文章编号：2096-1472(2016)-08-01-07

基于线性散列索引的时间序列查询方法研究

戴 珂
（大连海事大学，辽宁 大连 116026）

摘 要：随着信息化的发展，大量的数据被产生。在新产生的数据中，时间序列数据是一种重要的数据类型，而对该类数据进行高效的查询处理成为了当前研究的热点。本文针对线性散列的索引机制，提出了一种新型的时间序列的查询处理方法，以降低索引创建时间和提高查询效率。实验证明，本方法中的线性散列索引，在创建时的时间耗费有所下降。在查询阶段采用K近邻与下界距离相结合的方法，能有效地过滤掉多余的结果，提高了时间序列查询处理的效率和精确度。

关键词：时间序列；线性散列；K-近邻；下界距离
中图分类号：TP391 **文献标识码**：A

A Research on the Query Method of Time Series Based on Linear Hash Index

DAI Ke
(*Dalian Maritime University*,*Dalian* 116026,*China*)

Abstract: With the development of information science,more and more data is generated through different applications. Time series is an important data type,and the research on how to query time series data efficiently has drawn more and more attention.This paper proposes a new time series query processing method based on linear hash,aiming to reduce the index construction time and improve the query efficiency.The experiment results show that the index construction time has been reduced to some extent.Through the combined method of the K-nearest neighbor and the lower bounding distance in the query phase,redundant results can be effectively filtered,which improves the efficiency and accuracy of the time series query.

Keywords:time series;linear hash;K-nearest neighbor;lower bounding distance

1 引言(Introduction)

时间序列(Time Series)指同一指标的数值按其先后发生的时间顺序排列而成的数列，它作为时态数据的一种特殊形式出现在许多领域，如金融的股票交易价格、医学的心脑电图、气象的温湿度走势、企业的产销走势等。时间序列的表示是针对时间序列的结构复杂而采取的，将时间序列进行变形的技术；时间序列的索引是针对如何进行高效存储，以及快速查询时间序列的技术。

由于时间序列的数据量大和复杂结构，为表示和索引提出了难题。现有的检索技术，处理大数据时经常会耗费大量的时间。其相似性度量也往往不够准确。国内外研究学者提供了许多相似性度量的技术，但查询的完整和准确程度仍有待提高。

本文结合时间序列分段集成近似表示(Piecewise Aggregate Approximation，PAA)方法[1,2]，提出了基于线性散列作为索引技术来处理时间序列查询的新方法。在时间序列的预处理阶段，提出一种新的规范化方法，很好的保留了时间序列的原始形态。采用线性散列索引机制，对时间序列进行有效灵活的存取，自然地处理存储过程中产生的冲突。

在时间序列的查询阶段，提出了一种下界距离方法。并与K近邻方法相结合，提高了查询的效果，完成了用户对于时间序列的查询需求。

基金项目：教育部新世纪优秀人才支持计划项目(No.NCET-11-0861).

2 相关工作(Related work)

时间序列是随着时间的先后顺序而变化的多维的复杂数据类型。形式上时间序列表示为$X=(x_1,x_2,\ldots,x_n)$，其中元素x_i是点的序列，$x_i=(v_i,t_i)$，其中t_i代表时间，v_i代表时间序列在时刻的值。

国内外研究学者关于查询处理的研究，大致可分成时间序列的表示方法、索引技术和相似性度量研究。

时间序列表示方法有离散小波变换(Discrete Wavelet Transform，DWT)[3,4]和离散傅立叶变换(Discrete Fourier Transform，DFT)[5]、奇异值分解法(Singular Value Decomposition，SVD)[6]、分段线性表示(Piecewise Linear Representation，PLR)[7,8]、符号化近似(Symbolic Approximation，SAX)[9,10]方法等。

时间序列索引技术分基于空间划分的索引和基于数据划分的索引，基于空间的划分有K-D树[11,12]、四叉树[13]、网格文件[14]等，基于数据的划分有R-tree[15]、iSAX-tree[16]和ADS-tree[17]。

