计算机科学2000Vol.27№.5

时序数据库中相似序列的挖掘

Mining Similar Sequence in Time-Series Database

段立娟 高 文 王伟强

(中国科学院计算技术研究所 北京 100080)

Abstract Time-series data are of growing importance in many new database applications, such as data mining or warehousing. Sequences constitute a large portion of data stored in computers. There have been several efforts to model time-sequenced data, to design languages to query such data, and to develop access structures to efficiently process such as queries. Most of the work, however, is focussed on "exact" queries. New emerging application, particularly database mining applications, require that databases be enhanced with the capability to process "similarity" queries. Three methods of similarity-based search are introduced, and the characteristics of these methods are also discussed.

Key words Time series, Similarity-based search

1 引言

时间序列是指按时间顺序取得的一系列观测值[1~3],这里的"时间"具有广义坐标轴的含义,既可以指按时间的先后顺序排列的数据,也可以指按空间的前后顺序排列的随机数据[1],从经济到工程技术,从天文到地理和气象,几乎在各种领域都会遇到时间序列。例如某地区的逐月降雨量,其实际记录结果,按月份先后排列,便是一个时间序列。在科技飞速发展的今天,越来越多的时间序列信息被存储在计算机上,例如证券公司的计算机积累了大量的股票信息,商场的POS系统搜集了大量的销售信息,人造卫星观测的气象信息和科学仪器所检测到的大量生物、地矿等信息也都被存储在计算机上。

时间序列中包含着很多有用的信息。例如,心电图和脑电图就包含着关于人健康状况的丰富信息。对时间序列进行分析具有很重要的价值。在数据挖掘和数据仓库这些新的数据库技术中,已经开始对时间序列分析重视起来。目前,在时间序列方面的研究主要集中在,时间序列查询语言和快速存取结构的设计上,这些工作着重于精确查找,而大多数新型的数据库应用,特别是数据挖掘应用需要数据库具备相似(similarity)查找能力[4,5]。如何在几兆,甚至几十兆的时间序列数据库中发现两个模式相似的序列,显然,靠手工处理很难胜任这样的工作,传统的数据库查找方法也难以完成此类任务,因此时间序列相似性查找成为目前数据挖掘领域的一个新的研究课题。

在本文中,我们用如下符号来表示序列[1~3]及序列的相似性[4~8]:

(1)$X=(x_t|t=0,1,2,\cdots,n-1)$表示一个序列;

(2)$Len(X)$表示序列 X 的长度;

(3)$First(X)$表示序列 X 的第一个元素;

(4)$Last(X)$表示序列 X 的最后一个元素;

(5)$X[t]$表示 X 在 t 时刻的取值,$X[t]=x_t$;

(6)序列上元素之间的"<"关系,在序列 X 上,如果 $i<j$,那么 $X[i]<X[j]$;

(7)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 $X_{S1},X_{S2},\cdots,X_{SK}$;

(8)子序列间的<关系,X_{Si},X_{Sj}为 X 的子序列,如果 $First(X_{Si})<First(X_{Sj})$,则称 $X_{Si}<X_{Sj}$;

(9)子序列重叠。假定 X_{S1},X_{S2}为 X 的两个子序列,如果 $First(X_{S1})\leqslant First(X_{S2})\leqslant Last(X_{S1})$或 $First(X_{S2})\leqslant First(X_{S1})\leqslant Last(X_{S2})$成立,则 X_{S1}与 X_{S2}重叠。

(10)序列的相似性查找就是在序列数据库中发现与给定序列的模式很相似的序列,在进行序列的相似性查找之前要给定一个相似性评价函数和一个阈值 ε,如果函数值小于等于 ε,则表明序列相似。通常用 X 与 Y 之间的距离函数 $D(X,Y)$来作为序列X与Y的相似性判别函数。相似性匹配可分为两类[5~7],①完全匹配。给定N个序列 $Y_1,Y_2,\cdots,Y_n$和一个查询序列X,这些序列有相同的长度,如果存在 $D(X,Y_i)\leqslant\varepsilon$,那么我们称 X 与 Y_i 完全

匹配。

②子序列匹配。给定 N 个具有任意长度的序列 $Y_1, Y_2, \cdots, Y_n$ 和一个查询序列 X，以及参数 ϵ，子序列匹配就是在 $Y_i (1 \leqslant i \leqslant N)$ 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 ϵ。

