由长度和斜率符号两个特征来表示，因此经过斜率符号化的时间序列的 IRST 在细节上仍然和纯字符串的 IRST 有少许差异，为此我们时间序列的 **IRST** 模型为 **SIRST** 模型（**Sequence IRST**）。图 3-5SIRST 结构。

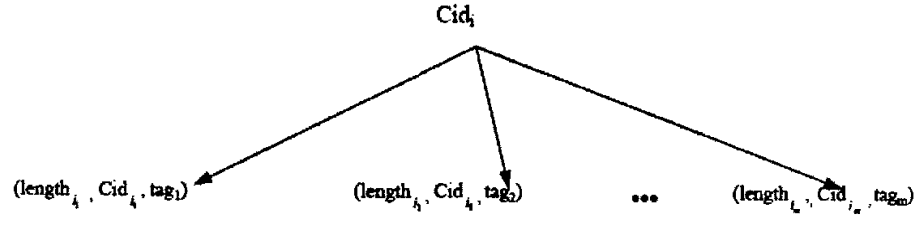


图 3-5 SIRST 的结构

在图 3-5 中，Cid$i$ 与 Cidi_j 为线段斜率的变化类型符，lengthi_j 是线段长度(j=1,...m)，分支 Cid$i$→(lengthi_j, Cidi_j,tag$_j$)表示线段(lengthi_j, Cid$i$)的后继线段是以符号 Cidi_j 为其斜率，其长度在以 Cidi_j 根项的后继树中的第 tag$_j$ 分支节点上。

例 3-1 对于经斜率符号化后的时间序列{l_1A_1,l_2A_2,l_3A_3,l_4A_2,l_1A_1,l_2A_2,l_5A_3,l_4A_2, l_3A_3, l_6A_2}，其生成的 SIRST 模型如图 3-6 所示：

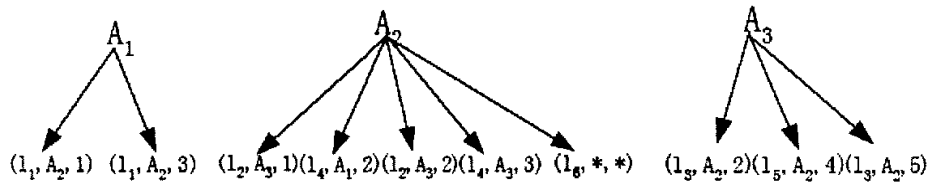


图 3-6 SIRST 模型

下面我们给出其构造 SIRST 的具体算法。

算法 3-1： **Procedure Create_SIRST (S^C, SIRST)**

输入： 由线段 $l_iA_{j_i}$ (i=1,..,n, j=1,…,k)组成的序列 S^C（假定以*为结束符）；

输出： 序列 S^C 的互关联后继树 **SIRST**。

程序：

1) **SIRST** ={}；
2) read(S^C[i])；*//S^C[i]表示 S^C 中的第 i 个线段，A_{j_i}，l_i 分别是其斜率符号与长度。*
3) if exit T_k∈**SIRST** and T_k.rootname=A_{j_i} then append(T_k, A_{j_i} →(l_i,*,*)) *//向 T_k 添加分支 A_{j_i} →(l_i,*,*)。*

   else create_tree(T_k, A_{j_i} →(l_i, *,*))； *//创建仅含分支 A_{j_i} →(l_i, *,*)的后继树 T_k*
4) if i≠1 then modify (($A_{j_{i-1}}$ →(l_{i-1}, *,*),$A_{j_{i-1}}$ →(l_{i-1}, A_{j_i} , tag_{k_i})； *//对处理第 S^C[i-1]个线段时所添加的分支叶节点修改，使线段 S^C[i]成为其后继，tag_{k_i} 是 S^C[i]在 T_k 中所处的分支序号。*
5) if i≠|S^C| then { i=i+1, goto 2 }。 *//|S^C|是序列 S^C 的长度。*

算法 2-5 仅需一次扫描整个序列，其时间复杂度为 O(|S^C|)。从所构造的 **SIRST** 中同样可以还原对应的符号序列，为此可得出类似定理 3-1 的 **SIRST** 等价描述性定理。

定理 3-2 SIRST 是符号序列等价的描述模型。

证明：根据 **SIRST** 的结构与定理 3-1 可以得证。

由定理 3-2 我们不难得出下面两个性质。

性质 3-1. 若线段 $l_iA_{j_i}$ 在序列 S^C 中出现 f 次，则在 **SIRST** 中以 A_{j_i} 为树根的后继树中共有 f 个 length 为 l_i 的分支叶节点；反之亦然。

证明： 根据 SIRST 的结构可知，在以 A_{j_i} 为根项的后继树中，其所有分支节点包含了全部以 A_{j_i} 为相对斜率的线段，并且其长度分别存放在该树的后继树分之上。又由定理 3-2 可得性质 3-1 成立。

性质 3-2. 若相邻线段($l_iA_{j_i}l_{i+1}A_{j_{i+1}}$)序列 S^C 出现了 f 次，则在以 A_{j_i} 为根的后继树中包含 $A_{j_{i+1}}$ 的叶节点至少有 f 个；假设 tag_t (t=1,..s, s≥f)是这些节点中对应的后继标号，则在以 $A_{j_{i+1}}$ 为根项的后继树，在其分支 tag_t 中共有 f 个 length 为 l_{i+1}

的分支叶节点；反之亦然。

证明：根据 SIRST 的结构与定理 3-2 可知，在以 A_{j_i} 为根项的后继树中，其所有分支节点包含了全部的以 A_{j_i} 为前驱的后继线段，这些后继线段的斜率符号在以 A_{j_i} 为根项的后继树分支节点中，而其长度在以 $A_{j_{i+1}}$ 为根项的后继树分之上，又由性质 3-1 可得性质 3-2 成立。

