

# 上次课回顾

## 内容提要

- 聚类方法概述
- 划分聚类方法: k-means、k-prototype、k—medoids (PAM)
- 层次聚类方法: 凝聚的层次聚类 (AGNES)  
分裂的层次聚类 (DIANA)
- 密度聚类方法
- 其它聚类方法

# 第六章 时间序列和序列模式挖掘

## 内容提要

- 时间序列及其应用
- 时间序列预测的常用方法
- 基于ARMA模型的序列匹配方法
- 基于离散傅立叶变换的时间序列相似性查找
- 基于规范变换的查找方法
- 序列挖掘及其基本方法
- AprioriAll 算法
- AprioriSome 算法
- GSP算法



# 时间序列的散点图



12月02日 周三 农历十月十八



6°C  
雾(实时)

6 ~ 8°C

多云转阴

北风3-4级

201 重度

周四  
12月03日



5 ~ 7°C

阴转小雨

北风3-4级

良

周五  
12月04日



4 ~ 10°C

多云

北风微风

良

周六  
12月05日



4 ~ 9°C

多云

北风微风

良

周日  
12月06日



4 ~ 11°C

多云

北风微风

良



24小时温度

风力风向

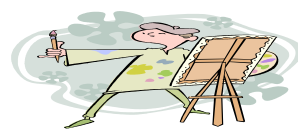
降水量

相关指数



- **时间序列** (Time Series)
- **时间序列数据挖掘**通过研究信息的时间特性，深入洞悉事物进化的机制，是获得知识的有效途径。

近年来，时间序列挖掘在宏观的经济预测、市场营销、客流量分析、太阳黑子数、月降水量、河流流量、股票价格变动等众多领域得到应用。事实上，社会、科学、经济、技术等领域中广泛存在着大量的时间序列数据有待进一步的分析和处理。





# 国家外汇储备金额 (亿万)

月度	金额	月度	金额	月度	金额	月度	金额
99年12月	1,546.75	02年6月	2,427.63	04年12月	6,099.32	07年6月	13,326.25
00年1月	1,561.00	02年7月	2,465.34	05年1月	6,236.46	07年7月	13,852.00
00年2月	1,565.59	02年8月	2,530.90	05年2月	6,426.10	07年8月	14,086.41
00年3月	1,568.20	02年9月	2,586.30	05年3月	6,591.44	07年9月	14,336.11
00年4月	1,568.46	02年10月	2,655.39	05年4月	6,707.74	07年10月	14,548.98
00年5月	1,580.19	02年11月	2,746.25	05年5月	6,910.12	07年11月	14,969.06
00年6月	1,585.68	02年12月	2,864.07	05年6月	7,109.73	07年12月	15,282.49
00年7月	1,585.96	03年1月	3,044.60	05年7月	7,327.33	08年1月	15,898.10
00年8月	1,592.17	03年2月	3,082.50	05年8月	7,532.09	08年2月	16,471.34
00年9月	1,600.92	03年3月	3,160.10	05年9月	7,690.04	08年3月	16,821.77
00年10月	1,613.44	03年4月	3,262.91	05年10月	7,849.02	08年4月	17,566.55
00年11月	1,639.11	03年5月	3,400.61	05年11月	7,942.23	08年5月	17,969.61
00年12月	1,655.74	03年6月	3,464.76	05年12月	8,188.72	08年6月	18,088.28
01年1月	1,686.23	03年7月	3,564.86	06年1月	8,451.80	08年7月	18,451.64
01年2月	1,747.73	03年8月	3,647.34	06年2月	8,536.72	08年8月	18,841.53
01年3月	1,758.47	03年9月	3,838.63	06年3月	8,750.70	08年9月	19,055.85
01年4月	1,771.78	03年10月	4,000.92	06年4月	8,950.40	08年10月	18,796.88



# 设置预报参数

 时间区间

 周期: 12

☒ 将记录扩展至未来

未来指示器字段:

预报中使用的未来值

选择要添加到数据的字段值:

字段	值

间隔 构建 估计 预报 注解

确定 取消 应用 重置



# 设置时间序列模型

**时间序列建模器: 专家建模器标准** [X]

**模型类型**

☒ 所有模型

☐ 仅限于指数平滑的模型

☐ 仅限于 ARIMA 模型

☒ 专家建模器考虑季节模型

**事件和干涉**

	字段

事件和干涉字段是特殊的独立字段，使用它对外部效应进行建模，如洪水、罢工、或者新产品线的推出。检查要用作事件和交互字段的所有字段。

**模型** 离群值

**确定** **取消** **帮助**



# 模型的参数

**专家建模**

文件 生成

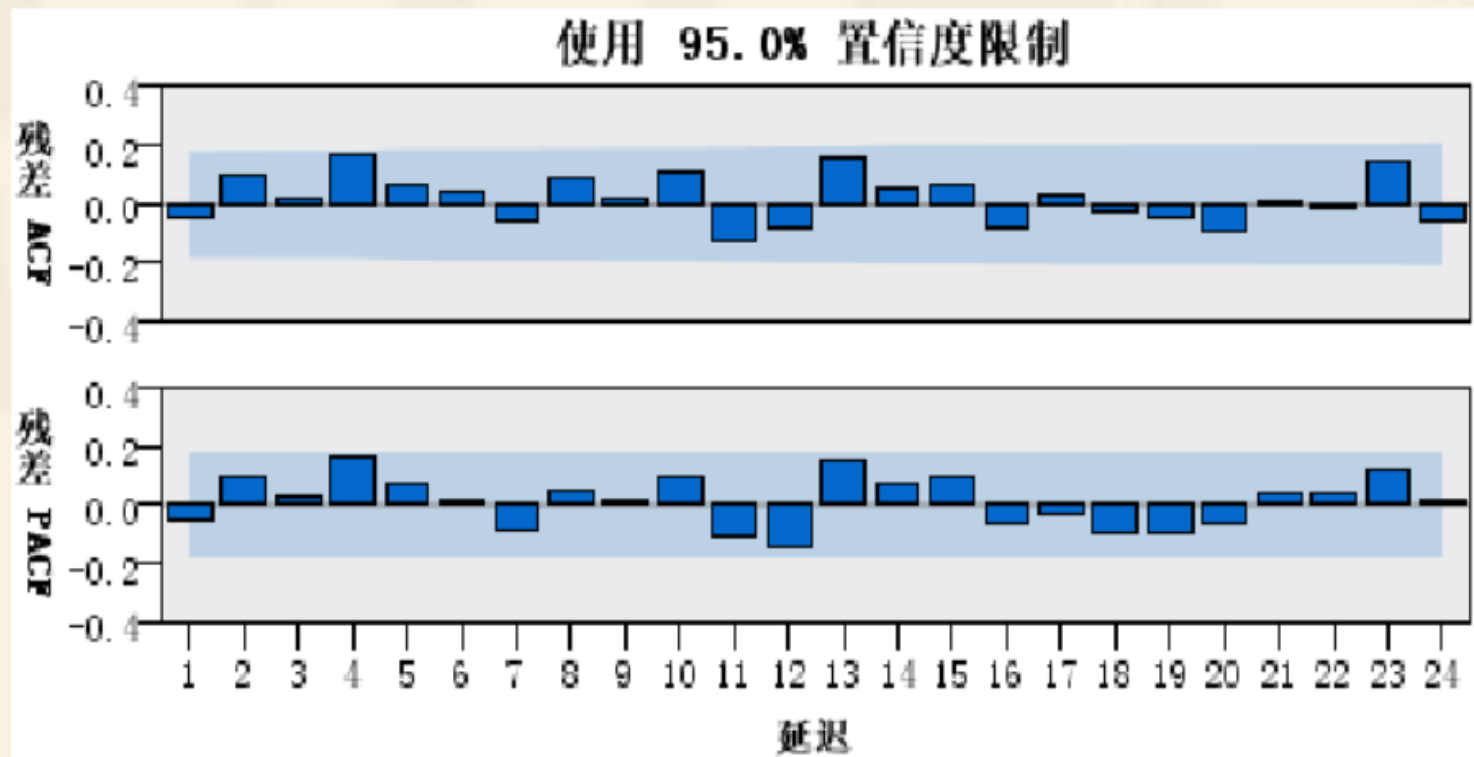
显示模型的参数: 金额 - ARIMA(1,1,0)(0,0,0) ▾

目标	模型	字段	转换	参数	延迟	评估	SE	t	Sig.
金额	ARIMA(1,1,0)...	金额	自然对数	常量		0.023	0.002	11.308	0.0
				AR	延迟 1	0.365	0.087	4.209	0.0
				差		1.0			