时间序列的相似性度量是衡量时间序列相互之间联系的方法。相似性度量是数据挖掘中的一项重要的任务。一般情况下，时间序列的每一种相似性度量方法都能够对应一种或多种时间序列特征表示。例如，经典的时间序列PAA特征表示方法用到了欧式距离的度量方法，离散傅立叶变换通常会用到动态时间弯曲距离的度量方法。不同的特征表示选择不同的度量方法，这要与设计的算法相结合。选择一个适合的度量方法，可以提高算法的性能，提高时间序列查询的查全率与查准率。经典的相似性度量方法主要有编辑距离、欧氏距离、动态时间弯曲距离等。为了进一步提高查询效率，国外学者提出了下界距离的概念，下届距离有LB-Yi距离和LB-Kim距离[18,19]等。

3 时间序列预处理(Preprocessing the time series)

3.1 时间序列规范化

给定某企业近几年的原始交易值数据如图1所示。

图1中，横坐标表示时间(天)，纵坐标表示交易值(元)。从图中可以看出，交易值数据分布相对不均匀，这就给后期建立索引，近似性匹配带来了困难。所谓时间序列的规范化，就是在保留原来的总体趋势的情况下，将原始的时间序列转换至区间内。

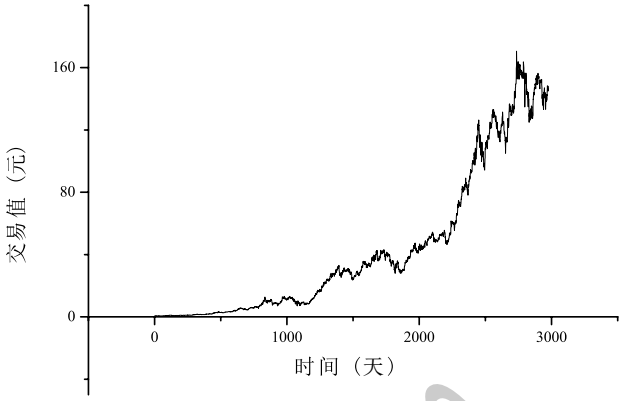

图1 原始数据

Fig.1 Raw data

转换的公式如下：

$$Y=\frac{2y_i-(TS_{min}+TS_{max})}{2(TS_{max}+TS_{min})}*Range$$

其中，y_i为原始序列中时间t_i对应的值，Y_i为进行转换计算后的对应的值。TS_{max}、TS_{min}分别为原始时间序列中的最大值与最小值。$Range$表示要进行规整的范围，$Range=|y_m-y_n|$。

图2代表原始的交易值序列，序列分布在(0,160)，转化后在(−2，2)，如图2所示，其中$y_m=-2$，$y_n=2$。

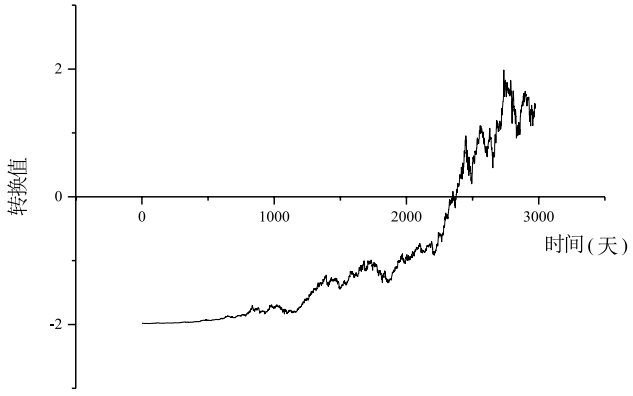

图2 规范化数据

Fig.2 Normalized data

这样原始的时间序列就经过了规范化，与之前相比，规范后的时间序列，完美地保留了原始序列的趋势，数据分布相对比较均匀。

3.2 时间序列分段

通常单条的时间序列规模也较大。如我国某地连续一年的气温数据。假设每小时检测一次，那么连续一年有8760个

数据点，索引全部点在时间和空间上代价都比较大。观察天气预报的规律可知，常常会报道某一天的平均气温，通常一天中温度变化不大(个别地区除外)。就可以用平均气温来代表一天的气温，以24小时对时间序列分段，全年的气温时间序列就分成365段，直接索引这365段要比原始时间序列代价小很多。