通常人们用 X 与 Y 之间的欧几里德距离或城区距离(City-block Distance)来代替 $D(X,Y)$[6]，如果计算结果小于等于给定的阈值 ϵ，则表明 X 与 Y 相似。基于上述距离函数比较方法的致命弱点是对噪音太敏感[3,6]，而且一般情况下，序列都很长，因此计算距离需要比较长的时间[6]。如果能从序列中抽取少量的、主要的特征则可以大大提高序列的查找速度[5,8]，基于此人们把时间序列分析的主要模型——自回归滑动平均模型(ARMA)[1~3]和离散傅立叶变换(DFT)[5~6]用在了时间序列的匹配方面。本文分别讨论基于这两种模型的时间序列相似性查找方法。本文所讨论的基于规范变换的方法在时序查找中考虑了噪音、幅度、偏移问题，与前面两种方法相比有较高的查询效率。

2 基于 ARMA 模型的序列匹配方法

ARMA(Autoregression Moving Average，自回归滑动平均)模型(特别是其中的 AR(自回归)模型)是时序方法中最基本的、实际应用最广的时序模型[1~3]。1927年，G. U. Yule 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[1]，因而它也可以用于时间序列的匹配。

2.1 基本概念

ARMA(n,m)模型[1~3] 对于平稳、正态、零均值的时序 $X=\{x_t | t=0,1,2,\cdots,n-1\}$，若 X 在 t 时刻的取值不仅与其前 n 步的各个值 $x_{t-1}, x_{t-2}, \cdots, x_{t-n}$ 有关，而且还与前 m 步的各个干扰 $a_{t-1}, a_{t-2}, \cdots, a_{t-m}$ 有关($n,m=1,2,\cdots$)，则按多元线性回归的思想，可得到最一般的 ARMA 模型：

$$x_t=\sum_{i=1}^{n}\varphi_i x_{t-i}-\sum_{j=1}^{m}\theta_j a_{t-j}+a_t，其中\ a_t\sim NID(0,\delta_a^2) \quad (1)$$

ARMA(n,m)模型的特例

(1)自回归 AR(n)模型。在1式中，当 $\theta_j=0$ 时，模型中没有滑动平均部分，称为 n 阶自回归模型，记为 AR(n)。其形式为：

$$x_t=\sum_{i=1}^{n}\varphi_i x_{t-i}+a_t，其中\ a_t\sim NID(0,\delta_a^2) \quad (2)$$

(2)滑动平均 MA(m)模型。在式1中，当 $\varphi_i=0$ 时，模型中没有自回归部分，称为 m 阶滑动平均模型，记为 MA(m)。其形式为：

$$x_t=a_t-\sum_{j=1}^{m}\theta_j a_{t-j}，其中\ a_t\sim NID(0,\delta_a^2) \quad (3)$$

2.2 建立模型

首先应该建立序列的 ARMA 模型，然后构造判别函数作为衡量序列相似的尺度。由于计算速度的要求，建议采用 AR 模型[1]。建立 AR 模型的常用方法是最小二乘法[1~2]。具体如下：

对于 AR(n)模型，$x_t=\varphi_1 x_{t-1}+\varphi_2 x_{t-2}+\cdots+\varphi_n x_{t-n}+a_t$，$a_t\sim NID(0,\delta_a^2)$，由此得到以下线性方程组：

$$\begin{aligned}
x_{n+1}&=\varphi_1 x_n+\varphi_2 x_{n-1}+\cdots+\varphi_n x_1+a_{n+1}\\
x_{n+2}&=\varphi_1 x_{n+1}+\varphi_2 x_n+\cdots+\varphi_n x_2+a_{n+2}\\
&\cdots\cdots\\
x_N&=\varphi_1 x_{N-1}+\varphi_2 x_{N-2}+\cdots+\varphi_n x_{N-n}+a_N
\end{aligned}$$

可用式(4)所示的矩阵形式来表示上述线性方程组：

$$y=x\varphi+a$$

式中：
$$y=[x_{n+1}x_{n+2}\cdots x_N]^T$$
$$\varphi=[\varphi_1\ \varphi_2\cdots\varphi_n]^T$$
$$a=[a_{n+1}a_{n+2}\cdots a_N]^T$$
$$x=\begin{bmatrix} x_n & x_{n-1} & \cdots & x_1\\ x_{n+1} & x_n & \cdots & x_2\\ & \cdots\cdots & & \\ x_{N-1} & x_{N-2} & \cdots & x_{N-n}\end{bmatrix} \quad (4)$$