3.3.2 基于互关联后继树的发现算法

频繁模式的挖掘任务就是在线段化的时间序列 S^C 中，发现出现频度大于 f_{min} 的各种长度的变化模式。而 TSIRST 的性质能有效的帮助频繁模式的挖掘过程：1)通过性质 3-1 可以发现频率大于 f_{min} 的频繁模式；2)通过性质 3-2 可以从已发现频繁模式的叶结点的 Cid, tag 中发现更长的频繁模式。因此，基于 **SIRST** 的挖掘过程主要分为两步：

1) 从**SIRST**后继树的根开始扫描各叶子结点，对于相同length计算其出现次数，若次数大于f_{min}，则以length为长度，以根项为代表斜率变化的特征模式是频繁的。

算法 3-2 Procedure Frequent_Pattern (SIRST,f_{min}, Result);
输入: SIRST, f_{min}
输出: Result
程序:
(1) Result={};
(2) For i:=1 to k do *//假设 SIRST 有 k 个后继树，依次发现以树 T_i 根项开始的频繁模式*
(a). Search_branch=T_i.branch; *// T_i.branch 是 T_i 的全部分支;*
(b). result[i]={};
(c). (result[i],FP_leafnode)←DiscoverFP(T_i,f_{min},Search_branch); *// 利用性质1,在 T_i 的Search_branch 中对不同length 查找至少出现 f_{min} 次的分支叶节点，并返回相应长度的频繁模式到result[i] 中.*
(d). if FP_leafnode≠∅ then get each type Cid Cid_j and its all tags tagset from FP_leafnode *//利用性质2,在已发现的叶节点中,根据其最新后继斜率符及其所对应全部的后继标记，继续发现更长的频繁模式.*
(e). T_i=T_{Cid_j}; Search_branch= tagset
(f). goto (c);
(3) Result=∪result[i];
(4) Return(Result)

2) 在上述发现的频繁模式的叶结点中，依次选取每一种后继斜率分类符Cid以及其所有对应的后继标记集tags，然后在以Cid为根项的后继树中，以其对应tags分支节点集中，继续发现更长的频繁模式。

挖掘过程的具体算法如算法3-2所示，可以看出，算法最大的特点是整个过程无需生成候选模式，把挖掘过程转变成一种查找过程，这大大可提高挖掘的效率。其次，算法通过后继斜率符与下标进行查找的过程也充分体现了[9]关于频繁模式性质，即“任何频繁模式的子模式必定也是频繁的”。

定理3-3. 算法3-2能发现时间序列S^C中全部的频繁模式。

证明：由定理3-2可知，**SIRST**包含S^C中全部线段模式及其顺序位置信息，而没有任何线段模式的遗漏或冗余。而由性质3-1可知，算法3-2发现的模式必定是频繁的，性质3-2则保证算法3-2能发现各种长度的模式。因此算法3-2能发现S^C中全部的关联模式。

例3-2 下面我们结合例3-1（图3-6），具体阐述其频繁模式的发现过程，我们假设fmin=2；

1) 发现变化特征从A1开始的频繁模式

从后继树T_{A1}的根出发，遍历其所有叶结点，搜索length相同且相同数目大于2的叶节点。结果发现T_{A1}的第1，2分支的长度为l_1且相同的分支数≥2，因此(l_1A_1)是一个长度为1的频繁模式；从第1，2分支叶节点中获得的下一个斜率符Cid={A_2}与标号tag={1, 3}出发，发现树T_{A2}的第1, 3分支叶节点的length均为l_2，因此可以获得长度为2的频繁模式($l_1A_1l_2A_2$)；接着从T_{A2}第1，3分支的叶结点获得的再下一个符号Cid={A_3}与标号tag={1, 2}出发，没有发在其第1, 2个分支中有相同length的叶节点，则从A1开始频繁模式的发现过程结束；

2） 同样按照1）的方法，可发现从A2, A3开始的频繁模式有三个：(l_2A_2), (l_4A_2), (l_3A_3)。

3.3.3 算法讨论

基于互关联后继树的时间序列特征模式发现算法，由于利用了互关联后继树索引模型，挖掘时的效率很高[22]；而且采用长度和相对斜率表征的模式有很明显的含义，便于特征模式在实际中的应用。但互关联后继树模型本是一种应用在文本索引中的模型，将其应用到模式挖掘中，存在着一个重大缺陷：即它只能挖掘出紧密衔接的特征模式。例如将其应用在股票市场中，它可以方便的挖掘出形如“股价经历了10天大涨后，紧接着小跌15天”的规则，但假设股市中出现“股价在经历了10天大涨后，小涨1天，然后小跌15天”的情形，则该方法会认为此情形不属于前面的规则的实例。实际上这1天的小涨可以看作是一种噪音。另外，

该方法无法挖掘出形如“股价在经历了10天的大涨后，在30天内，会出现15天的小跃”的规则。这是由于互关联后继树本是应用于文本挖掘中的索引方法，它要求被索引的模式之间紧密衔接，如一个单词的后继是它的后一个单词，不考虑它之后的第二个单词。因此用这种发现方法建立的互关联后继树索引模型无法挖掘出第二类规则，而在时间序列分析中，往往第二类规则更有实际意义。

在下一节中，我们将对这种方法进行改进，加入一个时间窗口，以便从时间序列中挖掘出更多的规则。

3.4 带时间窗口的互关联后继树挖掘算法

在本节中， 我们在3.3中描述的互关联后继树挖掘算法的基础上，加入一个时间窗口，对互关联后继树模型的结构及挖掘算法都做出相应的改进，得出了一种新的带时间窗口的互关联后继树挖掘算法。实验表明，这种算法可以挖掘出更多、更广泛的特征模式。