定义一个时间序列$X=(x_1,x_2,\cdots,x_n)$，时间序列的集合组成了数据库$Y=\{y_1,y_2,\cdots,y_k\}$，假定Y的时间序列长度为n，将n单位的元素划分成N段数。为了简便，假定N是n的一个因数。

一条长度为n的时间序列X，通过向量表示为$\overline{X}=(\overline{x_1},\overline{x_2},\cdots,\overline{x_n})$，要素$\overline{x_i}$通过以下公式计算得到。

$$\overline{x_i}=\frac{N}{n}\sum_{j=\frac{n}{N}(i-1)+1}^{\frac{N}{n}i}x_j$$

假定只取一天当中的1点到16点的气温数据，组成了一条有16个元素的时间序列。选定分段数为4，这条时间序列被分成4个规整框。分别计算出每个规整框中数据的平均值。向量就成为时间序列分段之后的表示。这种转换将原始时间序列转换成了一个分段的常数近似。当$N=n$和$N=1$时转换后的表示与原序列相同，分段如图3所示。

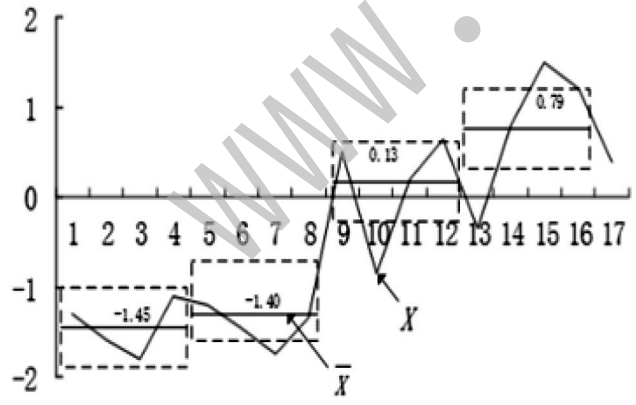

图3 时间序列的分段近似表示

Fig.3 Piecewise approximate representation of time series

3.3 时间序列离散化

大多数的时间序列都符合高斯分布，因而可采用离散化处理。高斯分布如表1所示。

表1 断点分布表

Tab.1 Breakpoint distribution table

β \ a	2	3	4	5	6	7
$\beta 1$	0.00	−0.43	−0.67	−0.84	−0.97	−1.07
$\beta 2$		0.43	0.00	−0.25	−0.43	−0.57
$\beta 3$			0.67	0.25	0.00	−0.18
$\beta 4$				0.84	0.43	0.18
$\beta 5$					0.97	0.57
$\beta 6$						1.07
$\beta 7$						

给定一条标准的符合正态分布的时间序列，决定“断点”使时间序列的目标变量分成若干个区域。表中a表示字母集的大小，即基。选择字母集的大小，并且确定分裂点。将分段累计近似得到的序列的表示确定到所属区间之后，用十进制进行编码。选定字母集的大小a为4，对应断点数为$\beta=3$。从高斯分布表里可以看出，这三个断点分别为$(-0.67,0.00,0.67)$。从图4中可以看出，断点数为3的区域划分情况。划分的区域分别用0、1、2、3表示。原始的时间序列X就会被离散化编码为“0023”。

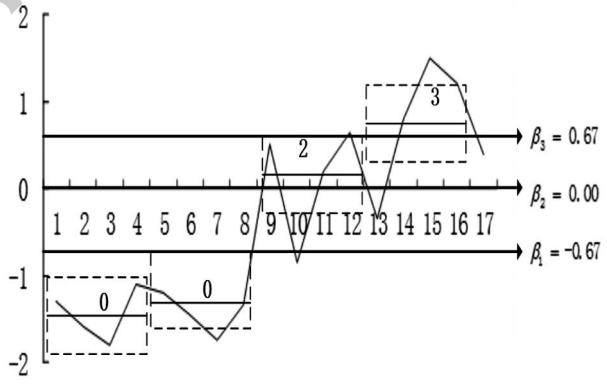

图4 离散表示

Fig.4 Discrete representation

4 线性散列索引(Linear hash index)