模型 参数 残差 汇总 设置 注解

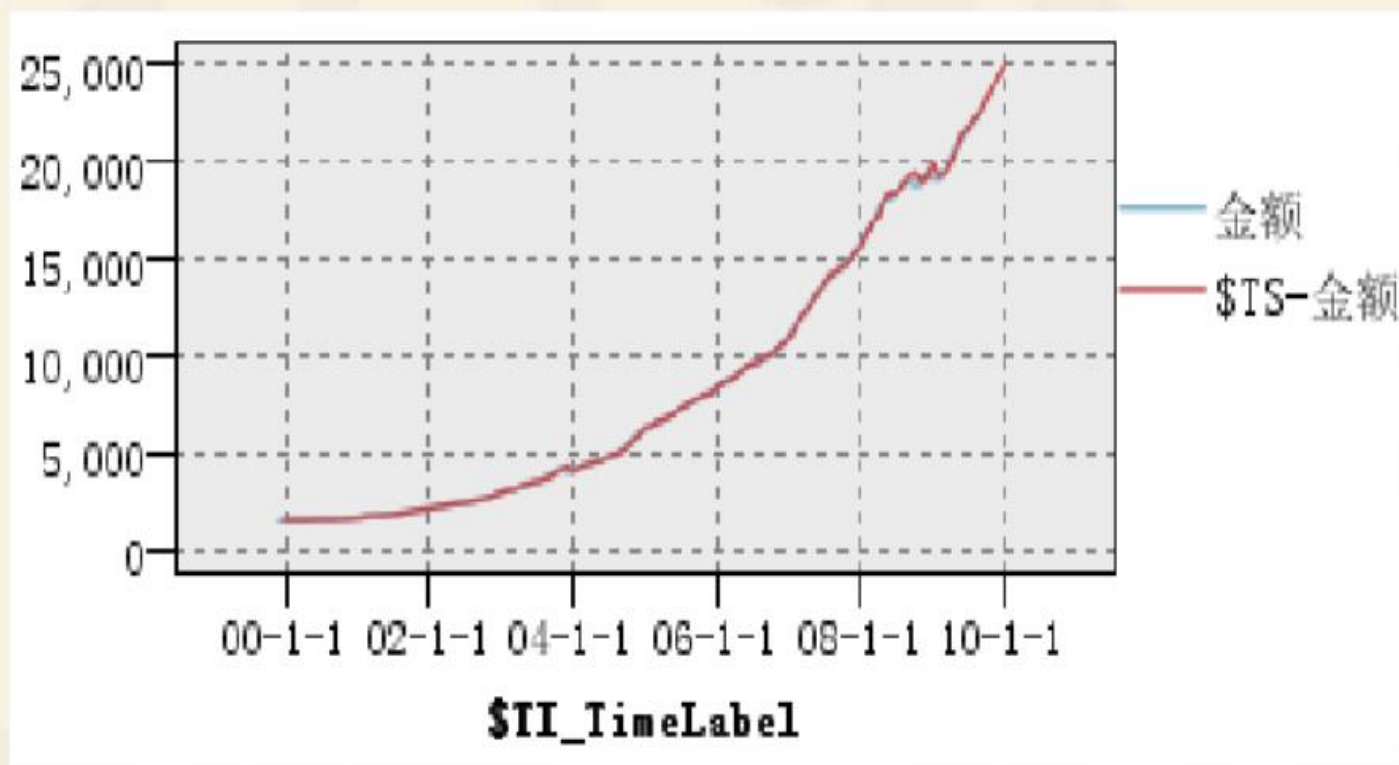
确定 取消 应用 重置

# 残差序列的自相关分析



残差序列的自相关系数全部落入95%置信区间内，说明残差序列是随机序列。

# 预测值与期望值的时间散点图



- ❖ 预测值曲线基本上覆盖在期望值之上，说明模型对数据的拟合效果很好。



# 时间序列有关概念

- 从统计意义上来讲，所谓时间序列就是将某一指标在不同时间上的不同数值，**按照时间先后顺序排列**而成的数列。
- **时间序列挖掘**就是要从大量的时间序列数据中提取人们事先不知道的、但又是潜在有用的与时间属性相关的信息和知识，并用于短期、中期或长期预测，指导人们的社会、经济、军事和生活等行为。通过对过去历史行为的客观记录分析，揭示其内在规律，进而完成预测未来行为等决策性工作。



# 时间序列有关概念

- 从数学意义上讲，如果我们对某一过程中的某一变量进行 $X(t)$ 观察测量，在一系列时刻 $t_1, t_2, \dots, t_n$  ( $t$ 为自变量，且 $t_1 < t_2 < \dots < t_n$ ) 得到的离散有序数集合 $X_{t_1}, X_{t_2}, \dots, X_{t_n}$ 称为离散数字时间序列。设 $X(t)$ 是一个随机过程， $X_{t_i}$  ( $i=1, 2, \dots, n$ )称为一次样本实现，也就是一个时间序列。
  - **一元**时间序列：如某种商品的销售量数列等，可以通过单变量随即过程的观察获得规律性信息。
  - **多元**时间序列。如包含气温、气压、雨量等在内的天气数据，通过多个变量描述变化规律。时间序列挖掘需要揭示各变量间相互依存关系的动态规律性。



时间序列的研究必须依据合适的理论和技术进行，时间序列的多样性表明其研究必须结合序列特点来找到合适的建模方法。

- **离散型**时间序列：如果某一序列中的每一个序列值所对应的时间参数为间断点，则该序列就是一个离散时间序列。
- **连续型**时间序列：如果某一序列中的每个序列值所对应的时间参数为连续函数，则该序列就是一个连续时间序列。
- **序列**的分布规律：序列的统计特征可以表现平稳或者有规律的震荡，这样的序列是分析的基础点。此外如果序列按某类规律（如高斯型）的分布，那么序列的分析就有了理论根据。

# 第六章 时间序列和序列模式挖掘

## 内容提要

- 时间序列及其应用
- 时间序列预测的常用方法
- 基于ARMA模型的序列匹配方法
- 基于离散傅立叶变换的时间序列相似性查找
- 基于规范变换的查找方法
- 序列挖掘及其基本方法
- AprioriAll 算法
- AprioriSome 算法
- GSP算法





# 时间序列预测的常用方法

时间序列分析的一个重要应用是预测，即根据已知时间序列中数据的变化特征和趋势，预测未来属性值。为了对时间序列预测方法有一个比较全面的了解，我们首先对时间序列预测的主要方法加以归纳。

- 确定性时间序列预测方法
- 随机时间序列预测方法
- 其他方法





# 时间序列预测的常用方法(续)

## ■ 确定性时间序列预测方法

■ 对于**平稳变化特征**的时间序列来说，假设未来行为与现在的行为有关，利用属性现在的值预测将来的值是可行的。例如，要预测下周某种商品的销售额，可以用最近一段时间的实际销售量来建立预测模型。

■ 一种更科学的评价时间序列变动的方法是将变化在多维上加以综合考虑，把数据的变动看成是长期趋势、季节变动和随机型变动共同作用的结果。

- **长期趋势**：随时间变化的、按照某种规则稳步增长、下降或保持在某一水平上的规律。
- **季节变动**：在一定时间内（如一年）的周期性变化规律（如冬季羽绒服销售增加）。
- **随机型变动**：不可控的偶然因素等。



# 时间序列预测的常用方法(续)

## ■ 确定性时间序列预测方法

设  $T_t$  表示长期趋势,  $S_t$  表示季节变动趋势项,  $C_t$  表示循环变动趋势项,  $R_t$  表示随机干扰项,  $y_t$  是观测目标的观测记录。则常见的确定性时间序列模型有以下几种类型:

- 加法模型:  $y_t = T_t + S_t + C_t + R_{t_0}$
- 乘法模型:  $y_t = T_t \cdot S_t \cdot C_t \cdot R_{t_0}$
- 混合模型:  $y_t = T_t \cdot S_t + R_t$  或  $y_t = S_t + T_t \cdot C_t \cdot R_{t_0}$



# 时间序列预测的常用方法(续)

## ■ 随机时间序列预测方法

- 通过建立**随机**模型，对随机时间序列进行分析，可以预测未来值。
- 若时间序列是**平稳**的，可以用自回归(Auto Regressive, 简称AR)模型、移动回归模型(Moving Average, 简称MA)或自回归移动平均(Auto Regressive Moving Average, 简称**ARMA**)模型进行分析预测。

# 第六章 时间序列和序列模式挖掘

## 内容提要

- 时间序列及其应用
- 时间序列预测的常用方法
- 基于ARMA模型的序列匹配方法
- 基于离散傅立叶变换的时间序列相似性查找
- 基于规范变换的查找方法
- 序列挖掘及其基本方法
- AprioriAll 算法
- AprioriSome 算法
- GSP算法





# 基于ARMA模型的序列匹配方法

- ARMA模型（特别是其中的AR模型）是时序方法中最基本的、实际应用最广的时序模型。早在1927年，G. U. Yule就提出了AR模型，此后，AR模型逐步发展为ARMA模型、多维ARMA模型。ARMA通常被广泛用于预测。由于ARMA模型是一个**信息的凝聚器**，可将系统的特性与系统状态的所有信息凝聚在其中，因而它也可以用于时间序列的匹配。

## ■ 1. ARMA模型

对于平稳、正态、零均值的时序  $X = \{x_t | t = 0, 1, 2, \dots, n-1\}$ ，若  $X$  在  $t$  时刻的取值不仅与其前  $n$  步的各个值  $x_{t-1}, x_{t-2}, \dots, x_{t-n}$  有关，而且还与前  $m$  步的各个干扰  $\alpha_{t-1}, \alpha_{t-2}, \dots, \alpha_{t-m}$  ( $n, m=1, 2, \dots$ )，则按多元线性回归的思想，可得到最一般的ARMA ( $n, m$ ) 模型：

$$x_t = \sum_{i=1}^n \varphi_i x_{t-i} - \sum_{j=1}^m \theta_j \alpha_{t-j} + \alpha_t$$

其中  $\alpha_t \sim NID(0, \delta_a^2)$



- **ARMA模型**参数估计的方法很多：
- 如果模型的**输入**序列 $\{u(n)\}$ 与**输出**序列 $\{a(n)\}$ 均能被测量时，则可以用最小二乘法估计其模型参数，这种估计是线性估计，模型参数能以足够的精度估计出来
- 从理论上推出了一些ARMA模型参数的最佳估计方法，但它们存在计算量大和不能保证收敛的缺点。因此工程上提出次最佳方法，即分别估计AR和MA参数，而不像最佳参数估计中那样同时估计**AR**和**MA**参数，从而使计算量大大减少。



# 基于ARMA模型的序列匹配方法(续)

## ■2. AR模型

AR (n) 模型是ARMA (n, m) 模型的一个特例。在上面ARMA (n, m) 模型表达中, 当  $\theta_i = 0$  时, 有

$$x_t = \sum_{i=1}^n \varphi_i x_{t-i} + \alpha_t$$

其中  $\alpha_t \sim NID(0, \delta_a^2)$ 。由于此时模型中没有滑动平均部分, 所以称为n阶自回归模型, 记为AR (n)。

## ■3. MA模型

MA (m) 模型是ARMA (n, m) 模型的另一个特例。在上面ARMA (n, m) 模型表达中, 当  $\varphi_i = 0$  时, 有

$$x_t = \alpha_t - \sum_{j=1}^m \theta_j \alpha_{t-j}$$

其中  $\alpha_t \sim NID(0, \delta_a^2)$  由于模型中没有自回归部分, 所以称为m阶滑动平均 (Moving Average) 模型, 记为MA (m)。



# 建立AR模型

建立AR模型的最常用方法是最小二乘法。具体方法如下：

对于AR (n) 模型，有  $x_t = \varphi_1 x_{t-1} + \varphi_2 x_{t-2} + \dots + \varphi_n x_{t-n} + \alpha_t$ ，其中  $\alpha_t \sim NID(0, \sigma_a^2)$ ，  
即可以用以下线性方程组表示：

$$x_{n+1} = \varphi_1 x_n + \varphi_2 x_{n-1} + \dots + \varphi_n x_1 + \alpha_{n+1},$$

$$x_{n+2} = \varphi_1 x_{n+1} + \varphi_2 x_n + \dots + \varphi_n x_2 + \alpha_{n+2},$$

.....)