3.4.1 带时间窗口的互关联后继树索引模型

在3.3.3的算法讨论中，我们提到互关联后继树挖掘算法的一个缺陷：无法挖掘出不连续(紧密衔接)的特征模式，下面我们更加详细的讨论这个问题。

考虑如图 3-7 中两个从股票价格变动图中取出的两个波形片段，它们分段后的线段表示也列在图中，其符号化结果分别为$\{6A_1,5A_3,6A_2\}$和$\{6A_1,5A_4,6A_2\}$。可以看出，这两个波形非常相似，但由于它们中间部分的跃幅分别为 A_3 和 A_4，因此如果用[22]中的互关联后继树挖掘算法进行挖掘，会认为这两个波形不属于同一个特征模式。

图 3-7 两个波形片段及其线段表示

这是由于[22]中的规则格式为：

如果 A 发生，则 B 紧接着发生。

这种规则的定义过于死板，将会忽略很多的潜在规则；另外如果时间序列中存在很多噪音，如股市中股价在短期内的剧烈变动，将会对挖掘结果产生很大的影响。因此，在本文中，将规则的格式扩展为：

如果 A 发生，则在时间 T 内 B 发生[18]。

则图 3-7 中的两个波形可认为属于同一特征模式“6 天的 A_1 后，在 10 天内，会出现 6 天的 A_2”。为找出这种格式的规则，我们采用时间窗口扩展了后继的概念，规定一条线段的后继并不只包括它的相邻线段，而是包括所有的在它之后的一个时间窗口 T 内出现的所有线段。并据此对 SIRST 的结构进行改进，改进后的 SIRST 结构如图 3-8 所示：

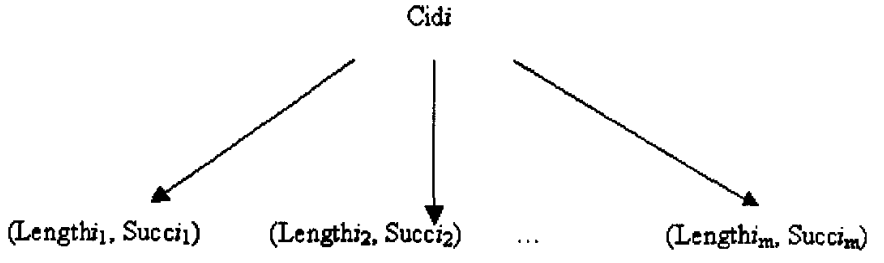


图 3-8 TSIRST 的结构

在图 3-8 中 $Succi_j$ 表示在时间窗口 T 内的后继列表，是由 Cid 和 tag 配对组成的数组，其中，Cid、Length 和 tag 所代表的意义与 SIRST 中相同。我们称这种带时间窗口的互关联后继树模型为 **TSIRST**(Sequence IRST with Time Window)。

下面给出 TSIRST 的详细构造算法：

算法 3-3: Procedure Create_TSIRST(S^C,T, TSIRST)
输入：由线段 $l_iA_{j_i}$ 组成的序列 S^C(假定以*为结束符)和时间窗口大小 T；
输出：序列 S^C 的带时间窗口的互关联后继树模型 TSIRST。
程序：

1) TSIRST = {};
2) read(S^C[i]);*//$S^C[i]$表示S^C中的第i个线段，A_{j_i}与l_i分别是其斜率符号和长度。*
3) if exist $T_k \in$ TSIRST and T_k.rootname == A_{j_i} then append (T_k, A_{j_i} → (l_i,[]))*//向T_k添加分支A_{j_i} → (l_i,[])。*
   else create_tree(T_k, A_{j_i} →(l_i,[])); *//创建仅含分支A_{j_i} →(l_i,[])的后继树T_k。*
4) if i≠1 then for every A_{j_m}→(l_m, $Succ_m$)(m<i and S^C[i] is included in the successive time window of S^C[m]) add (A_{j_i}, tagk_i) to $Succ_m$; *//对所有后继时间窗口T内包含$S^C[i]$的线段所对应的叶子结点进行修改，把(A_{j_i}, tagk_i)添加到其后继数组中，其中tagk_i为$S^C[i]$在T_k中所处的分支序号。*
5) if i≠$|S^C|$ then {i = i+1, goto 2}。*//$|S^C|$是序列S^C的长度。*

算法 2 只需扫描序列一次，时间复杂度为 $O(\frac{T}{\bar{l}}|S^C|)$，其中 $\bar{l}$ 为线段序列的平均长度。

不难证明，带时间窗口的互关联后继树与普通的互关联后继树类似，有以下两个性质：

性质 3-3：若线段 $l_iA_{j_i}$ 在序列 S^C 中出现 f 次，则在 TSIRST 中以 A_{j_i} 为树根的后继树中共有 f 个 length 为 l_i 的分支叶节点；反之亦然。

性质 3-4：若特征模式($l_iA_{j_i}$, $l_{i+1}A_{j_{i+1}}$)在序列 S^C 中出现 f 次，则在以 A_{j_i} 为根的后继树中，length 为 l_i 的叶节点至少有 f 个;假设 tag_t(t = 1,..,s, s≥f)是这些叶结点的 succ 数组中 Cid 为 $A_{j_{i+1}}$ 对应的后继标号，则在以 $A_{j_{i+1}}$ 为根的后继树中，在其分支 tag_t 中共有 f 个 length 为 l_{i+1} 的分支结点。反之亦然。

3.4.2 挖掘算法

与 SIRST 相似，基于 TSIRST 的挖掘过程主要分为两步：

1) 从TSIRST后继树的根开始扫描各叶子结点，对于相同length计算其出现次数，若次数大于f_{min}，则以length为长度，以根项为代表斜率变化的特征模式是频繁的。
2) 发现的频繁模式的叶结点中，根据其后继数组succ依次选取每一种后继斜率分类符Cid及其对应的后继标记集tags，然后在以Cid为根项的后继树中，以其对应tags分支结点集中，继续发现更长的频繁模式。