根据多元线性回归理论，参数矩阵 φ 的最小二乘估计为：

$$\hat{\varphi}=(x^Tx)^{-1}x^Ty \quad (5)$$

2.3 构造判别函数

根据(5)式可以获得待测序列 $X=\{x_t | t=0,1,2,\cdots,n-1\}$ 的参数模型 φ_x，同样我们也可以得到序列数据库中的其它序列的参数模型 φ_{r_i}。φ_x 和 φ_{r_i} 都是 n 维向量，故均可视为 n 维空间上的点，待检点与哪个参考点近，就表明相应的序列非常相似。从而，把问题归结为实 n 维空间 R^n 中的距离问题，可以采用如下几种距离判别函数[1]。

(1)欧几里德距离 这里，φ_x 表示待检模型，φ_r 表示参考模型，序列的相似性查找问题转化为 φ_x 与 φ_r 的欧几里德距离计算。φ_x 与 φ_r 之间的欧几里德距离：

$$D_E^2(\varphi_x,\varphi_r)=(\varphi_x-\varphi_r)^T(\varphi_x-\varphi_r) \quad (6)$$

如果待检模型与某个参考模型的欧几里德距离最小，则相应的两个序列最相似。欧几里德距离的最大缺陷是未考虑模式向量 φ 中各元素重要性的不同，而将 φ 中的所有 φ_i 均等同对待。为了克服这一缺陷，应将欧几里德距离进行加权处理，加权后欧几里德函数形式为：

$$D_W^2(\varphi_x,\varphi_r)=(\varphi_x-\varphi_r)^TW(\varphi_x-\varphi_r)$$

其中 W 为相应的加权矩阵。

(2)残差偏移距离判别 ARMA 模型的残差向量

a 中包含了时间序列与自回归参数两部分的信息，因此也可以根据 a 来构造距离函数。φ_X 与 φ_Y 之间的残差偏移距离函数为：

$$D_r^2(\varphi_X,\varphi_Y)=N(\varphi_X-\varphi_Y)^T r_x(\varphi_X-\varphi_Y) \tag{7}$$

其中 r_X 是待检序列的协方差矩阵，N 表示待检序列的长度。

(3)Mahalanobis 距离判别

$$D_{Ma}^2(\varphi_X,\varphi_Y)=\frac{N}{\delta_Y^2}(\varphi_X-\varphi_Y)^T r_Y(\varphi_X-\varphi_Y) \tag{8}$$

其中 r_Y 是参考序列的协方差矩阵。

(4)Mann 距离判别

$$D_{Ma}^2(\varphi_Y,\varphi_X)=\frac{N}{\delta_X^2}(\varphi_Y-\varphi_X)^T r_X(\varphi_Y-\varphi_X) \tag{9}$$

其中，r_X 为待检序列的协方差矩阵，δ_X^2 为待测时序的方差。

上面介绍的四个距离函数，均具有明显的几何距离的形式，其实质都是加权的欧几里德距离函数，只不过是形式不同而已。欧几里德距离 D_E^2 的权矩阵是单位矩阵 I，残差偏移距离 D_r^2 的权矩阵是待检序列的协方差矩阵，Mann 距离中还含有待测时序的方差 δ_X^2 这一特性，所以其判别能力较残差距离强[1]。

AR 模型对序列的长度要求并不很苛刻，只要序列足够长，就可以获得相应的参数模型。为了方便比较，对每个序列都提取 AR(n)模型，实质上并不是每个序列都适合 AR(n)模型，这是该方法的缺点，另外利用 ARMA 模型很难实现子序列匹配问题。

3 基于离散傅立叶变换的时间序列相似性快速查找

在时间序列分析方面，离散傅立叶变换具有独特的优点[2]。给定一个时间序列，可以用离散傅立叶变换把其从时域空间变换到频域空间。根据 Parseval 的理论，时域能量函数与频域能量谱函数相同[5~6]，且频域空间的大部分能量集中在前几个系数上，因此可以不考虑离散傅立叶变换得到的其它系数，把这些被保留的系数看作从时间序列上提取的特征，这样从每个序列获得 k 个特征，并进一步把它们映射到 k 维空间上；然后用一些目前被广泛采用的多维索引方法（如 R* 树、k-d-B 树、线性四叉树、格子文件(grid-file)）来存储和检索这些多维空间的点。下面分别描述一下如何进行基于离散傅立叶变换的完全匹配和子序列匹配。