,

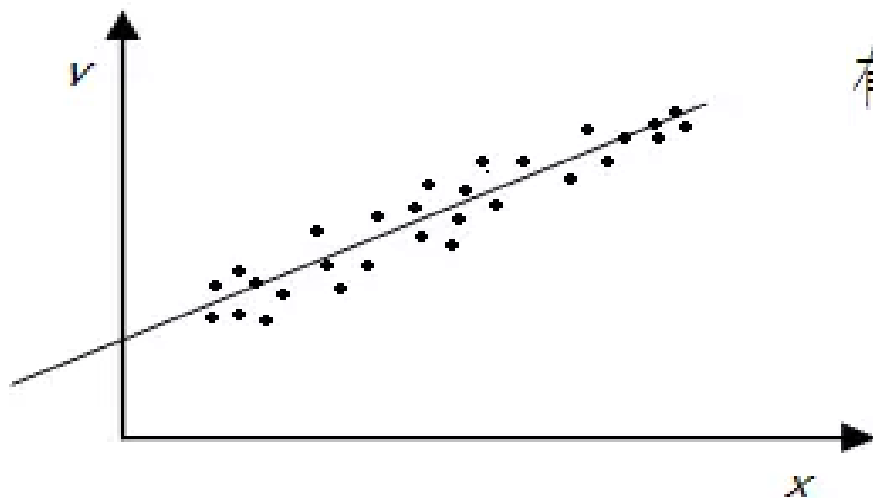
$$x_N = \varphi_1 x_{N-1} + \varphi_2 x_{N-2} + \dots + \varphi_n x_{N-n} + \alpha_N$$





# 补充：最小二乘法

- 如图，根据 $n$ 个点拟合一一条直线：



有： $[x_1, x_2, x_3, \dots, x_n]$ ,  $[y_1, y_2, y_3, \dots, y_n]$ ，求  $a$ ,  $b$ 。

$$y = ax + b$$

- 令： $f = (y_1 - ax_1 + b)^2 + (y_2 - ax_2 + b)^2 + \dots + (y_n - ax_n + b)^2$ 。
- 当 $f$ 值最小时，所有点距离目标直线距离最小。



## 补充：最小二乘法

- 由于 $f$ 是凸函数，偏微分为零的点就是极值点。

$$\frac{\partial f}{\partial a} = \sum_{i=1}^n 2(y_i - ax_i + b)x_i$$

$$\frac{\partial f}{\partial b} = \sum_{i=1}^n 2(y_i - ax_i + b)$$

- 由此可得：

$$\frac{\partial f}{\partial a} = \sum_{i=1}^n y_i x_i - a \sum_{i=1}^n x_i^2 + b \sum_{i=1}^n x_i = 0$$

$$\frac{\partial f}{\partial b} = \sum_{i=1}^n y_i - a \sum_{i=1}^n x_i + nb = 0$$

- 以上两个方程两个未知数，可以求得 $a$ ,  $b$



# 基于ARMA模型的序列匹配方法(续)

## ■2. AR模型

AR (n) 模型是ARMA (n, m) 模型的一个特例。在上面ARMA (n, m) 模型表达中, 当  $\theta_i = 0$  时, 有

$$x_t = \sum_{i=1}^n \varphi_i x_{t-i} + \alpha_t$$

其中  $\alpha_t \sim NID(0, \delta_a^2)$ 。由于此时模型中没有滑动平均部分, 所以称为n阶自回归模型, 记为AR (n)。

## ■3. MA模型

MA (m) 模型是ARMA (n, m) 模型的另一个特例。在上面ARMA (n, m) 模型表达中, 当  $\varphi_i = 0$  时, 有

$$x_t = \alpha_t - \sum_{j=1}^m \theta_j \alpha_{t-j}$$

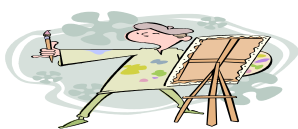
其中  $\alpha_t \sim NID(0, \delta_a^2)$  由于模型中没有自回归部分, 所以称为m阶滑动平均 (Moving Average) 模型, 记为MA (m)。

# 第六章 时间序列和序列模式挖掘

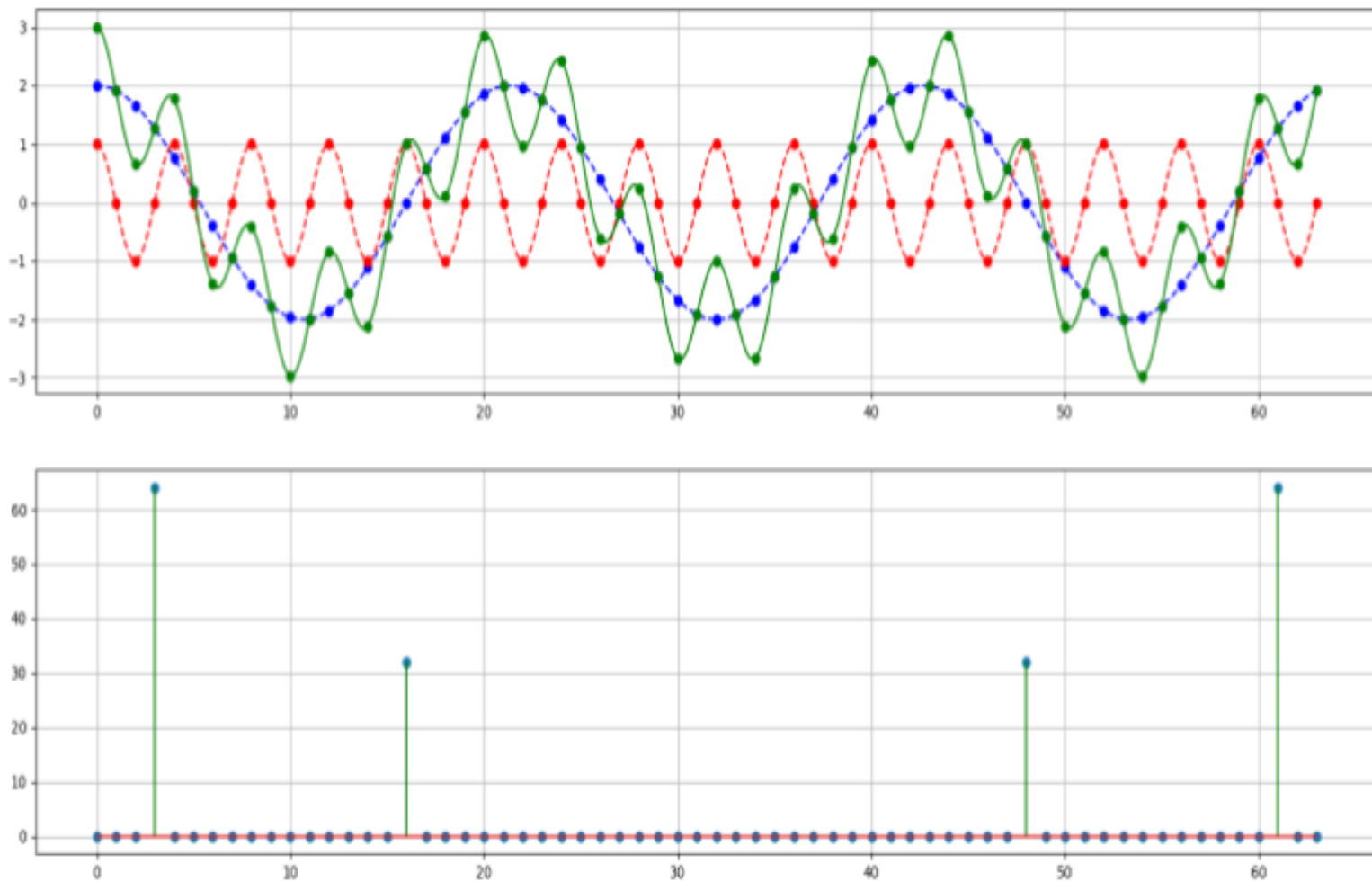
## 内容提要

- 时间序列及其应用
- 时间序列预测的常用方法
- 基于ARMA模型的序列匹配方法
- 基于离散傅立叶变换的时间序列相似性查找
- 基于规范变换的查找方法
- 序列挖掘及其基本方法
- AprioriAll 算法
- AprioriSome 算法
- GSP算法





# 为什么要做傅里叶变换

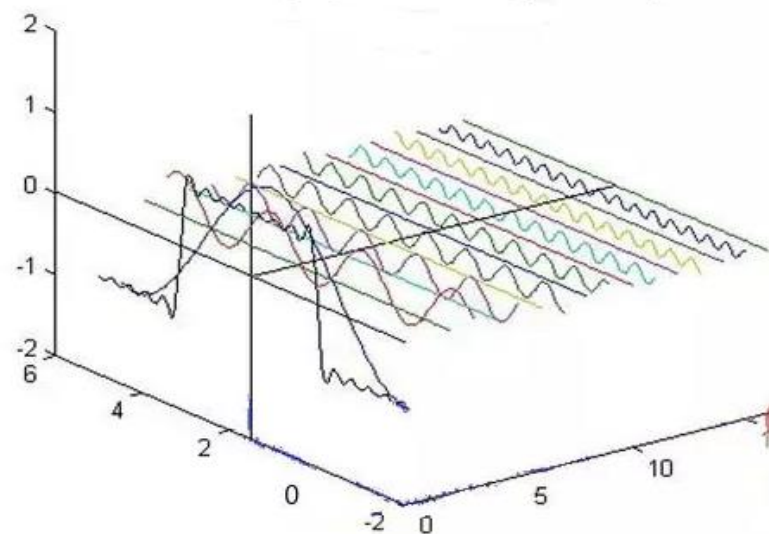
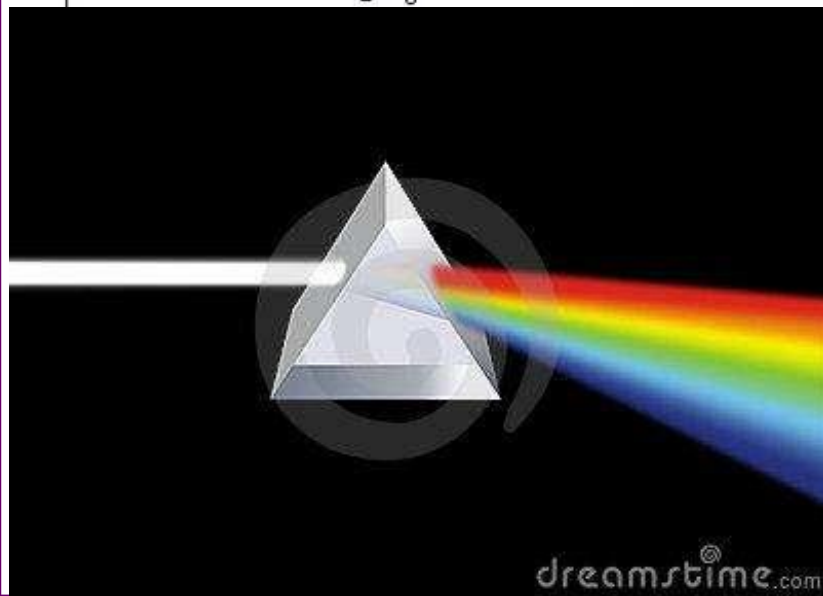
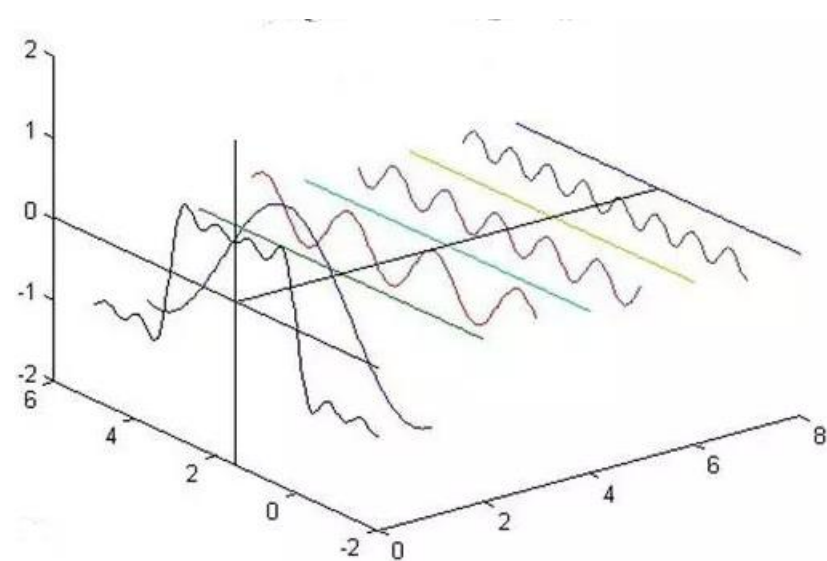
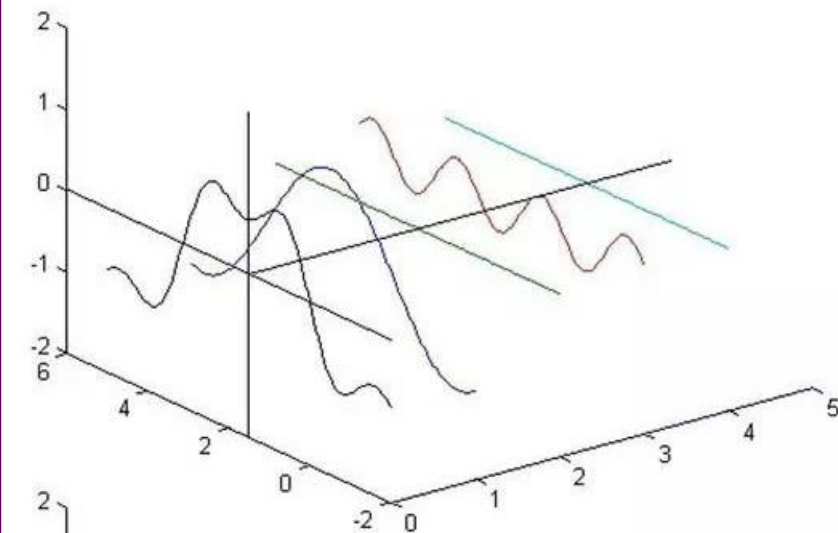
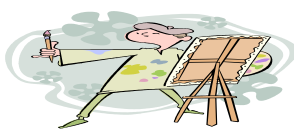