4.1 线性散列索引概述

线性散列是一种动态的散列技术，基本思想是利用散列函数，将检索的时间序列的值映射到固定的散列桶号，然后就可以找到待查的时间序列。

通过时间序列的规范化，使时间序列绝大部分都分布在标准正态分布的$(\mu\pm1.96\sigma)$区间中，能解决数据分布的不均匀性。

线性散列轮转分裂机制：定义一个循环级$Level(Level \geq 0, Level \in Z)$，在一个循环级内使用$h_{level}$和$h_{level+1}$这两个散列函数。开始循环后，文件中的桶逐个分裂，一次循环分裂结束以后就开始下一轮的分裂，直至循环结束。有三种类型桶,已被分裂的桶、将要分裂的桶和分裂创建的映像桶。

线性散列索引在执行插入操作时，在相应的桶编号对应的桶存满的情况下就要触发桶的分裂，在这之前增加一个溢出页来存储要插入的时间序列离散化后的表示。此时，要确定哪个编号的桶分裂，根据上文中的轮转分裂机制确定，分裂的方式是依次分裂。即将要发生分裂的桶，也就是在第一个循环级，$Level=0$，$Next=0$的桶。要分裂的桶是以循环分裂的方式进行选择，全部的桶都要进行分裂。

4.2 创建线性散列索引

假定散列表初始桶为N_0，则值$\log N$为时间序列的二进制离散化表示。时间序列二进制离散化表示时的最小位数用d_0表示$d_0=\log N$。

用$Level$表示当前循环级数，则每轮的桶数$N=N_0*2^{Level}$。第1轮$Level=0$，初始桶为$N=N_0$。hash桶依次按编号分裂，用$Next$指示。

每次发生分裂的桶总由$Next$决定，为处理溢出情况，可以引入溢出页来解决。线性散列索引文件初始创建的形态如图5所示。

分裂点	桶编号	主存储页			
Next=0	00	2300	1320	1212	3232
	01	2201	2013	3201	
	10	2002	2102	1230	2130
	11	0023	1003	2311	

图5 初始桶示例

Fig.5 Example of initial hash bucket

此时，$Level=0$，$d_0=2$，$Next=0$，$N_0=4$，桶编号采用两位表示，分别为00、01、10、11，此时分裂点指向00桶，图5所示是已经插入一部分数据时的状态。

4.3 插入与删除线性散列索引

线性散列索引在插入时，首先判断时间序列符号化表示所对应的桶能否触发分裂，如果条件成立，则发生分裂，产生映像桶和溢出页。

插入$h(r)$=2321=100100010001，对应01编号，该桶未满，不发生分裂，直接插入。插入$h(r)$=1330=10100110010，对应10编号，该桶已存满，在插入“1330”时$Next$指向的00桶发生分裂，产生映像桶，映像桶的编号为$Next+N0$=4=100，同时，10桶产生溢出页暂存“1330”，00桶里的数据在00桶和它的映像桶100桶之间进行重新分布。

分裂点	桶编号(3位)	桶编号(2位)	主存储页				溢出页			
	000	00		1320		3232				
	001	01	2201		3201	2321				
Next=2	010	10	2002	2102	1230	2130	1330			
	011	11	0023	1003	2311	0211	2031			
	100	00	2300		1212					
	101	01	2013							

图6 插入“0211”“2031”示例

Fig.6 Example of inserting "0211" and "2031"

插入$h(r)$=0211=11010011，取二进制的最后两位11，对应11编号的桶，该桶未满，不发生分裂，直接插入。插入$h(r)$=2031=11111101111，取二进制的最后两位11，对应11编号的桶，该桶在继插入“0211”之后已经存满，所以在插入“2031”时$Next$指向的01桶就需要进行分裂，产生映像桶，映像桶的编号为$Next+N0$=5=101，同时，11桶产生溢出页暂存“2031”，11桶和它的映像桶101桶之间进行数据重新分布，操作完成之后的数据分布如图6所示。$Next$下移一位，$Next$=2。