3.1 完全匹配

对于完全匹配来说，被查找的序列与给出的序列有相同的长度。与子序列匹配相比，相对来说比较简单一些，下面给出该方法的数学描述。

(1)特征提取　给定一个时间序列 $X=\{x_t|t=0,1,2,\cdots,n-1\}$，对 X 进行离散傅立叶变换，得到：

$$X_f=1/\sqrt{n}\sum_{t=0}^{n-1}x_t\exp(-i2\pi ft/n)\quad f=0,1,\cdots,n-1 \tag{10}$$

这里，X 与 x_t 代表时域信息，而 $\vec{X}$ 与 X_f 代表频域信息，$\vec{X}=\{X_f|f=0,1,2,\cdots,n-1\}$，$X_f$ 为傅立叶系数。

(2)首次筛选　根据 Parseval 的理论，时域能量谱函数与频域能量谱函数相同。

$$\|X-Y\|^2\equiv\|\vec{X}-\vec{Y}\|^2 \tag{11}$$

衡量两个序列是否相似的一般方法是用欧几里德距离，如果两个序列的欧几里德距离小于 ε 的话，认为这两个序列相似。即满足如下式子：

$$\|X-Y\|^2=\sum_{f=0}^{n-1}|x_t-y_t|^2\leqslant\varepsilon^2 \tag{12}$$

按照 Parseval 的理论(11)式，如下式子也应该成立：

$$\|\vec{X}-\vec{Y}\|^2=\sum_{f=0}^{n-1}|X_f-Y_f|^2\leqslant\varepsilon^2 \tag{13}$$

对大多数序列来说，能量集中在傅立叶变换后的前几个系数，也就是说一个信号的高频部分相对来说并不重要，因此我们只取前面 f_c 个系数[5~6]。显然

$$\sum_{f=0}^{f_c-1}|X_f-Y_f|^2\leqslant\sum_{f=0}^{n-1}|X_f-Y_f|^2\leqslant\varepsilon^2 \tag{14}$$

因此(15)式成立：

$$\sum_{f=0}^{f_c-1}|X_f-Y_f|\leqslant\varepsilon^2 \tag{15}$$

首次筛选所做的工作就是，从提出特征后的频域空间中找出满足(15)式的序列。这样就滤掉一大批与给定序列的距离大于 ε 的序列。

(3)最终验证　按照(15)式，可以滤掉一大批与给定序列的距离大于 ε 的序列。但是由于只考虑了前面几个傅立叶系数，所以并不能保证剩余的序列就相似[5~6]。还需要进行最终验证工作，即计算每个首次被选中的序列与给定序列在时域空间的欧几里德距离，如果两个序列的欧几里德距离小于或等于 ε，则接受该序列。

实践表明，上述方法在完全匹配查找方面非常有效，而且只取1～3个系数就可以达到很好的效果，随着序列的数目和序列长度的增加，其执行效果更好[5~6]。

3.2 子序列匹配

子序列匹配比完全查找要复杂[6]，在 N 个长度不同的序列 $Y_1,Y_2,\cdots,Y_n$ 中找到与给定的查询序列 X 相似的子序列，如果对 $Y_1,Y_2,\cdots,Y_n$ 的任何一个可能匹配的位置都扫描，则其复杂度为 $O(N^2l^2)$（l 为 $Y_1,Y_2,\cdots,Y_n$ 的平均长度），因此设计准确、快速、适合任意查找长度的子序列匹配算法是非常必要的。由于离散傅立叶变换在完全匹配方面非常成功，Christos[6]在

前人工作的基础上，提出了基于离散傅立叶子序列快速匹配的方法。

Christos 没有对整个序列进行特征提取，而是设定了滑动窗口，对滑动窗口内的子序列进行特征提取。滑动窗口的长度依据查找长度而定。在算法的第一步，先定义一个查找长度 $\omega(\omega \geqslant 1)$，$\omega$ 的选定与具体的应用有关。例如，在股票分析中，人们所感兴趣的往往是一周或一个月的模式(时间太短，易受噪声的影响)，所以相应的 ω 为7天或30天。接着，把长度为 ω 的滑动窗口放置在每一个序列上的起始位置，此时滑动窗口对应序列上的长度为 ω 的一段子序列，对这段序列进行傅立叶变换，这样每一个长度为 ω 的子序列对应 f 维特征空间上的一个点[6]，然后滑动窗口向后移，再以序列的第二个点为起始单位，形成另一个长度为 ω 的子序列，依次类推，一共可以得到 $Len(S)-\omega+1$ 个 f 维特征空间上的点。