# 为什么要做傅里叶变换

- 离散傅里叶变换 (DFT) 的公式:

$$\begin{aligned} X(f) &= \sum_{t=0}^{N-1} x(t) e^{i * \frac{f}{N} * 2\pi * t}, t, f = 0, 1, \dots, N-1 \\ &= \sum_{t=0}^{N-1} x(t) \cos\left(\frac{f}{N} * 2\pi * t\right) + i * x(t) \sin\left(\frac{f}{N} * 2\pi * t\right) \end{aligned}$$





$$\frac{4 \sin \theta}{\pi}$$



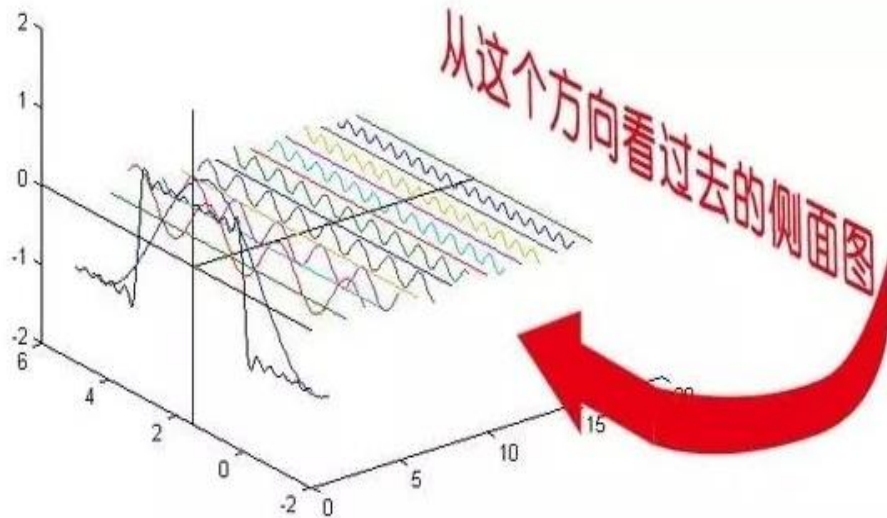
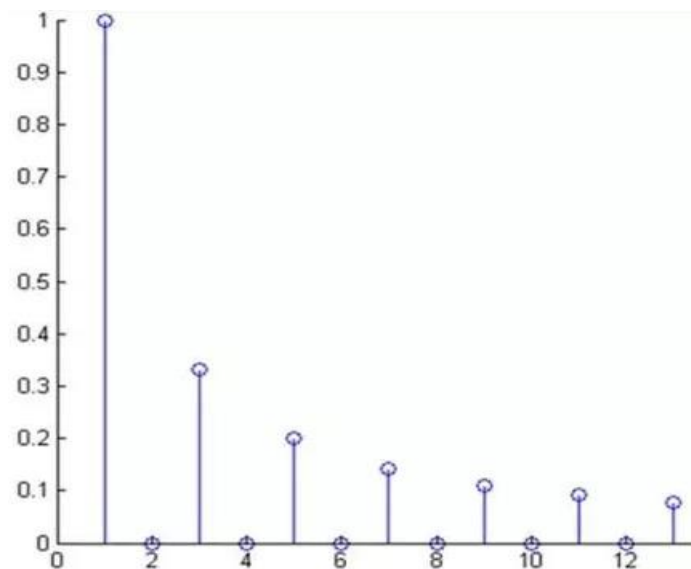
$$\frac{4 \sin 3\theta}{3\pi}$$



$$\frac{4 \sin 5\theta}{5\pi}$$



$$\frac{4 \sin 7\theta}{7\pi}$$







# 基于离散傅立叶变换的时间序列相似性查找

为了方便讨论，我们首先给出一些符号来表示序列及序列的相似性：

- $X = \{x_t | t = 0, 1, 2, \dots, n-1\}$  表示一个序列；
- $\text{Len}(X)$  表示序列 $X$ 的长度；
- $\text{First}(X)$  表示序列 $X$ 的第一个元素；
- $\text{Last}(X)$  表示序列 $X$ 的最后一个元素；
- $X[i]$ 表示 $X$ 在 $i$ 时刻的取值， $X[i] = x_i$ ；
- 序列上元素之间的“ $<$ ”关系，在序列 $X$ 上，如果 $i < j$ ，那么 $X[i] < X[j]$ ；
- 本文用  $x_s$  表示 $X$ 的子序列，如果序列 $X$ 有 $k$ 个子序列，则把这些子序列分别表示为  $X_{s_1}, X_{s_2}, \dots, X_{s_k}$ 。
- 子序列间的 $<$ 关系， $X_{s_i}, X_{s_j}$ 为 $X$ 的子序列，如果 $\text{First}(X_{s_i}) < \text{First}(X_{s_j})$ ，则称  $X_{s_i} < X_{s_j}$ 。
- 子序列重叠 (Overlap)，假定 $X_{s_1}, X_{s_2}$ 为 $X$ 的两个子序列，如果  $\text{First}(X_{s_1}) \leq \text{First}(X_{s_2}) \leq \text{Last}(X_{s_1})$  或  $\text{First}(X_{s_2}) \leq \text{First}(X_{s_1}) \leq \text{Last}(X_{s_2})$  成立，则 $X_{s_1}$ 与 $X_{s_2}$ 重叠。



# 为什么要做傅里叶变换

## 大学生破译周鸿祎手机号 李开复放“橄榄枝”

2012年08月31日21:32

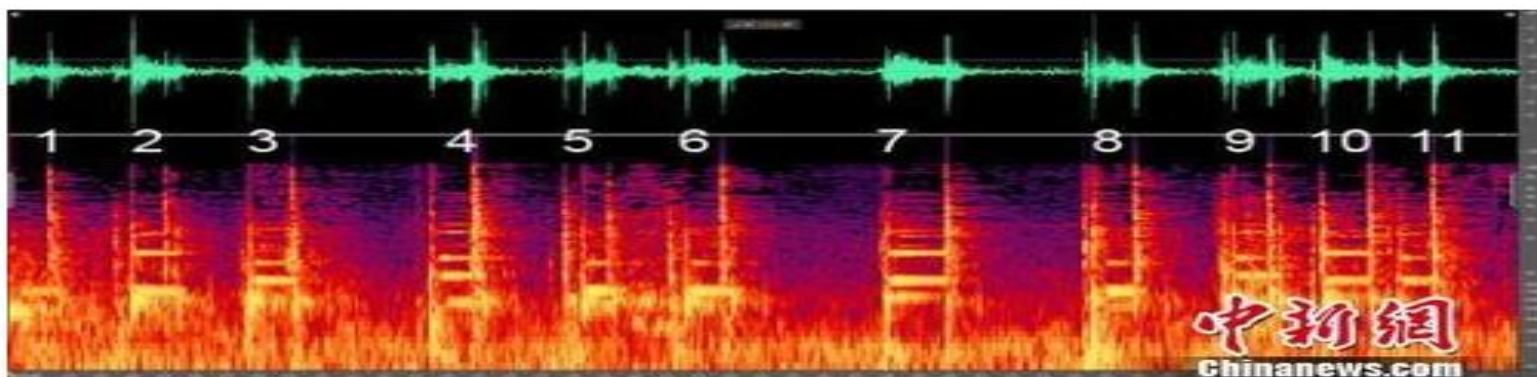
中国新闻网[微博]

孟祥磊

我要评论(0)

字号：T | T

[导读] 30日，南京大学学生刘靖康突发奇想，利用视频中记者采访时的拨号声音还原出了手机号码。



号码按键声波图 刘靖康供图 摄





# 基于离散傅立叶变换的时间序列相似性查找

- 一般地，相似性匹配可分为两类：
- **完全匹配** (Whole Matching)。给定 $N$ 个序列  $Y_1, Y_2, \dots, Y_n$  和一个查询序列 $X$ ，这些序列**有相同的长度**，如果存在  $D(X, Y_i) \leq \varepsilon$ ，那么我们称  $X$  与 $Y_i$  完全匹配。
- **子序列匹配** (Subsequence Matching)。给定 $N$ 个具有任意长度的序列  $Y_1, Y_2, \dots, Y_n$  和一个查询序列 $X$ 以及参数 $\varepsilon$ 。子序列匹配就是在  $Y_i (1 \leq i \leq N)$ 上找到**某个子序列**，使这个子序列与 $X$ 之间的距离小于等于 $\varepsilon$ 。

# 第六章 时间序列和序列模式挖掘


## 内容提要

- 时间序列及其应用
- 时间序列预测的常用方法
- 基于ARMA模型的序列匹配方法
- 基于离散傅立叶变换的时间序列相似性查找
- 基于规范变换的查找方法
- 序列挖掘及其基本方法
- AprioriAll 算法
- AprioriSome 算法
- GSP算法





# 美国国立生物技术信息中心



NCBI Resources How To Sign in to NCBI

NCBI National Center for Biotechnology Information

All Databases Search

NCBI Home

Resource List (A-Z)

All Resources

Chemicals & Bioassays

Data & Software

DNA & RNA

Domains & Structures

Genes & Expression

Genetics & Medicine

Genomes & Maps

Homology

Literature

Proteins

Sequence Analysis

Taxonomy

Training & Tutorials

Variation

## Welcome to NCBI

The National Center for Biotechnology Information advances science and health by providing access to biomedical and genomic information.