5 时间序列的查询 (Querying the time series)

5.1 近似查询

数据挖掘应用需要近似查询，线性散列索引能够支持快速近似查询，由于两个相似的时间序列的符号化表示往往会相同，一次访问就可实现。从线性散列索引文件中查询所要结果的时间序列表示，顺序扫描相应的时间序列作为近似查询结果。

如给定时间序列，经过序列转换后表示为“3102”，$h(r)$=3102=110000011110，取后两位“10”位于$Next$=4和N=4之间，所以对应的桶已经发生分裂，此时取后三位“110”，定位相应110桶，查找到“3102”。返回查询的结果BSF(Best-So-Far)，它是一个粗略的结果集，叫做输入实例时间序列的近似查询。采用一种近似查询和KNN(K-Nearest Neighbor)查询组合的方式来缩小查询空间，以提高

查询效率和查询精度。

基于线性散列索引的精确查询算法，将近似查询得到的结果(BSF)作为输入。因为BSF结果里的时间序列之间的距离相对较小，给最近邻查询创造条件，在近似查询阶段将大部分的搜索空间进行修剪，既提高了查询精度，也降低了查询时间。

K近邻的核心在于找到实例时间序列的邻居，也就是找到与目标序列相邻的时间序列。衡量两条时间序列是不是邻居的判定标准，可直观地理解为两条时间序列之间的距离，如果距离在可接受的范围，就可判定这两条时间序列是邻居。因为特征空间中两条时间序列的距离能反映出两条时间序列之间的相似程度。

K近邻查询将近似查询得到的结果作为K近邻中的数据集，当输入新的时间序列时，在时间序列数据集中找到与目标时间序列最近邻的K条邻居，可认为这K条时间序列与目标时间序列最为相似。当K取1时，查询到达了精确查询。

输入：BSF结果集，大小为$\mathrm{S}(\overline{X_q})$，$S=\{\overline{X_1},\overline{X_2},\ldots,\overline{X_N}\}$其中，$\overline{X_q}$为时间序列的特征向量，目标时间序列特征向量。

输出：目标时间序列特征向量$\overline{X_q}$的邻居。

根据距离度量方法，在时间序列数据集(BSF结果集)S中找出与$\overline{X_q}$最临近的K条时间序列，涵盖这K条时间序列的$\overline{X_q}$的邻域记做$S_k(\overline{X_q})$。相应的K近邻法的模型对应的特征空间划分如图7所示。

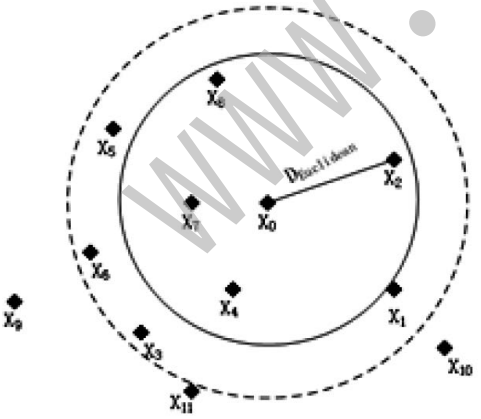

图7 X的K近邻

Fig.7 The K-NN of X

假定特征空间所有的实例点组成了近似查询结果集BSF，大小为$S(\overline{X_q})$。给定一个查询实例时间序列X_0，通过近似查询得到X_0粗略的结果集BSF，再从已有的BSF结果集中尽可能的剔除与X_0不相近的时间序列，得到比较精确的结果集。两条时间序列之间的距离表示它们之间的相似程度，有多种距离的计算方法，通常使用欧式距离、L_p距离和Minkowski距离。假设X_i和X_j是特征空间里的两条时间序列，它们之间的L_p距离为：

$$L_p(X_i,X_j)=\left(\sum_{l=1}^{n}\left|X_i^{(l)}-X_j^{(l)}\right|^p\right)^{\frac{1}{p}} \tag{1}$$

p取值不同表示不同的距离度量方法，如$p\geqslant 1$，当$p=2$时，称为欧式距离(Euclidean distance)，即：

$$L_2(X_i,X_j)=\left(\sum_{l=1}^{n}\left|X_i^{(l)}-X_j^{(l)}\right|^2\right)^{\frac{1}{2}} \tag{2}$$