显然，特征空间上的点组成的数据库远远大于原来的序列数据库，几乎序列上的每个点都要对应 f 维空间上的一个点，这样带来了索引的困难[6]，为了方便计算，可以只取前 f_c 个(通常2～3个就足够了)傅立叶系数，因为能量主要集中在前 f_c 个系数上。每个傅立叶系数都有一个模，f_c 个傅立叶系数就有 f_c 个模，把 f_c 个模映射到 f_c 维空间，这样每个滑动窗口对应的序列就转化为 f_c 维空间上的点。因为相邻的滑动窗口内的序列内容非常相似，所以得到的模的轨迹应该是很平滑的。

为了加快查找速度，把模的轨迹分成几段子轨迹，每一段用最小边界矩形 MBR 来代替，用 R* 树来存储和检索这些 MBR[6]。当提出一个查找子序列请求的时候，首先在 R* 上进行检索，找到包含该子序列的 MBR，从而避免了对整个轨迹的搜索。

如何将模的轨迹转化为 MBR，也就是如何把轨迹分段的问题，一种非常直观的方法是，根据一个事先给定的长度(例如50)来将模的轨迹分段，还可以用一些简单的函数(如 $\sqrt{Len(S)}$)来分段，然而这样得到的分段结果是非常糟糕的。图1、2显示了一个有9个点的轨迹的分段情况，如果按 $\sqrt{Len(S)}$ 来划分轨迹，则分段如图1所示，显然它不如图2。图2中每个 MBR 所包含的点的个数并不固定，而是自适应的[6]。

为得到类似图2所示的自适应分段，Christos 给出了一个贪心算法。具体做法是：以模的轨迹的第一个点和第二个点为基准建立第一个 MBR(此时 MBR 仅包括这两个点)，按式(16)计算出边界代价函数值 mc，然后考虑第三个点，并计算新的边界代价函数值 mc，如果 mc 增大，则开始另外一个 MBR，否则把这个点加入到原来的 MBR 中，继续。Christos 给出的边界代价

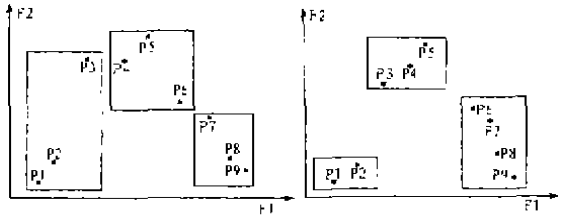

图1 事先固定点个数的分段情况　图2 自适应分段

函数如下：

$$mc = DA(\vec{L})/k \tag{16}$$

其中 $DA(\vec{L}) = \prod_{i=1}^{n}(L_i + 0.5)$，$\vec{L} = (L_1, L_2, \cdots, L_n)$ 表示 MBR 的边。

上面讨论了如何建立索引的问题，下面讨论如何查找。

如果查找长度正好等于 ω 的话，待查找序列 X 被映射为 f_c 维特征空间上的点 X'，查找结果是一个以 X' 为中心，以 ε 为半径的球体。如果待查找序列 X 的长度大于 ω 的话，就比较复杂了[6]，原因是从 R* 树上只能索引到长度等于 ω 的子序列，目前在解决这一问题上采用了两种方法。一种方法是前缀查找(Prefix Search)；另外一种是二次分段查找法，设想 $Len(X)$ 是 ω 的整倍数，把 X 分为 p 段长度为 ω 的子序列，处理每一段子序列，并将子查找结果合并起来。

尽管离散傅立叶变换较好地解决了时间序列的完全匹配与子序列匹配问题，但是该方法并没有考虑序列取值问题，有些情况下两个序列的取值相差很大，而变化趋势却很相似[16]。例如有两种股票的历史数据，一个价格是在10 $ 附近波动，另一种是在75 $ 附近波动，在比较这两个序列是否相似之前，应该适当做一些偏移变换(Offset Translation)和幅度调整(Amplitude Scaling)。

4 基于规范变换的查找方法

针对基于距离的比较方法和基于离散傅立叶变换时间序列查找方法的缺点，Agrawal[16]提出了一种新的方法，他认为在比较序列时应该考虑噪音、幅度和偏移问题。当比较两个序列是否相似时，首先应适当做一些幅度调整和偏移变换，并且还可以忽略一些不匹配的小区域。例如图3所示的两个序列，在序列 S 上有非常小的区域 g1(这个区域相对于序列 X 来说很短，通常称这种区域为 Gap)，如果我们把这部分忽略的话(如图4所示)，而且做相应的偏移变换(如图5所示)与幅度缩放(如图6所示)，则这两个序列很相似。