[About the NCBI](#) | [Mission](#) | [Organization](#) | [Research](#) | [NCBI News](#)

### Get Started

- [Tools](#): Analyze data using NCBI software
- [Downloads](#): Get NCBI data or software
- [How To's](#): Learn how to accomplish specific tasks at NCBI
- [Submissions](#): Submit data to GenBank or other NCBI databases

### Genetic Testing Registry

A portal to clinical genetics resources with detailed information about genetic tests and laboratories.

GO

1 2 3 4 5 6 7 8

### Popular Resources

- PubMed
- Bookshelf
- PubMed Central
- PubMed Health
- BLAST**
- Nucleotide
- Genome
- SNP
- Gene
- Protein
- PubChem

### NCBI Announcements

GenBank release 204.0 is now available via FTP  
Oct 22, 2014

Release 204.0 (10/20/2014) has  
130,000,000 1,000 2014

dbSNP human Build 142 released  
Oct 17, 2014

dbSNP human Build 142, based on the GRCh38 and GRCh37.p13 assemblies, is  
1,000 1,000 1,000

Amazon Web Services (AWS) Marketplace provides the easiest way to start an NCBI BLAST instance  
Oct 16, 2014

点击  
进入



# 美国国立生物技术信息中心

Solanum lycopersicum chromosome chr7, complete genome

Sequence ID: emb|HG975519.1 Length: 65272350 Number of Matches: 1

Range 1: 63965831 to 63965973 [GenBank](#) [Graphics](#) ▼ Next Match ▲ Previous Match

Score	Expect	Identities	Gaps	Strand
228 bits(123)	2e-56	137/143(96%)	3/143(2%)	Plus/Minus

Query 21 GTACTAATGGA--TGTCGAAATTGGAAATTGAGA-GGCAAACGGGGAAGAAAGTCCAGC 77

Sbjct 63965973 GTACTAATGGAATGTGTGCGAAATTGGAAATTGAGA-GGCAAACGGGGAAGAAAGTCCAAC 63965914

Query 78 TTAAGCTGAGTTCAGTGGGATTTTCTAATTGGACCAGAGACTTGTAAATCACTCACAAC 137

Sbjct 63965913 TAAGCTGAGTTCAGTGGGATTTTCTAATTGGACCAGAGACTTGTAAATCACTCACTACT 63965854

Query 138 TCTTTCTCTCTCTTTCCATATAG 160

Sbjct 63965853 TCTTTCTCTCTCTTTCCATATAG 63965831

匹配范围

输入序列被随机搜索出来的概率, 该值越小越好

序列编号

匹配序列长度

相似序列, 即输入序列和搜索到序列的匹配率

对比空白



BLAST®

Basic Local Alignment Search Tool

Home

Recent Results

Saved Strategies

Help

My NCBI

Sign In Register

NCBI/BLAST/blastn suite/Formatting Results - 4EXV87RA014

[Edit and Resubmit](#) [Save Search Strategies](#) [Formatting options](#) [Download](#)

[YouTube](#) [How to read this page](#) [Blast report description](#)

ZB04091969(MALINGSHU)-A68-M13+\_D09

RID [4EXV87RA014](#) (Expires on 10-23 17:22 pm)

Query ID [ldj7191](#)

Description ZB04091969(MALINGSHU)-A68-M13+\_D09

Molecule type nucleic acid

Query Length 333

Database Name nr

Description Nucleotide collection (nt)

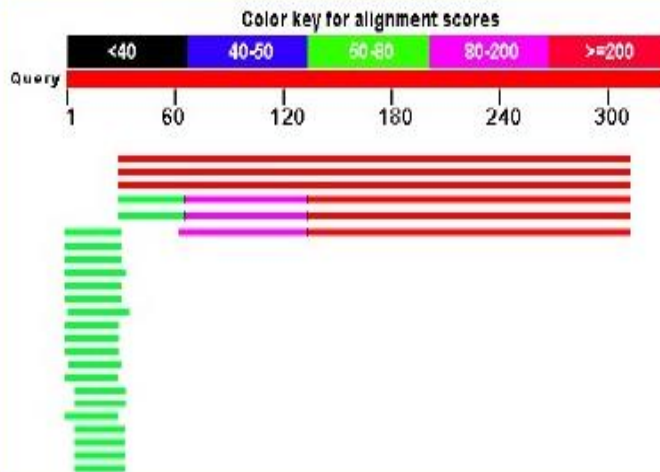
Program BLASTN 2.2.30+ [Citation](#)

Other reports: [Search Summary](#) [Taxonomy reports](#) [Distance tree of results](#)

### Graphic Summary

Distribution of 30 Blast Hits on the Query Sequence

Mouse over to see the define, click to show alignments





- **序列挖掘** 或称序列模式挖掘，是指从序列数据库中发现蕴涵的序列模式。时间序列分析和序列模式挖掘有许多相似之处，在应用范畴、技术方法等方面也有很大的重合度。但是，序列挖掘一般是指相对时间或者其他顺序出现的序列的高频率子序列的发现，典型的应用还是限于离散型的序列。
- 近年来序列模式挖掘已经成为数据挖掘的一个重要方面，其应用范围也不局限于交易数据库，在**DNA分析**等尖端科学研究领域、**Web访问**等新型应用数据源等众多方面得到针对性研究。





# 序列挖掘—基本概念

- **定义6-3:** 一个序列 (Sequence) 是项集的有序表, 记为  $\alpha = \alpha_1 \rightarrow \alpha_2 \rightarrow \dots \rightarrow \alpha_n$ , 其中每个  $\alpha_i$  是一个项集 (Itemset)。一个序列的长度 (Length) 是它所包含的项集。具有  $k$  长度的序列称为  $k$ -序列。

- **定义6-4:** 设序列  $\alpha = \alpha_1 \rightarrow \alpha_2 \rightarrow \dots \rightarrow \alpha_n$ , 序列  $\beta = \beta_1 \rightarrow \beta_2 \rightarrow \dots \rightarrow \beta_m$ 。若存在整数  $i_1 < i_2 < \dots < i_n$ , 使得

$$\alpha_1 < \beta_{i_1}, \alpha_2 < \beta_{i_2}, \dots, \alpha_n < \beta_{i_n},$$

则称序列  $\alpha$  是序列  $\beta$  的子序列, 或序列  $\beta$  包含序列  $\alpha$ 。在一组序列中, 如果某序列  $\alpha$  不包含其他任何序列中, 则称  $\alpha$  是该组中最长序列 (Maximal sequence)。



# 序列挖掘—基本概念

- 序列模式挖掘最早是由Agrawal等人提出的，它的最初动机是针对带有交易时间属性的交易数据库中发现频繁项目序列以发现某一时间段内客户的购买活动规律。
- 定义6-5 给定序列 $S$ ，序列数据库 $D_T$ ，序列 $S$ 的支持度 (Support) 是指 $S$ 在 $D_T$ 中相对于整个数据库元组而言所包含 $S$ 的元组出现的百分比。支持度大于最小支持度(min-sup)的 $k$ -序列，称为 $D_T$ 上的频繁 $k$ -序列。



# 序列挖掘—数据源的形式

表6-1带交易时间的交易数据源示例

客户号 (Cust_id)	交易时间 (Tran_time)	物品 (Item)
1	June 25'99	30
1	June 30'99	90
2	June 10'99	10, 20
2	June 15'99	30
2	June 20'99	40, 60, 70
3	June 25'99	30, 50, 70
4	June 25'99	30
4	June 30'99	40, 70
4	July 25'99	90
5	June 12'99	90

表6-2顾客序列表示例

客户号 (Cust_id)	顾客序列 (Customer Sequence)
1	< (30) (90) >
2	< (10, 20) (30) (40, 60, 70) >
3	< (30, 50, 70) >
4	< (30) (40, 70) ((90)) >
5	< (90) >



# 序列挖掘—数据源的形式(续)

操作系统及其系统进程调用是评价系统安全性的一个重要方面。

表6-3系统进程调用数据示例

进程号 (Pro_id)	调用时间 (Call_time)	调用号 (Call_id)
744	04: 01: 10: 30	23
744	04: 01: 10: 31	14
1069	04: 01: 10: 32	4
9	04: 01: 10: 34	24
1069	04: 01: 10: 35	5
744	04: 01: 10: 38	81
1069	04: 01: 10: 39	62
9	04: 01: 10: 40	16
-1		

表6-4系统调用序列数据表示例

进程号 (Pro_id)	调用序列 (Call_sequence)
744	< (23, 14, 81) >
1069	< (14, 24, 16) >
9	< (4, 5, 62) >



# 序列模式挖掘的一般步骤

序列模式挖掘主要有五个步骤：排序阶段、大项集阶段、转换阶段、序列阶段以及选最大阶段。

## 1. 排序阶段

■对数据库进行排序 (Sort)，排序的结果将原始的数据库转换成序列数据库（比较实际可能需要其他的预处理手段来辅助进行）。例如，上面介绍的交易数据库，如果以客户号 (Cust\_id) 和交易时间 (trans-time) 进行排序，那么在通过对同一客户的事务进行合并就可以得到对应的序列数据库。

客户号	交易时间	物品
1	June 25'99	30
1	June 30'99	90
2	June 10'99	10, 20
2	June 15'99	30
2	June 20'99	40, 60, 70
3	June 25'99	30, 50, 70
4	June 25'99	30
4	June 30'99	40, 70
4	July 25'99	90
5	June 12'99	90

客户号	顾客序列
1	< (30) (90) >
2	< (10, 20) (30) (40, 60, 70) >
3	< (30, 50, 70) >
4	< (30) (40, 70) ((90)) >
5	< (90) >