需要注意的是，SIRST 中每个结点只有一个后继，而 TSIRST 中每个结点的后继有多个，是一个后继数组，需要进行特别的处理，下面给出挖掘过程的具体算法：

算法 3-4 Procedure Frequent_Pattern(TSIRST, f_{min}, Result):
输入：TSIRST, f_{min}
输出：Result
程序：

1) Result = {};
2) For i:=1 to n do *//假设 TSIRST 有 n 棵后继树，依次发现从树 T_i 根项开始的频繁模式。*
   (a) Search_branch=T_i.branch;
   (b) result[i]={};
   (c) (result[i], FP_leafnode) ← DiscoverFP(Ti, f_{min}, Search_branch); *// 在 Ti 的 Search_branch 中对不同 length 查找至少出现 f_{min} 次的分支叶结点，并返回相应长度的频繁模式到 result[i] 中。*
   (d) if FP_leafnode≠Φ then get each type Cid **Cid_j** and its all tags **tagset** from FP_leafNode's succ list. *//在已发现的叶结点中，根据其后继数组中的每种后继斜率符及其所对应的全部后继标记，继续发现更长的频繁模式*
   (e) Ti = Cidj; Search_branch = tagset;
   (f) goto(c);
3) Result=∪result[i];
4) return Result;

算法的整体思路与算法3-2相似，需要注意的是时间窗口的大小不能选得太大，否则如果分段所得的**PLR**的线段很短，则每个结点的后继数组中元素太多，会导致算法的时间复杂度呈指数级增长。实验证明，对股票每日收盘价时间序列，时间窗口取在5~10天为宜。另外需要对原始时间序列进行一些预处理，如傅利叶变换、移动平均等，使曲线更平滑，达到一定的除噪效果，这样可以防止产生很多小的分段，影响挖掘效率。实验表明，时间窗口大小为10时，在对3000个点的时间序列进行傅利叶变换后，用算法3-2和算法3-3进行挖掘需要的时间在一个数量级，均只需不到200ms。

3.4.3 实验

在股票交易市场中，利用数据挖掘对大量的历史交易数据进行分析、处理，揭示隐含在大量数据中的有价值的序列变化模式，可以对投资者的投资起到辅助、指导作用。我们用 TSIRST 方法对我国的股市技术分析作了试探性的实验。实验数据取自沪市 1990-12-19~2004-8-2 的数百支股票的日收盘价。实验在 Athlon 2500+, 512M 内存，Windows XP professional 环境下进行。我们先对数据用傅利

叶变换进行预处理，然后取 R=1.005 作为重要点参数对序列进行分段，然后提取长度和相对斜率两个特征进行符号化。对于分段后的线段序列，我们分别采用文献[22]中的方法和本文中的方法进行频繁模式挖掘。

图 3-9 用 TSIRST 挖掘出的模式

图 3-9 给出了用 TSIRST 挖掘出几个模式，可以看出，模式中的线段之间不都是紧密衔接的，这是由于时间窗口的存在，一条线段的后继是它之后的一个时间窗口内的所有线段。由于模式是用长度和相对斜率来描述，因此挖掘出的模式有明确的意义，容易解释。例如图 3-9 中的第一个特征模式可以解释为“股价在经过连续 3 天的小幅度下跌后，在 10 天内，将出现连续 5 天的小幅度下跌”。

实验还考察了在不同的频繁阈值下，用两种方法挖掘出的频繁模式数量及频繁模式中规则(即线段数大于 1 的特征模式)所占的比重。图 3-10 给出了时间窗口大小为 10 时，分别采用 TSIRST 和[22]中 SIRST 挖掘出的模式的数量的比较，可以看出，TSIRST 可以挖掘出更多更有实际意义的频繁模式，并且在频繁阈值 f_{min} 较小时，TSIRST 挖掘出的特征模式数是文献[22]中的方法挖掘出的特征模式数的 2 倍。图 3-11 给出了时间窗口大小为 10 时，TSIRST 挖掘出的频繁模式中规则所占的比重与 SIRST 的比较。可以看出，TSIRST 挖掘出的频繁模式中

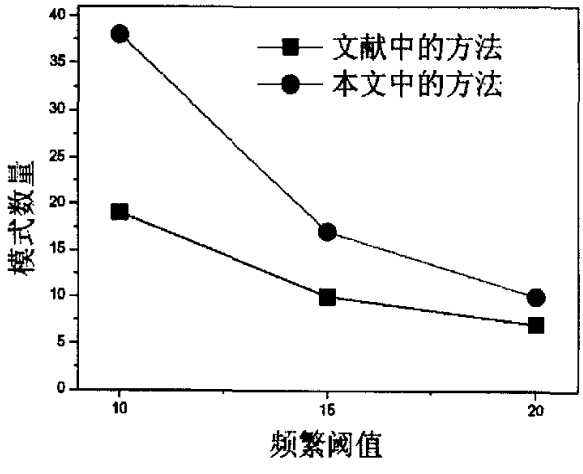


图 3-10 在不同的频繁阈值下两种方法挖掘出的模式数

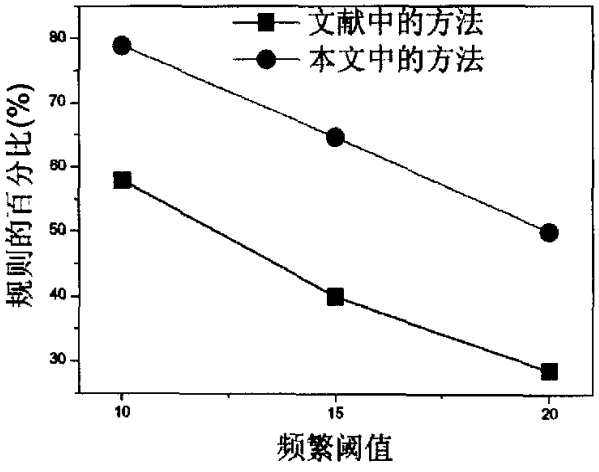


图 3-11 在不同频繁阈值下两种方法挖掘出的模式中规则的百分比

规则所占的比重远高于 SIRST。这说明利用 TSIRST 方法可以挖掘出更多的规则，而规则与普通的特征模式相比，在实际应用中有着更大的价值。股票市场的用户可以根据挖掘出的规则和股价前一段时间的变动，预计股价在之后的一段时期将出现的变动，这样可以起到指导投资者投资的作用。