4.1 基本概念

Agrawal认为如果两个序列有足够多的、不相互

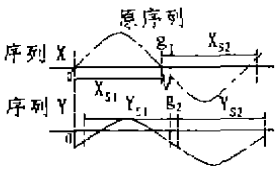

图3 序列 X 与 Y

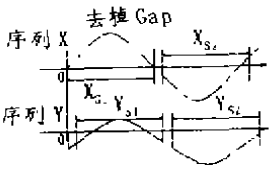

图4 去掉Gap后的序列 X 与 Y

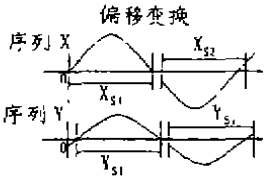

图5 偏移变换后的序列 X 与 Y

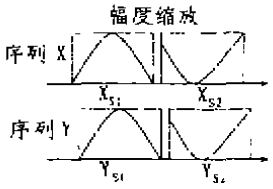

图6 幅度缩放后的序列 X 与 Y

重叠、按时间顺序排列且相似的子序列，则这两个序列相似[16]。下面是他对序列相似概念的形式化描述：

如果序列 X 所包含的不相互重叠的子序列 $X_{S1}\cdots X_{Sm}$ 和与 Y 所包含的不相互重叠的子序列 $Y_{S1}\cdots Y_{Sm}$ 满足如下3个条件，可以认为 X 与 Y 是 ε-相似：

(1)对任意的 $1\leqslant i\leqslant j\leqslant m$，$X_{Si}<X_{Sj}$ 与 $Y_{Si}<Y_{Sj}$ 都成立。

(2)存在一些比例因子 λ 和一些偏移 θ 使得下式成立：

$$\forall_{i=1}^{m}\theta(\lambda(X_{Si}))\approx Y_{Si} \tag{17}$$

其中，$\approx$ 表示两个子序列相似，$\theta(\lambda(X_{Si}))$ 表示对子序列 X_{Si} 以 λ 为比例因子进行缩放，按照 θ 进行偏移变换。

(3)
$$\frac{\sum_{i=1}^{m}Len(X_{Si})+\sum_{i=1}^{m}Len(Y_{Si})}{Len(X)+Len(Y)}\geqslant\xi \tag{18}$$

式(18)是两个序列是否相似的评价函数，如果序列 X 与 Y 匹配的长度之和与这两个序列的长度之和的比值大于 ξ，则认为序列 X 与 Y 是 ξ-相似的，ξ 是事先给定的阈值。

上面是对序列相似的一般性定义，根据具体的应用可以对上述公式进行适当修改。例如当被比较的序列 X 与 Y 的长度非常悬殊时，我们可以用如下函数来评价相似度：

$$\frac{\sum_{i=1}^{m}Len(X_{Si})+\sum_{i=1}^{m}Len(Y_{Si})}{2\times\min(Len(X),Len(Y))}\geqslant\xi \tag{19}$$

4.2 查找方法

Agrawal 把 X 与 Y 的相似性比较问题分为三个子问题：原子序列匹配；窗口缝合；子序列排序。

4.2.1 *原子序列匹配* 与基于离散傅立叶变换的时间序列查找方法相同，Agrawal 也采用了滑动窗口技术，根据用户事先给定的一个 ω(通常为5～20)，将序列映射为若干长度为 ω 的窗口，然后对这些窗口按照(20)式进行标准化，即进行幅度缩放与偏移变换。

$$\tilde{W}[i]=(W[i]-\frac{W_{min}+W_{max}}{2})/\frac{W_{max}-W_{min}}{2} \tag{20}$$

$W[i]$ 表示窗口第 i 个点的值，W_{max}、W_{min} 分别表示窗口内所有点的最大值与最小值，通过(13)式使得窗口中内的每个点的值都落在(-1,-1)之间。把这种标准化后的窗口称为原子，这里用 W 表示窗口，$\tilde{W}$ 表示相应的原子。如果 $\forall i|\tilde{W}_1[i]-\tilde{W}_2[i]|\leqslant\varepsilon$，则可认为原子 $\tilde{W}_1$、$\tilde{W}_2$ 是 ε-相似。

上述是对关于原子匹配的定义，在原子匹配阶段所完成的工作是将所有相似的原子都找出来，为了提高查找速度，把每个原子看作 ω 维空间上的一个点，采用 R^* 来建立索引[13,14,15]。