# 序列模式挖掘的一般步骤

序列模式挖掘主要有五个步骤：排序阶段、大项集阶段、转换阶段、序列阶段以及选最大阶段。

## 2. 大项集阶段

- 这个阶段要找出所有频繁项集（即大项集）组成的集合  $L$ 。实际上，也同步得到所有大1-序列组成的集合，即  $\{ \langle I \rangle \mid I \in L \}$ 。
- 在上面表6-2给出的顾客序列数据库中，假设支持数为2，则大项集分别是 (30)，(40)，(70)，(40, 70) 和 (90)。

客户号	顾客序列
1	$\langle (30) (90) \rangle$
2	$\langle (10, 20) (30) (40, 60, 70) \rangle$
3	$\langle (30, 50, 70) \rangle$
4	$\langle (30) (40, 70) (90) \rangle$
5	$\langle (90) \rangle$

Large Itemsets	Mapped To
(30)	1
(40)	2
(70)	3
(40, 70)	4
(90)	5



# 序列模式挖掘的一般步骤(续)

序列模式挖掘主要有五个步骤：排序阶段、大项集阶段、转换阶段、序列阶段以及选最大阶段。

## 3. 转换阶段

- 在寻找序列模式的过程中，我们要不断地进行检测一个给定的大序列集合是否包含于一个客户序列中。

表6-7给出了表6-2数据库经过转换后的数据库。比如，在对ID号为2的客户序列进行转换的时候，交易(10, 20)被剔除了，因为它并没有包含任何大项集；交易(40, 60, 70)则被大项集的集合{(40), (70), (40, 70)}代替。

Large Itemsets	Mapped To
(30)	1
(40)	2
(70)	3
(40, 70)	4
(90)	5

## 4. 序列阶段

- 利用转换后的数据库寻找频繁的序列，即大序列 (Large Sequence)。

## 5. 选最大阶段

- 在大序列集中找出最长序列 (Maximal Sequences)。

# 第六章 时间序列和序列模式挖掘

## 内容提要

- 时间序列及其应用：时间序列挖掘的概念
- 时间序列预测的常用方法
- 基于ARMA模型的序列匹配方法
- 基于离散傅立叶变换的时间序列相似性查找
- 基于规范变换的查找方法
- 序列挖掘及其基本方法
- AprioriAll 算法
- AprioriSome 算法
- GSP算法







- 在AprioriAll之前先说一下Apriori思想：如果某个项集是频繁的，那么它的所有子集也是频繁的。
- AprioriAll本质上是Apriori思想的扩张，只是在产生候选序列和频繁序列方面考虑序列元素有序的特点，将项集的处理改为序列的处理。
- AprioriAll算法将序列模式挖掘过程分为5个具体阶段，即排序阶段、找频繁项集阶段、转换阶段、产生频繁序列阶段、最大化阶段。



- AprioriAll也是如此：
- 1) **排序阶段**。数据库D以客户号为主键，交易时间为次键进行排序。这个阶段将原来的事务数据库转换成由客户序列组成的数据库。
- 2) **频繁项集阶段**。找出所有频繁项集组成的集合L。也同步得到所有频繁1-序列组成的集合。
- 3) **转换阶段**。在找序列模式的过程中，要不断地进行检测一个给定的频繁集是否包含于一个客户序列中。
- 4) **序列阶段**。利用已知的频繁集的集合来找到所需的序列。类似于关联的Apriori算法。
- 5) **最大化阶段**。在频繁序列模式集合中持出最大频繁序列模式集合



# 1.排序阶段

- 对原始数据表按客户号（作为主关键字）和交易时间（作为次关键字）进行排序，排序后通过对同一客户的事件进行合并得到对应的序列数据库，minsupport=40%。

交易数据库

客户号SID	交易时间TID	商品列表（事件）
$s_1$	6月25日	30
	6月30日	80
$s_2$	6月10日	10, 20
	6月15日	30
	6月20日	40, 60, 70
$s_3$	6月25日	30, 50, 70
$s_4$	6月25日	30
	6月30日	40, 70
	7月25日	80
$s_5$	6月12日	80



序列数据库

客户号	客户序列
$s_1$	<{30}, {80}>
$s_2$	<{10, 20}, {30}, {40, 60, 70}>
$s_3$	<{30, 50, 70}>
$s_4$	<{30}, {40, 70}, {80}>
$s_5$	<{80}>

ps://blog.csdn.net/w\_z\_y1997



## 2.找频繁项集阶段

■

1-项集	计数
{10}	1
{20}	1
{30}	4
{40}	2
{50}	1
{60}	1
{70}	3
{80}	3

删除少于 min\_sup 的项集

1-项集	计数
{30}	4
{40}	2
{70}	3
{80}	3

自连接产生候选 2-项集

2-项集	计数
{40, 70}	2

删除少于 min\_sup 的项集

2-项集	计数
{30, 40}	0
{30, 70}	1
{30, 80}	0
{40, 70}	2
{40, 80}	0
{70, 80}	0

[https://blog.csdn.net/w\\_z\\_y1997](https://blog.csdn.net/w_z_y1997)



- 然后将频繁1-项集映射成连续的整数。例如，将上面得到的L1映射成表6.4对应的整数。由于比较频繁项集花费一定时间，这样做后可以减少检查一个序列是否被包含于一个客户序列中的时间，从而使处理过程方便且高效。

频繁项集	映射成整数
{30}	1
{40}	2
{70}	3
{40, 70}	4
{80}	5

[https://blog.csdn.net/w\\_z\\_y1997](https://blog.csdn.net/w_z_y1997)



客户号	原客户序列	新客户序列	映射后
$s_1$	$\langle \{30\}, \{80\} \rangle$	$\langle \{30\}, \{80\} \rangle$	$\langle \{1\}, \{5\} \rangle$
$s_2$	$\langle \{10, 20\}, \{30\}, \{40, 60, 70\} \rangle$	$\langle \{30\}, \{\{40\}, \{70\}, \{40, 70\}\} \rangle$	$\langle \{1\}, \{2, 3, 4\} \rangle$
$s_3$	$\langle \{30, 50, 70\} \rangle$	$\langle \{\{30\}, \{70\}\} \rangle$	$\langle \{1, 3\} \rangle$
$s_4$	$\langle \{30\}, \{40, 70\}, \{80\} \rangle$	$\langle \{30\}, \{\{40\}, \{70\}, \{40, 70\}\}, \{80\} \rangle$	$\langle \{1\}, \{2, 3, 4\}, \{5\} \rangle$
$s_5$	$\langle \{80\} \rangle$	$\langle \{80\} \rangle$	$\langle \{5\} \rangle$

[https://blog.csdn.net/w\\_z\\_y1997](https://blog.csdn.net/w_z_y1997)



### 3.转换阶段

- 在寻找序列模式的过程中，要不断地进行检测一个给定的大序列集合是否包含于一个客户序列中。为此做这样的转换：
- 每个事件被包含于该事件中所有频繁项集替换。
- 如果一个事件不包含任何频繁项集，则将其删除。
- 如果一个客户序列不包含任何频繁项集，则将该序列删除。
- 转换后，一个客户序列由一个频繁项集组成的集合所取代。每个频繁项集的集合表示为 $\{e_1, e_2, \dots, e_k\}$ ，其中 $e_i$  ( $1 \leq i \leq k$ ) 表示一个频繁项集。



# AprioriAll算法

- AprioriAll算法源于频繁集算法Apriori，它把Apriori的基本思想扩展到序列挖掘中，也是一个多遍扫描数据库的算法。
- 在每一遍扫描中都利用前一遍的大序列来产生候选序列，然后在完成遍历整个数据库后测试它们的支持度。
- 在第一遍扫描中，利用大项目集阶段的输出来初始化大1-序列的集合。
- 在每次遍历中，从一个由大序列组成的种子集开始，利用这个种子集，可以产生新的潜在的大序列。
- 在第一次遍历前，所有在大项集阶段得到的大1-序列组成了种子集。





## 4.产生频繁序列阶段

- 利用转换后的序列数据库寻找频繁序列，AprioriAll算法如下：
- 输入：转换后的序列数据库 $S$ ，所有项集合 $I$ ，  
最小支持度阈值 $\text{min\_sup}$
- 输出：序列模式集合 $L$
- 方法：链接&剪枝

# 第六章 时间序列和序列模式挖掘

## 内容提要

- 时间序列及其应用
- 时间序列预测的常用方法
- 基于ARMA模型的序列匹配方法
- 基于离散傅立叶变换的时间序列相似性查找
- 基于规范变换的查找方法
- 序列挖掘及其基本方法
- AprioriAll 算法
- **AprioriSome 算法**
- GSP算法





# AprioriSome算法

AprioriSome算法可以看作是AprioriAll算法的改进，具体过程分为两个阶段：

- **前推阶段：**找出指定长度的所有大序列。
- **回溯阶段：**查找其他长度的所有大序列。

## 算法6-3 AprioriSome算法

输入：大项集阶段转换后的序列数据库  $D_T$

输出：所有最长序列

// *Forward Phase* — 前推阶段；

(1)  $L_1 = \{\text{large 1-sequences}\}$ ; //大项目集阶段的结果；

(2)  $C_1 = L_1$ ;

(3)  $\text{last} = 1$ ; //最后计数的Clast

(4) FOR ( $k = 2$ ;  $C_{k-1} \neq \emptyset$  and  $L_{\text{last}} \neq \emptyset$ ;  $k++$ ) DO BEGIN

(5) IF ( $L_{k-1} \text{ known}$ ) THEN  $C_k = \text{New candidates generated from } L_{k-1}$ ;

// $C_k$ =产生于 $L_{k-1}$ 新的候选集

(6) ELSE  $C_k = \text{New candidates generated from } C_{k-1}$ ; //  $C_k$ =产生于 $C_{k-1}$ 新的候选集

(7) IF ( $k = \text{next}(\text{last})$ ) THEN BEGIN

(9) FOR each customer-sequence  $c$  in the database DO //对于在数据库中的每一个客户序列 $c$

(10) Sum the count of all candidates in  $C_k$  that are contained in  $c$ ;

//求包含在 $c$ 中的 $C_k$ 的候选者的数目之和

(11)  $L_k = \text{Candidates in } C_k \text{ with minimum support}$ ; //  $L_k$ =在 $C_k$ 中满足最小支持度的候选者

(12)  $\text{last} = k$ ;

(13) END;

(14) END;



# AprioriSome算法(续)

// Backward Phase — 回溯阶段;

(15) FOR ( $k \leftarrow k - 1$ ;  $k \geq 1$ ;  $k \leftarrow k - 1$ ) DO

(16) IF ( $L_k$  not found in forward phase) THEN BEGIN //  $L_k$   
在前推阶段没有确定的情况