当$p=1$时，称为曼哈顿距离(Manhattan distance)，即：

$$L_1(X_i,X_j)=\sum_{l=1}^{n}\left|X_i^{(l)}-X_j^{(l)}\right| \tag{3}$$

当p为无穷大时，表示各个坐标距离的最大值，即：

$$L_\infty(X_i,X_j)=\max_l\left|X_i^{(l)}-X_j^{(l)}\right| \tag{4}$$

如图7所示，要得出X_0的四个邻居就要分别计算X_0与特征空间中的时间序列的Euclidean距离，记作D_E，计算X_0与各点之间的距离取最小的四个。经过计算得出X_0的邻居为$\{X_2, X_4, X_7, X_8\}$。

$$\min_4\{D_E(X_0,X_1),D_E(X_0,X_2),\cdots,D_E(X_0,X_N)\}$$

当$K=1$时，说明要返回X_0的一个邻居，经过计算得出X_0的最近邻是X_7，即输入X_0的精确查询。

5.2 紧密性讨论

不直接取顺序遍历BSF结果集是因为时间耗费仍然比较大，用加入界限紧密性TLB(Tightness of Lower Bound)的方式来过滤掉一部分结果。所谓的TLB是一个对相似性度量非常有意义的方法，它的表达式如公式(5)所示。

$$\mathrm{TLB}=\frac{\mathrm{LowerBoundDist(T,S)}}{\mathrm{TrueEuclideanDist(T,S)}} \tag{5}$$

其中，T和S是两条时间序列。TLB的优点是B实现了完全的自由测量，可对索引的有效性进行有效的预测。如果TLB的值

为0，则证明这个索引需要从磁盘顺序读出时间序列，索引不具有高效性。如果TLB的值为1，则证明对索引稍微调整一下就可检索出需要的一个时间序列，并且能够保证得到真实的最近邻。

给定时间序列T和S，长度都为n。假定T为查询实例时间序列，S为待查序列，设置一个规整窗口R，恰好能把T分成$w(w=1,2,\cdots,w)$块，如图11所示，当规整窗口R向指示的方向移动时T被分成了w块区域，每个R就是一个区域，在每个区域里都会存在一个最大值，将每一块的最大值，组成一条序列，取这条序列中最大值的均值，记作$AT_{\mathrm{Max}R}$；最小值组成一条序列，取这条序列中最小值的均值，记作$AT_{\mathrm{Min}R}$。它们的定义如公式(6)所示。

$$\begin{cases} AT_{\mathrm{Max}R}=\dfrac{\sum_{i=1}^{w} T_{\mathrm{Max}R_i}}{w} \\ AT_{\mathrm{Min}R}=\dfrac{\sum_{i=1}^{w} T_{\mathrm{Min}R_i}}{w} \end{cases} \tag{6}$$

时间序列T用如图8所示的两条虚线表示，T与S之间的下界距离分为两部分：高于S部分，这些点的距离和，记为LB_{upper}；低于S部分，这些点的距离和，记为LB_{lower}。下界距离为两部分之和，如公式(7)所示。

$$\mathrm{LowerBoundDist}(T,S)=LB_{\mathrm{upper}}+LB_{\mathrm{lower}} \tag{7}$$

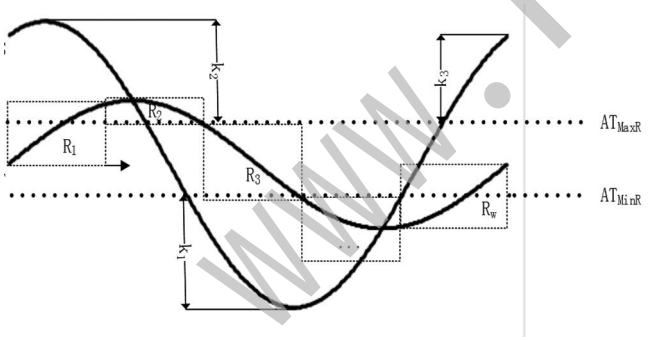

图8 下界距离

Fig.8 Lower bound distance

6 实验评估(Experiments evaluation)