3.5 小结

如何在时间序列数据库中高效地挖掘出有价值的、能方便地在实际中得以应用的频繁模式(或特征模式)是一项具有实际意义的重要课题，也是时间序列分析的一个重要方法。

传统的时间序列特征模式挖掘算法多是基于 Apriori 算法，需要产生大量的侯选模式，效率不佳。而且挖掘出的特征模式没有明显的意义，不利于在实践中使用。基于互关联后继树的特征模式挖掘算法没有以上两个缺点，但它只能挖掘出紧密衔接的模式，忽略了大量潜在的规则。

本文在基于互关联后继树的特征模式挖掘算法的基础上，加入时间窗口的概念，对互关联后继树的结构及挖掘算法做出了改进，从而可以挖掘出更多、更广泛、更具有实际意义的特征模式。实验证明，带时间窗口的互关联后继树挖掘算法可以发现更多的规则。

第四章 原型系统

根据第二章介绍的分段算法和第三章介绍的时间序列特征模式挖掘算法，我们用 java 实现了一个时间序列特征模式挖掘系统的原型，以方便进行实验。所用的开发工具为 Eclipse 2.1.2，JDK 版本为 Sun 公司的 j2sdk1.4.2_03，操作系统为 Windows XP。

图 4-1 是时间序列特征模式挖掘的流程图。

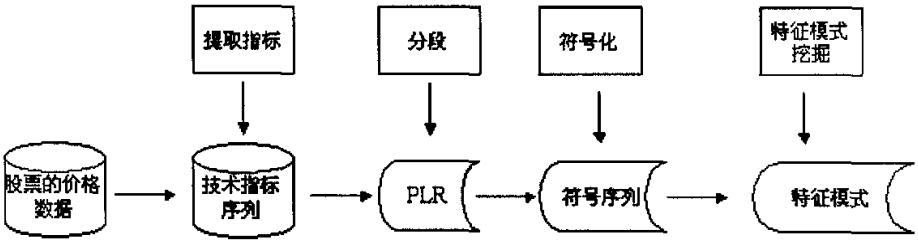


图 4-1 时间序列特征模式挖掘流程图

实验采用的数据集是沪市 1990-12-19~2004-8-2 的数百支股票的日收盘价，从图 4-1 中可以看出，时间序列特征模式挖掘的主要步骤包括：将原始数据集经过预处理转化为时间序列；分段得到时间序列的 **PLR** 表示；符号化得到一个符号序列；利用互关联后继树挖掘时间序列中的特征模式。

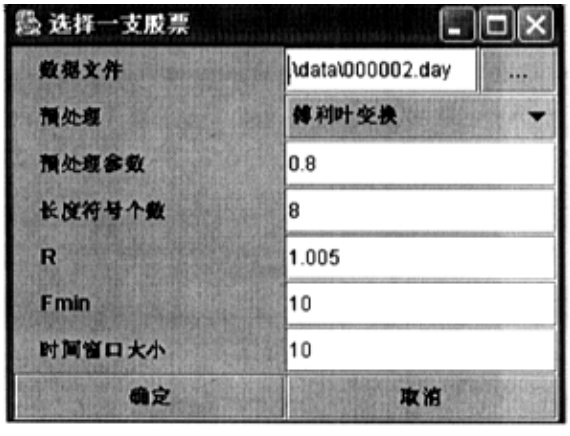


图 4-2 参数设置界面

图 4-2 是原型系统中的挖掘参数设置界面，需要设置的参数主要有：数据来源，预处理方法及参数，长度符号的个数，重要点分段的选取参数 R，特征模式的频繁阈值 Fmin 以及时间窗口的大小。