具体到序列 X，本文用 $X_{w1}\cdots X_{wm}$ 表示从 X 上获得的一系列窗口，用 $\tilde{X}_1\cdots\tilde{X}_m$ 表示相应的原子。

4.2.2 *窗口缝合* 即子序列匹配，其主要任务是将相似的原子连接起来形成比较长的彼此相似的子序列。设 $\tilde{X}_1\cdots\tilde{X}_m$ 和 $\tilde{Y}_1\cdots\tilde{Y}_m$ 分别为 X 与 Y 上 m 个标准化后的原子，且对于任意的 i 都有 $\tilde{X}_i$ 与 $\tilde{Y}_i$ 相似，另外对于任何 $j>i$，$First(X_{wi})\leqslant First(X_{wj})$，$First(Y_{wi})\leqslant First(Y_{wj})$。除此之外，如果还满足下面两个条件，则可分别缝合 $\tilde{X}_1\cdots\tilde{X}_m$ 和 $\tilde{Y}_1\cdots\tilde{Y}_m$ 使它们形成一对相似的子序列：

(1)对任何 $i>1$，如果 $\tilde{X}_i$ 不与 $\tilde{X}_{i-1}$ 重叠，且 $\tilde{X}_i$ 与 $\tilde{X}_{i-1}$ 之间的 Gap 小于等于 γ，同时 Y 也满足这个条件；如果 $\tilde{X}_i$ 与 $\tilde{X}_{i-1}$ 重叠，重叠长度为 d，$\tilde{Y}_i$ 与 $\tilde{Y}_{i-1}$ 也重叠且重叠长度也为 d。

(2)X 上的每个窗口进行标准化时所用的比例因子大致相同，Y 上的每个窗口进行标准化时所用的比例因子也大致相同。

例如有 X 与 Y 两个序列，在原子匹配过程中，得到三个相似的原子对 $(\tilde{X}_1,\tilde{Y}_1)$，$(\tilde{X}_2,\tilde{Y}_2)$，$(\tilde{X}_3,\tilde{Y}_3)$，并且满足上述窗口缝合的条件，因此可以把它们缝合起来得到一对相似的子序列，图7、8、9描述了窗口缝合的过程。图7中，原子 $\tilde{X}_1$ 与 $\tilde{X}_2$ 重叠，重叠长度为 d，$\tilde{Y}_1$ 与 $\tilde{Y}_2$ 也重叠，重叠长度也为 d，满足条件1，同时它们满足窗口缝合的其它条件，可以把 $\tilde{X}_1$ 与 $\tilde{X}_2$ 缝合，$\tilde{Y}_1$ 与 $\tilde{Y}_2$ 缝合。图8中，原子 $\tilde{X}_2$ 与 $\tilde{X}_1$ 不重叠，两者之间有一个长度不大于 γ 的 Gap，$\tilde{Y}_2$ 与 $\tilde{Y}_1$ 也不重叠，两者之间也有一个长度不大于 γ 的 Gap，满足条件1，同时它们满足窗口缝合的其它条件，把 $\tilde{X}_2$ 与 $\tilde{X}_3$ 缝合，$\tilde{Y}_2$ 与 $\tilde{Y}_3$ 缝合。图9表示了窗口缝合过程对相似的原子对 $(\tilde{X}_1,\tilde{Y}_1)$，$(\tilde{X}_2,\tilde{Y}_2)$，

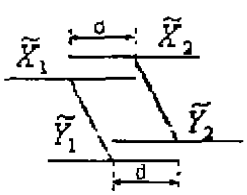

图7 有重叠的情况

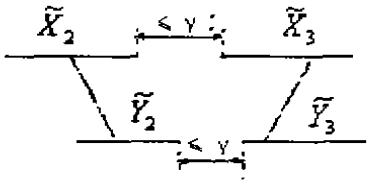

图8 不重叠，Gap 小于等于 γ

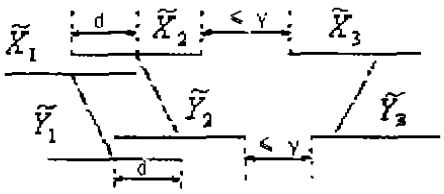

图9 缝合结果

$(\tilde{X}_1,\tilde{Y}_3)$的缝合结果。

窗口缝合技术中考虑了 Gap，这样就把一些噪音数据和两个序列上有差异但在相似性比较时可以忽略的部分过滤掉，这是此方法较其它方法有显著改进的地方[16]。

4.2.3 *子序列排序* 通过对窗口缝合得到一些相似的子序列，再对这些子序列排序，则可以找到两个彼此匹配的序列。子序列排序的主要任务是从没有重叠的子序列匹配中找出匹配得最长的那些序列[16]。如果把所有的相似的原子对看作图论中的顶点，两个窗口的缝合看作两个顶点之间的边的话，那么从起点到终点有多条路径，子序列排序就是寻找最长路径。