(17) Delete all sequences in  $C_k$  contained in Some  $L_i$ ,  $i > k$ ;  
//删除所有在 $C_k$ 中包含在 $L_k$ 中的序列,  $i > k$

(18) FOR each customer-sequence  $c$  in  $D_T$  DO //对于在  
DT中的每一个客户序列 $c$

(19) Sum the count of all candidates in  $C_k$  that are contained  
in  $c$ ;

//对在 $C_k$ 中包含在 $c$ 中的所有的候选这的计数

(20)  $L_k =$  Candidates in  $C_k$  with minimum support //  $L_k =$ 在  
 $C_k$ 中满足最小支持度的候选者

(21) END;

(22) ELSE Delete all sequences in  $L_k$  contained in Some  $L_i$ ,  $i$   
 $> k$ ; //  $L_k$  已知

(23) Answer =  $L_1 \cup L_2 \cup \dots \cup L_m$  //从 $k$ 到 $m$ 求 $L_k$ 的并集



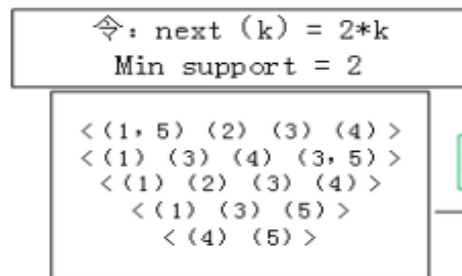
## AprioriSome算法(续)

在前推阶段（forward phase）中，我们只对特定长度的序列进行计数。比如，前推阶段我们对长度为1、2、4和6的序列计数（计算支持度），而长度为3和5的序列则在回溯阶段中计数。**next**函数以上次遍历的序列长度作为输入，返回下次遍历中需要计数的序列长度。

算法**6-4** next (k: integer)

```
IF ( $\text{hit}_k < 0.666$ ) THEN return  $k + 1$ ;  
ELSEIF ( $\text{hit}_k < 0.75$ ) THEN return  $k + 2$ ;  
ELSEIF ( $\text{hit}_k < 0.80$ ) THEN return  $k + 3$ ;  
ELSEIF ( $\text{hit}_k < 0.85$ ) THEN return  $k + 4$ ;  
ELSE THEN return  $k + 5$ ;
```

$\text{hit}_k$ 被定义为大k-序列（large k-sequence）和候选k-序列（candidate k-sequence）的比率，即 $|L_k|/|C_k|$ 。这个函数的功能是确定对哪些序列进行计数，在对非最大序列计数时间的浪费和计算扩展小候选序列之间作出权衡。



(a) 原序列数据

1-Sequences	Support
$\langle 1 \rangle$	4
$\langle 2 \rangle$	2
$\langle 3 \rangle$	4
$\langle 4 \rangle$	4
$\langle 5 \rangle$	4

(b) L1 C1  
Last = 1

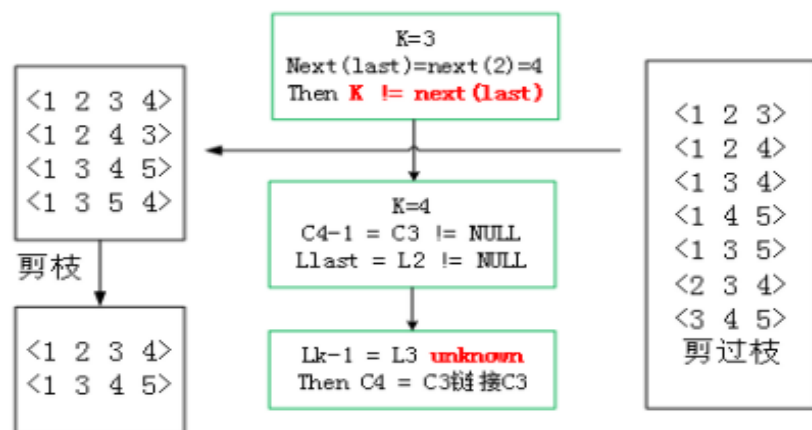
$K=2, C_{k-1} \neq \text{NULL}$   
 $L_{\text{last}} = L1 \neq \text{NULL}$

$L_{k-1} = L1$  **known**  
Then  $C2 = L1$ 链接 $L1$

$\langle 1, 2 \rangle$	$\langle 2, 1 \rangle$	$\langle 3, 1 \rangle$	$\langle 4, 1 \rangle$	$\langle 5, 1 \rangle$
$\langle 1, 3 \rangle$	$\langle 2, 3 \rangle$	$\langle 3, 2 \rangle$	$\langle 4, 2 \rangle$	$\langle 5, 2 \rangle$
$\langle 1, 4 \rangle$	$\langle 2, 4 \rangle$	$\langle 3, 4 \rangle$	$\langle 4, 3 \rangle$	$\langle 5, 3 \rangle$
$\langle 1, 5 \rangle$	$\langle 2, 5 \rangle$	$\langle 3, 5 \rangle$	$\langle 4, 5 \rangle$	$\langle 5, 4 \rangle$

(c) C2

$K=2$   
 $\text{next}(\text{last}) = \text{next}(1) = 2$   
Then  $K = \text{next}(\text{last})$



(e) C3

Last = k = 2  
 $K=3$   
 $C3-1 \neq \text{NULL}$   
 $L2 \neq \text{NULL}$

$L3-1 = L2$  **known**  
Then  $C3 = L2$ 链接 $L2$

2-Sequences	Support
$\langle 1 2 \rangle$	2
$\langle 1 3 \rangle$	4
$\langle 1 4 \rangle$	3
$\langle 1 5 \rangle$	2
$\langle 2 3 \rangle$	2
$\langle 2 4 \rangle$	2
$\langle 3 4 \rangle$	3
$\langle 3 5 \rangle$	2
$\langle 4 5 \rangle$	2

(d) L2

(f) C4

$K=4$   
 $\text{Next}(\text{last}) = \text{next}(2) = 4$   
Then  $K = \text{next}(\text{last})$

Last = k = 4  
 $K=5$   
 $C4 \neq \text{NULL}$   $L4 \neq \text{NULL}$

$L5-1=L4$  **known**  
Then  $C5=L4$ 链接 $L4$

$K=5$   
 $\text{Next}(\text{last}) = \text{next}(4) = 8$   
 $C5 = \text{NULL}$

$C5 = \text{NULL}$

Forward  
phase  
End

4-Sequences	Support
$\langle 1 2 3 4 \rangle$	2
$\langle 1 3 4 5 \rangle$	1

(g) L4

Forward  
phase



3-Sequences	Support
<del>&lt;1 4 5&gt;</del>	4
<1 3 5>	2
<del>&lt;3 4 5&gt;</del>	4

(j) L3

$K=K-- = 2$   
**L2 known**  
 删除<1 2 3 4>和  
 <1 3 5>的子序列

2-Sequences	Support
<del>&lt;1 2&gt;</del>	2
<1 3>	4
<del>&lt;1 4&gt;</del>	3
<del>&lt;1 5&gt;</del>	2
<del>&lt;2 3&gt;</del>	2
<del>&lt;2 4&gt;</del>	2
<del>&lt;3 4&gt;</del>	3
<del>&lt;3 5&gt;</del>	2
<4 5>	2

(k) L2

$K=K-- = 1$   
**L1 known**

L1 = NULL

Sequences	Support
<1 2 3 4>	2
<1 3 5>	2
<4 5>	2

(l) sequence pattern

~~<1 2 3>~~  
~~<1 2 4>~~  
~~<1 3 4>~~  
 <1 4 5>  
 <1 3 5>  
~~<2 3 4>~~  
 <3 4 5>

(i) C3

$K=K-- = 3$   
**L3 unknown**

4-Sequences	Support
<1 2 3 4>	2

(h) L4

$K=K-- = 4$   
**L4 known**

**Backward  
phase**



# AprioriAll和AprioriSome比较

## AprioriAll和AprioriSome比较

- AprioriAll用 $L_{k-1}$ 去算出所有的候选 $C_k$ ，而AprioriSome会直接用 $C_{k-1}$ 去算出所有的候选 $C_k$ ，因为 $C_{k-1}$ 包含 $L_{k-1}$ ，所以AprioriSome会产生比较多的候选。
- 虽然AprioriSome跳跃式计算候选，但因为它所产生的候选比较多，可能在回溯阶段前就占满内存。
- 如果内存满了，AprioriSome就会被强迫去计算最后一组的候选，（即使原本是要跳过此项）。这样，会影响并减少已算好的两个候选间的跳跃距离，而使得AprioriSome会变的跟AprioriAll一样。
- 对于较低的支持度，有较长的大序列，也因此有较多的非最大序列，所以AprioriSome比较好。

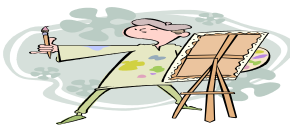


# 第六章 时间序列和序列模式挖掘

## 内容提要

- 时间序列及其应用
- 时间序列预测的常用方法
- 基于ARMA模型的序列匹配方法
- 基于离散傅立叶变换的时间序列相似性查找
- 基于规范变换的查找方法
- 序列挖掘及其基本方法
- AprioriAll 算法
- AprioriSome 算法
- **GSP算法**





## GSP算法主要包括三个步骤:

- ①扫描序列数据库, 得到长度为1的序列模式 $L_1$ , 作为初始的种子集;
- ②根据长度为 $i$ 的种子集 $L_i$ 通过连接操作和剪切操作生成长度为 $i+1$ 的候选序列模式 $C_{i+1}$ ; 然后扫描序列数据库, 计算每个候选序列模式的支持数, 产生长度为 $i+1$ 的序列模式 $L_{i+1}$ , 并将 $L_{i+1}$ 作为新的种子集;
- ③重复第二步, 直到没有新的序列模式或新的候选序列模式产生为止。

## 其中, 产生候选序列模式主要分两步:

- **连接阶段:** 如果去掉序列模式 $S_1$ 的第一个项目与去掉序列模式 $S_2$ 的最后一个项目所得到的序列相同, 则可以将 $S_1$ 与 $S_2$ 进行连接, 即将 $S_2$ 的最后一个项目添加到 $S_1$ 中。
- **剪切阶段:** 若某候选序列模式的某个子序列不是序列模式, 则此候选序列模式不可能是序列模式, 将它从候选序列模式中删除。

## 候选序列模式的支持度计算按照如下方法进行:

- 对于给定的候选序列模式集合 $C$ , 扫描序列数据库 $D$ , 对于其中的每一条序列 $d$ , 找出集合 $C$ 中被 $d$ 所包含的所有候选序列模式, 并增加其支持度计数。



# GSP算法部分例子

例子 **6-5** 表6-9演示了从长度为3的序列模式产生长度为4的候选序列模式的过程:

- 序列 $\langle (1, 2), 3 \rangle$ 可以与 $\langle 2, (3, 4) \rangle$ 连接, 因为 $\langle (*, 2), 3 \rangle$ 与 $\langle 2, (3, *) \rangle$ 是相同的, 两序列连接后为 $\langle (1, 2), (3, 4) \rangle$
- $\langle (1, 2), 3 \rangle$ 与 $\langle 2, 3, 5 \rangle$ 连接, 得到 $\langle (1, 2), 3, 5 \rangle$ 。

剩下的序列是不能连接的, 比如 $\langle (1, 2), 4 \rangle$ 不能与任何长度为3的序列连接, 这是因为其他序列没有 $\langle (2), (4, *) \rangle$ 或者 $\langle (2), (4), (*) \rangle$ 的形式。

表6-9 GSP算法举例

Sequential patterns With Length 3	Candidate4-Sequences	
	After Join	After Pruning
$\langle (1, 2), 3 \rangle$ $\langle (1, 2), 4 \rangle$ $\langle 1, (3, 4) \rangle$ $\langle (1, 3), 5 \rangle$ $\langle 2, (3, 4) \rangle$ $\langle 2, 3, 5 \rangle$	$\langle (1, 2), (3, 4) \rangle$ $\langle (1, 2), 3, 5 \rangle$	$\langle (1, 2), (3, 4) \rangle$

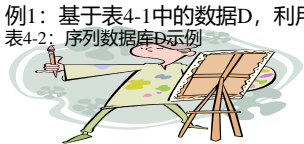
在修剪阶段 $\langle (1, 2), 3, 5 \rangle$ 将被剪掉, 这是因为 $\langle 1, 3, 5 \rangle$ 并不在 $L_3$ 中, 而 $\langle (1, 2), (3, 4) \rangle$ 的长度为3的子序列都在 $L_3$ 因而被保留下来。



# GSP算法完整举例

例：基于表中的数据D，利用GSP算法进行序列模式挖掘，最小支持数为2  
(在不引起误解的情况下表中省略了逗号和括号)

序列数据库D	
用户	访问序列
10	$\langle a(abc)(ab)d(cf) \rangle$
20	$\langle (ad)c(bc)(ae) \rangle$
30	$\langle (ef)(ab)(df)cb \rangle$
40	$\langle eg(af)cbc \rangle$



# GSP算法完整举例

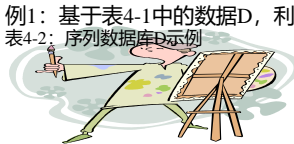
1) 对原始数据库D进行扫描，查找长度为1的频繁序列，构造种子集S1  
扫描发现频繁序列及其支持度  $\langle a \rangle:4$ ;  $\langle b \rangle:4$ ;  $\langle c \rangle:4$ ;  $\langle d \rangle:3$ ;  $\langle e \rangle:3$   
 $\langle f \rangle:3$ ，从而得到种子集  $S1 = \{\langle a \rangle, \langle b \rangle, \langle c \rangle, \langle d \rangle, \langle e \rangle, \langle g \rangle, \langle h \rangle\}$ ;  
由种子集S1构造长度为2的候选集C2  
可以直接连接S1无需判断，也无需进行剪枝。

2) S2生成过程。

$\langle aa \rangle, \langle aa \rangle, \langle ab \rangle, \langle ab \rangle, \langle ba \rangle, \langle ac \rangle, \langle ac \rangle, \langle ca \rangle, \langle ad \rangle, \langle ad \rangle,$   
 $\langle da \rangle, \langle ae \rangle, \langle ae \rangle, \langle ea \rangle, \langle af \rangle, \langle af \rangle, \langle fa \rangle, \langle bb \rangle, \langle bb \rangle, \langle bc \rangle, \langle$   
 $bc \rangle, \langle cb \rangle, \langle bd \rangle, \langle bd \rangle, \langle db \rangle, \langle be \rangle, \langle be \rangle, \langle eb \rangle, \langle bf \rangle, \langle bf \rangle, \langle fb$   
 $\rangle, \langle cc \rangle, \langle cc \rangle, \langle cd \rangle, \langle cd \rangle, \langle dc \rangle, \langle ce \rangle, \langle ce \rangle, \langle ec \rangle, \langle cf \rangle, \langle cf \rangle,$   
 $\langle fc \rangle, \langle dd \rangle, \langle dd \rangle, \langle de \rangle, \langle de \rangle, \langle ed \rangle, \langle df \rangle, \langle df \rangle, \langle fd \rangle, \langle ee \rangle, \langle$   
 $ee \rangle, \langle ef \rangle, \langle ef \rangle, \langle fe \rangle, \langle ff \rangle, \langle ff \rangle$

因为1项序列都频繁，所以无需剪枝，可以直接保留连接后的序列。

S2的种子集:  $\{\langle aa \rangle, \langle a,b \rangle, \langle ab \rangle, \langle ba \rangle, \langle ac \rangle, \langle ca \rangle, \langle ad \rangle,$   
 $\langle ea \rangle, \langle af \rangle, \langle bb \rangle, \langle b,c \rangle, \langle bc \rangle, \langle cb \rangle, \langle bd \rangle, \langle db \rangle, \langle bf \rangle,$   
 $\langle cc \rangle, \langle dc \rangle, \langle ec \rangle, \langle cf \rangle, \langle fc \rangle, \langle ef \rangle, \langle ff \rangle\}$ 。



## GSP算法完整举例

3) 由种子集S2构造长度为3的候选集C3,由于S2的序列太多，仅以S2中的三个序列 $\langle aa \rangle$ ,  $\langle (ab) \rangle$ ,  $\langle ab \rangle$ 来演示连接和剪枝的原理：

连接阶段： $\langle aa \rangle$ 序列去掉第一个元素a后得到序列 $\langle a \rangle$ ， $\langle (a,b) \rangle$ 序列去掉最后一个元素b后也得到序列 $\langle a \rangle$ ，所以 $\langle aa \rangle$ 与 $\langle (a,b) \rangle$ 这两个序列是可以连接的，将 $\langle (a,b) \rangle$ 的最后一个元素b连接到 $\langle aa \rangle$ 序列的最后，由于元素b在原序列属于最后一个集合中，因此要将b与a合并形成集合 $(a,b)$ ，所以连接后得到的序列为 $\langle a(a,b) \rangle$ 。同理，发现 $\langle aa \rangle$ 与 $\langle ab \rangle$ 也可以连接，得到序列 $\langle aab \rangle$ 。

剪枝阶段： $\langle a(a,b) \rangle$ 其所有相邻子序列 $\langle ab \rangle$ ， $\langle aa \rangle$ ， $\langle (a,b) \rangle$ 都包含在S2中，不用剪掉； $\langle aab \rangle$ 同理剪枝后

$S3 = \{ \langle aaa \rangle, \langle a(a,b) \rangle, \langle aab \rangle, \langle baa \rangle, \langle aac \rangle, \langle caa \rangle, \langle aad \rangle, \langle eaa \rangle, \langle aaf \rangle, \langle (a,b)a \rangle, \langle b(a,b) \rangle, \langle bab \rangle, \langle c(a,b) \rangle, \langle cab \rangle, \langle (a,b)b \rangle, \langle (a,b)c \rangle, \langle (a,b)d \rangle, \langle (a,b)f \rangle, \langle aba \rangle, \langle abb \rangle, \langle a(b,c) \rangle, \langle abc \rangle, \langle abd \rangle, \langle abf \rangle, \langle bac \rangle, \langle bad \rangle, \langle baf \rangle, \langle bba \rangle, \langle cba \rangle, \langle aca \rangle, \langle cac \rangle, \langle eac \rangle, \langle acb \rangle, \langle acc \rangle, \langle acf \rangle, \langle caf \rangle, \langle (b,c)a \rangle, \langle bca \rangle, \langle cca \rangle, \langle eca \rangle, \langle adb \rangle, \langle adc \rangle, \langle eaf \rangle, \langle afc \rangle, \langle aff \rangle, \langle bbb \rangle, \langle b(b,c) \rangle, \langle bbc \rangle, \langle cbb \rangle, \langle bbd \rangle, \langle dbb \rangle, \langle bbf \rangle, \langle (b,c)b \rangle, \langle c(b,c) \rangle, \langle cbc \rangle, \langle d(b,c) \rangle, \langle dbc \rangle, \langle (b,c)c \rangle, \langle (b,c)f \rangle, \langle bcb \rangle, \langle bcc \rangle, \langle bcf \rangle, \langle cbf \rangle, \langle ccb \rangle, \langle dcb \rangle, \langle bdb \rangle, \langle bdc \rangle, \langle bfc \rangle, \langle bff \rangle, \langle ccc \rangle, \langle dcc \rangle, \langle ecc \rangle, \langle ccf \rangle, \langle fcc \rangle, \langle ecf \rangle, \langle cfc \rangle, \langle fcf \rangle, \langle cff \rangle, \langle efc \rangle, \langle ffc \rangle, \langle eff \rangle, \langle fff \rangle \}$





## GSP算法完整举例

3) 由种子集S3构造长度为3的候选集S4

$C4 = \{ \langle (a,b)(c,d) \rangle, \langle (a,b)dc \rangle, \langle a(a,b,c) \rangle, \langle a(b,c)a \rangle, \langle e a(a,c) \rangle, \langle eaca \rangle, \langle ea(b,c) \rangle, \langle eachb \rangle, \langle ea(c,c) \rangle, \langle eacc \rangle, \langle ea(c,f) \rangle, \langle eacf \rangle, \langle ad(b,c) \rangle, \langle adcb \rangle, \langle eafc \rangle, \langle bd(b,c) \rangle, \langle bdc b \rangle \}$

$S4 = \{ \langle (a,b)dc \rangle, \langle a(b,c)a \rangle, \langle eafc \rangle \}$ 。

4) 算法运行结束。





## 算法6-5 GSP算法

输入：大项集阶段转换后的序列数据库 $D_T$ 。

输出：最大序列

- (1)  $L_1 = \{\text{large 1-sequences}\}$ ; // 大项集阶段得到的结果
- (2) FOR ( $k = 2$ ;  $L_{k-1} \neq \emptyset$ ;  $k++$ ) DO BEGIN
- (3)      $C_k = \text{GSPgenerate}(L_{k-1})$ ;
- (4)     FOR each customer-sequence  $c$  in the database  $D_T$  DO
- (5)         Increment the count of all candidates in  $C_k$  that are contained in  $c$ ;
- (6)      $L_k = \text{Candidates in } C_k \text{ with minimum support}$ ;
- (7)     END;
- (8) Answer = Maximal Sequences in  $\bigcup_k L_k$ ;

# 第六章 时间序列和序列模式挖掘

## 内容提要

- 时间序列及其应用
- 时间序列预测的常用方法
- 基于ARMA模型的序列匹配方法
- 基于离散傅立叶变换的时间序列相似性查找
- 基于规范变换的查找方法
- 序列挖掘及其基本方法
- AprioriAll 算法
- AprioriSome 算法
- GSP算法





Thank you !!!