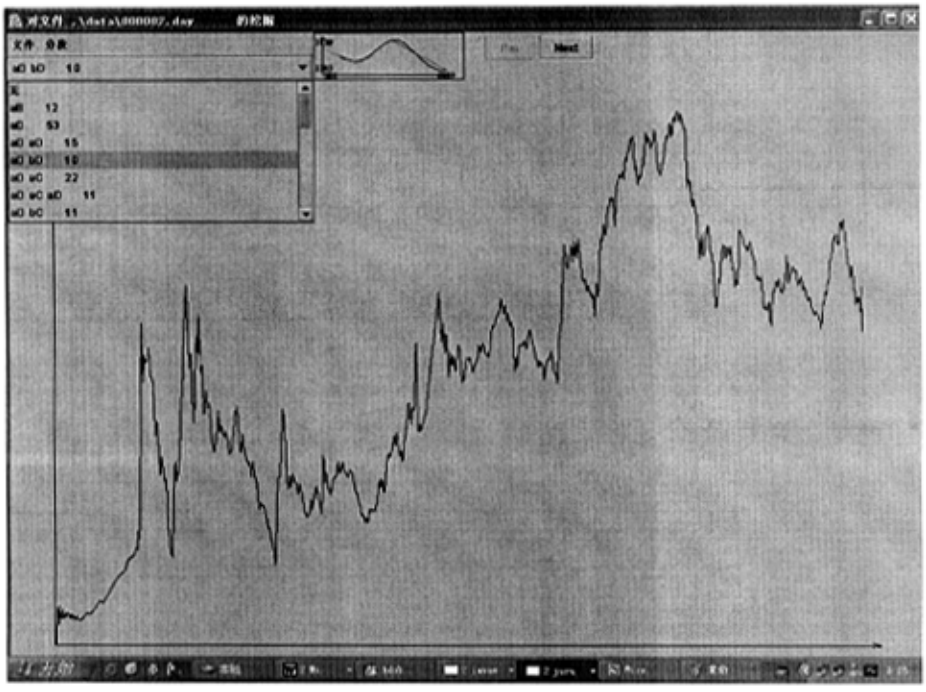


图 4-3 时间序列特征模式挖掘的主界面

图 4-3 是原型系统时间序列特征模式挖掘的主界面。占图片主体的黑色大座标图是时间序列的曲线图。图片的左上角有一个下拉框，称为模式下拉框，列出了挖掘出的所有特征模式及其频度。例如图中选中的 aD bD 10 表示特征模式 aDbD，出现频度为 10 次。模式下拉框右侧有一小的座标图，称为模式图，它的作用是绘出选中模式的放大图，黑色的曲线为原始曲线，红色的线段是线段表示。其座标上的值均为该模式在原始时间序列中出现的具体位置。模式图右侧的两个按键 Pre 和 Next 的作用是依次浏览选中模式在时间序列中的每一次出现，并且会在原始时间序列曲线图中用红色的线段标出，如图 4-3 所示。

图 4-4 是原型系统中的分段拟合效果图，方便实验时直观的判断分段效果。该图是一张重要点分段的拟合效果图，黑色的曲线为原始时间序列，红色的折线为分段拟合曲线。

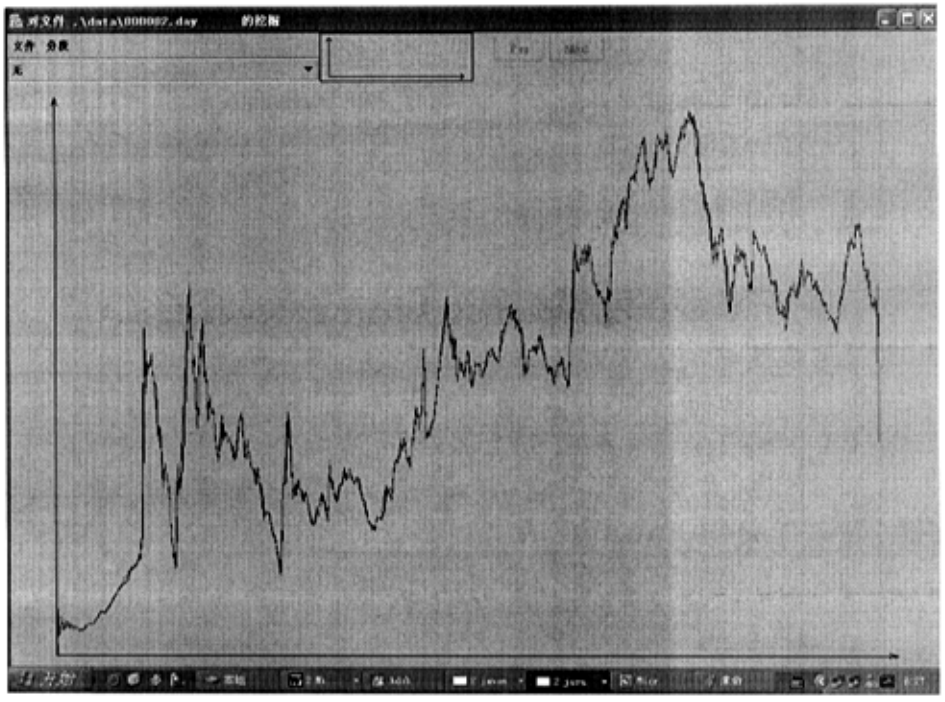


图 4-4 分段拟合效果图

第五章 结论

时间序列是一种重要的数据对象，在经济、气象等许多领域都大量存在，对这些数据进行分析，可揭示事物变化、发展规律，为科学决策提供依据。

分段是时间序列研究中的一个重要领域，对时间序列进行分段，得到时间序列的高级表示，可以减少存储空间，便于进一步对时间序列进行分析。传统的时间序列分段方法中，移动窗口分段最简单，最直观，计算量也较小，但是它分段拟合的准确性比较差，因为其只关注了序列局部的最小误差，而忽略了序列整体的变化特征；而自顶向下与自底向上分段效果较好，但计算量大。并且用这些方法得到的模式有一个共同的缺点：不易解释。

重要点分段方法的思想是找出序列中的“重要点”，即局部极大、极小点，将这些点依次按顺序用线段连接进来，即可得到时间序列的一个 **PLR**(分段线性表示)。算法效率高，并且每个分段都有明确的表征意义。提出了一种边缘算子分段方法，思路与重要点分段相似，只不过不是通过“重要点”，而是通过“边缘点”来分段。借鉴了数字图象领域中边缘算子的概念，利用边缘算子找出时间序列中的变化点。实验表明，边缘算子分段在大部分数据集中的拟合误差均小于重要点分段。文中采用统计的方法，将时间序列的 **PLR** 逐段提取长度和相对斜率两个特征进行符号化，在这基础上进行了特征模式挖掘的研究。

利用数据挖掘的方法发现时间序列中的特征模式，在实际应用中有着重要的意义。传统的时间序列特征模式挖掘算法多是基于 Apriori 算法，需要产生大量的侯选模式，效率不佳。而且挖掘出的特征模式没有明显的意义，不利于在实践中使用。基于互关联后继树的特征模式挖掘算法没有以上两个缺点，但它只能挖掘出紧密衔接的模式，忽略了大量潜在的规则。

本文在基于互关联后继树的特征模式挖掘算法的基础上，加入时间窗口的概念，对互关联后继树的结构及挖掘算法做出了改进，从而可以挖掘出更多、更广泛、更具有实际意义的特征模式。实验证明，带时间窗口的互关联后继树挖掘算法可以发现更多的规则。

我们今后的工作是在该算法的基础上进一步研究时间序列特征模式的在线挖掘，并尝试将这种新的互关联后继树模型应用的时间序列研究的其他领域，如相似性查询、异常检测等。

参考文献

[1]. 安鸿志，陈兆国，杜金观，潘一民，时间序列的分析与应用[M].科学出版社，1983年。

[2]. E. R. Tufte. The Visual Display of Quantitative Information[M]. Graphics Press, Cheshire, Connecticut, 1983.