经过原子匹配与窗口缝合就找出了相似的子序列，通过对子序列排序完成了序列的相似查找，因此该方法不仅适用于完全匹配，而且适用于子序列匹配。另外，这种方法中过滤掉了一些 Gap，而且对序列作幅度缩放和偏移变换，所以该方法具有良好的鲁棒性，在算法的具体执行中用户可以设定 $\omega,\gamma,\varepsilon$，增加了算法的适用性[16]。

结束语 目前在许多领域都涉及到时间序列相似性查询，除文中所提到的金融决策、股票分析外，还有 DNA 分析以及语音分析、视频流的关键帧提取等多媒体信息处理与查询方面。由于被比较的序列可能来自不同的采样频率、不同的观测环境，所以很难保证不同观测者对两个完全相同的序列得到的观测值一定相同，因此很有必要对序列的相似性查询进行研究。

本文所讨论的三种方法是目前数据挖掘领域中主要采用的时序匹配方法。对于文中所提的 ARMA 模型，要求对待检模型 φ_X 的阶数与所有参考模型 φ_Y 的阶数相同[3]。基于傅立叶变换的时间序列查询方法，不仅适用于完全匹配查找，而且还适用于子序列匹配查找[5,6]。基于规范变换的方法考虑了噪音、幅度和偏移问题，从而大大提高了查询效率，在时间序列的相似性查找中，该方法体现了充分的优越性[16]。

参 考 文 献

1 扬叔子，吴雅. 时间序列分析的工程应用. 华中理工大学出版社，1995

2 安鸿志. 时间序列分析. 华东师范大学出版社，1993

3 Box G E P, et al. 顾岚主译. 时间序列分析预测与控制. 中国统计出版社，1999

4 Agrawal R, et al. Database Mining: A Performance Perspective. IEEE Transaction on knowledge and Data Engineering, Special Issue on Learning and Discovery in Knowledge-Based Database

5 Agrawal R, et al. Efficient similarity search in sequence database. In: FODO Conf. Evanston, Illinois, Oct. 1993

6 Faloutsos C, et al. Fast Subsequence Matching in Time-Series Databases. SIGMOD Conference. 1994. 419～429

7 Raflei D, Mendelzon A. Efficient Retrieval of Similar Time Sequences Using DFT. In. Proc. of the 5th Intl. Conf. on Found of Data Org. And Alg. (FODO'98), Kobe, Japan, November 1998

8 Raflei D, Mendelzon A. Similarity-Based Queries for Time Series Data. In: ACM SIGMOD Conf. on the Management of Data(sigmod'97), May, 1997

9 Oppenheim A V, Schafer R W. Digital Signal Processing. Prentice-Hall, Englewood Cliffs, N. J. , 1975

10 Bloomfifld P. Fourier Analysis of Time Series: An Introduction. Princeton University, 1976

11 Jagadish H V, et al. Similarity-Based Queries. In: Proc. of ACM SGACT-SIGMOD-SIGAR Symposium on Principles of Database System(PODS'95), San Jose, 1995. 36～45

12 Jagadish H V. A retrieval technique for similar shapes. In: ACM SIGMOD Symposium on the Management of Data. 1995. 208～217

13 Sellis T, et al. The R+-Tree: A Dynamic Index For Multi-Dimensional Objects

14 Beckmann N, et al. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In: ACM SIGMOD Conf. on the Management of Data. 1990. 322～331

15 Papadias D, et al. Topological Relations in the World of Minimum Bounding Rectangles: A study with R-trees

16 Agrawal R, et al Fast Similarity Search in the Presence of Nois, Scaling, and Translation in Time-Series Database. VLDB 1995

17 Yi Byoung-Kee, et al. Efficient Retrieval of Similar Time Sequences under Time Warping. In: Proc. of the 14th Interl. Conf. on Data Engineering (ICDE'98). Orlando, 1998. 201～218

18 Aoki M. Application of Computer Aided Times Series Modeling. 1997