[3]. Hunkhouser, H.G, Historical Development of the Graphical Representation of Statistical Data[M]. Osiris. 3(1): 269-405. Reprinted Brugge, Belgium; St. Catherine Press 1937.

[4]. George E.P.Box, Gwilym M.Jeakins, Gregory C.Reinsel, Times Series Analysis:Forecasting an d Control(Third Edition)(中译本，顾岚译)[M]，中国统计出版社，1997年。

[5]. U.M.Fayyad, G.Piatetsky-Shapiro and P.Smyth, From data mining to knowledge discovery: An Overview[J]. In U.M,Fayyad, G.Piatesky-Shapiro, et al eds, Advances in Knowledge Discovery and Data Mining, pp.1-34, AAAI/MIT Press, Menlo Park, Ca,1996.

[6]. J. Han and M. Kamber. Data mining: concepts & techniques[M]. Morgan Kaufmann Publishers, 2001.

[7]. 陈文伟，邓苏，张维明. 数据开采与知识发现综述[J]. 计算机世界专题综述，1997.6.30。

[8]. Julian Kulkarn, Richark King. Business Intelligence Systems and Data Mining[C]. ASAS Institute White Paper. 1996.

[9]. R. Agrawal, T. Imielinski, A. Swami. Mining association rules sets of items in large databases[C]. Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD'93). Washington, DC: ACM, 1993.

[10]. Mannila H, Toivonen H, Inkeri Verkamo A. Efficient algorithms for discovery association rules[C]. In Proceedings of AAAI Workshop on Knowledge Discovery in Database July 1994. 181-192.

[11]. Hannu Toivonen, Mika Klemettinen ,Pirjo Ronkaine et al .Pruning and grouping disovered association Rules[C] . In:Mlnet Workshop on Statistics ,Machine Learning and Discovery in Database. Heraklion, Crete, April 1995.

[12]. Agrawal R, Srikant R. Fast Discovery of Association Rules[C].In Fayyad.II 1996.

[13]. Agrawal R, Srikant. Mining Sequential Patterns[C]. In Proc.95 Int'l Conf Data Engineering, Taibei, Taiwan, March, 5,1995.

[14]. H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering Frequent episodes in Sequences[C]. In Proc. Of KDD-95, pp210-215, Montreal, Canada, Aug,1995.

[15]. C. Bettini, X. Wang, S. Jajodia. Mining Temporal Relationships with Multiple Granularities in Time Sequences[J]. IEEE Data Engineering Bulletin, Volume 21 Number 1,1998 .

[16]. C. Bettini, X. Wang, S. Jajodia. Satisfiability of Quantitative Temporal Constraints with Multiple Granularities[C]. Third International Conference on Principles and Practice of Constraint Programming (CP97), Schloss Hagenberg, Austria October 29-November 1, 1997.

[17]. C. Bettini, X. Wang, S. Jajodia. A General Framework for Time Granularity and Its Application to Temporal Reasoning[J]. Annals of Mathematics and Artificial Intelligence. Vol. 22. 1998.

[18]. G. Das, K. Lin, H. Mannila, G. Renganathan, P. Smyth. Rule Discovery from Time Series[J]. KDD, 1998, 16-22.

[19]. Povinelli, R. J. Time Series Data Mining: Identifying Temporal Patterns for Characterization and Prediction of Time Series Events[D]. PhD Dissertation, Marquette University, Milwaukee, 1999.

[20]. Povinelli, R. J. and Feng, X., Temporal Pattern Identification of Time Series Data Using Pattern Wavelets and Genetic Algorithms[C]. Artificial Neural Networks in Engineering, St. Louis, Missouri, 691-696.

[21]. E Keogh, S Lonardi, B Chiu. Finding Surprising Patterns in a Time Series Database in Linear Time and Space[J]. SIGKDD '02, july, 23~26 2002.

[22]. 曾海泉. 时间序列挖掘与相似性查找技术研究[D]. 复旦大学博士学位论文，200. pp 16-34.

[23]. 胡运发. 互关联后继树—一种新型全文数据库数学模型[R]. 复旦大学计算机与信息技术系，技术报告：CIT-02-3，2002(Hu Yunfa. Inter-relevant successive trees—A novel mathematical model for full text database(in Chinese). Department of Computer and Information Technology, Fudan University, Tech Rep:CIT-02-3, 2002)。

[24]. Keogh E, Kasetty S. On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration[C]. Proc of the SIGKDD, Edmonton, Alberta, Canada, 2002.

[25]. Shatkay, H.. Approximate Queries and Representations for Large Data Sequences[R]. Technical Report cs-95-03, Department of Computer Science, Brown

University, 1995.
[26]. Park, S., Kim, S. W., & Chu, W. W.. Segment-Based Approach for Subsequence Searches in Sequence Databases[C], To appear in Proceedings of the 16th ACM Symposium on Applied Computing. 2001.
[27]. Wang, C. & Wang, S.. Supporting content based searches on time Series via approximation[C]. Proceedings of the 12th International Conference on Scientific and Statistical Database Management. 2000.
[28]. Li, C,. Yu, P. & Castelli V.. MALM: A framework for mining sequence database at multiple abstraction levels[C]. Proceedings of the 9th International Conference on Information and Knowledge Management. Pp 267-272.1998.
[29]. Park, S. & Lee, D., & Chu, W. W. Fast Retrieval of Similar Subsequences in Long Sequence Databases"[C]. Proceedings of the 3rd IEEE Knowledge and Data Engineering Exchange Workshop.1999.
[30]. Hunter, J. & McIntosh, N. Knowledge-based event detection in complex time series data[C]. Artificial Intelligence in Medicine. pp. 271-280. Springer.1999.
[31]. Eamonn J. Keogh, Michael J. Pazzani: An Enhanced Representation of Time Series Which Allows Fast and Accurate Classification, Clustering and Relevance Feedback[J]. KDD 1998: 239-243.
[32]. Keogh & Pazzani. Relevance feedback retrieval of time series[C]. The 22th International ACM-SIGIR Conference on Research and Development in Information Retrieval, 1999.
[33]. 陈文伟，邓苏，张维明. 数据开采与知识发现综述[J]. 计算机世界专题综述，1997，6,30。
[34]. Julian Kulkarn, Richark King. Business Intelligence Systems and Data Mining[C]. ASAS Institute White Paper. 1996.
[35]. Szladow, A. J.& Ziarko, W，Knowledge-based process control using rough sets, in S.Y. Huang, ed.，'Intelligent Decision Support: Handbook of Applications and Advances of the Rough Set Theory' [M]. Kulwer Academic Publishers, chapter 4, 49-60,.
[36]. Bjorvand, A, T, Time series and rough sets[D]. Master's thesis, Department of Computer Systems and Telematics, The Norwegian Institute of Technology 1996.
[37]. 尹旭日，商琳，何佳洲，陈世福， Rough集挖掘时间序列的研究[J]. 南京大学学报(自然科学版)，2001.2。
[38]. 马志锋，邢汉承，郑晓妹， 一种基于Rough集的时间序列数据挖掘策略[J].

系统工程理论与实践，2001年12期。

[39]. T.C. Fu, F.L. Chung, R. Luk and V. Ng, "Pattern Discovery from Stock Time Series Using Self-Organizing Maps"[C]. Workshop Notes of KDD2001 Workshop on Temporal Data Mining, 26-29 Aug., San Francisco, pp.27-37, 2001.

[40], Jain, A, K. and Dubes, R. C. Algorithms for Clustering Data[M]. Englewood Cliffs, NJ: Prentice-Hall.1988.

[41]. Kaufman, L., and Rousseauw, P. J. Finding Groups in Data: An Introduction to Cluster Analysis[M]. John Wiley and Sons 1990.

[42]. C.S.Daw, C.E.A.Finney, and E.R. Tracy. Symbolic anaysis of experimental data[C]. Review of Scientific Instruments 2001, Oct. 30-31 2001.

[43]. Y.W.Huang and P.Yu. Adaptive query processing for time-series data[J]. In S.Chaudhuri and D.Madigan editors, Proc. Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 282-286. ACM Press, Aug. 15-18 ,1999.

[44]. X.Ge and P.Smyth. Deformable markov model templates for time-series pattern matching[C]. In Proceeding of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 81-90, 2000.

附 录

1. 参与的科研项目

1. 上海市政府政务协作系统， 2002. 9—2003. 7。
2. 复旦大学个性化数字图馆项目， 2003. 7—2004. 7。
3. 国泰君安证券监控管理平台风险评估项目，2004. 9—2005. 6。

2. 发表的论文

1. 秦少辉，肖辉，胡运发. 互关联后继树在时间序列特征模式挖掘中的应用。计算机工程与设计，(已录用)2006, 9。

致 谢

转眼之间，三年的硕士学习生涯即将告一段落。在完成毕业论文，即将告别校园之际，回首这三年的学习生活，不免感慨万千。多谢各位师长、同学的支持、关心和帮助，我才能有今天的成绩。

论文是在导师胡运发教授的悉心指导和无微不至的关怀下完成的，导师严谨的治学态度、渊博的专业知识、开阔而活跃的思维方法、高度的敬业精神以及对科学问题敏锐的洞察力，深深的铭刻在我的记忆之中，使我终生难忘。在此，谨向胡老师表达自己最衷心的感谢和崇高的敬意。

特别要感谢曾海泉师兄和肖辉师兄，你们在时间序列方面的优秀科研成果是是我的研究工作的基础。还要感谢张锦师兄和李荣陆师兄，你们在学习上对我的指导和科研工作中对我的帮助使我受益非浅。

感谢于玉教授、葛家祥教授、陶晓鹏老师，施兰珍老师，他们或是在我论文选题或研究方向上给予许多宝贵的意见和建议，或是在我的一些日常工作过程中给予具体协助和指导。

衷心感谢同门的兄弟姐妹们，三年来和你们的愉快合作，我将永志不忘。感谢计算机系 02 级硕士班的全体同学，这是一个团结向上的集体。感谢好友王金榜、鲁之栩、谈国华等在学习生活中给我的帮助。

最后，还要感谢一直以来深爱着我的父母、大哥、二哥以及嫂子们，你们对我的关心和支持，是我不断前进的动力源泉！

秦少辉

复旦大学北区学生公寓

2005 年 4 月 30 日

论文独创性声明

本论文是我个人在导师指导下进行的研究工作及取得的研究成果。论文中除了特别加以标注和致谢的地方外，不包含其他人或其它机构已经发表或撰写过的研究成果。其他同志对本研究的启发和所做的贡献均已在论文中作了明确的声明并表示了谢意。

作者签名：秦小辉　日期：2005.6.7

论文使用授权声明

本人完全了解复旦大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其它复制手段保存论文。保密的论文在解密后遵守此规定。

作者签名：秦小辉　导师签名：[illegible]　日期：2005.6.7