实验环境：Intel(R) Core(TM) i3-4370 @ 3.8GHz，Windows 7操作系统，实验程序采用Java语言编写。

数据集来源：某企业股票交易值如图9所示，取长度分别为[480,960,1440,1920,2400]，时间序列的分段数大小设置为[8,16,24,32,40,48,56,64]。

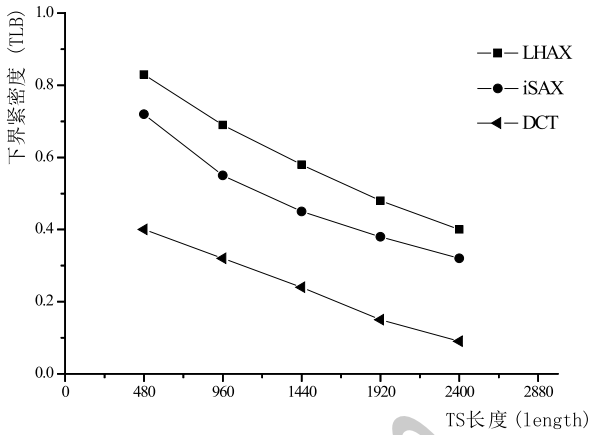

图9 TS长度影响

Fig.9 The influence of length

在进行离散化表示时选取的基数分别为[2,4,6,8,10]。

(1)选取不同的基数对下界紧密度(TLB)的影响。如图9所示，固定时间序列长度为480时，基数对TLB的影响是非线性的，当基数变大时，TLB增长，并且增长速度越来越快，造成这种现象的原因是，在进行离散化表示时，选取的基数越大，相应的断点数就越大，这样时间序列离散越细化，离散化表示之后的时间序列越接近于原始时间序列，下界距离与真实欧式距离的比值就越接近于1。

(2)选取不同的时间序列长度对下界紧密度(TLB)的影响。如图10所示，固定基数大小为10，之所以选取基数为10是因为从上图中可以看出当基数是10的时候，TLB的值相比更接近于1，这样实验误差更小一点，从图中可以看出，当时间序列的长度增大的时候，下界紧密度越来越小。在设置的规整窗口大小不变的情况下，随着时间序列的增大，规整窗口向右滑动时造成的误差也是越来越大的，这势必造成下界距离可靠性越来越差，TLB的值就会越来越小。

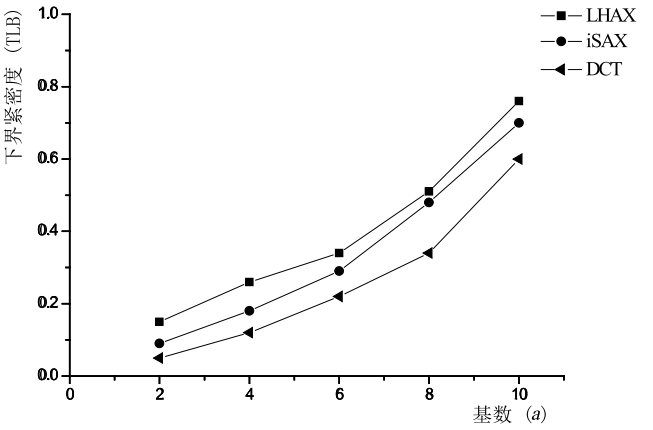

图10 基数影响

Fig.10 The influence of cardinality

(3)选取不同的分段大小对时间序列表示耗时的影响。如图11所示，固定时间序列的长度为480，随着分段大小的不断变大，系统耗时越来越小，当分段大小在40左右的时候，耗时在770ms处下降开始显得不明显，这主要是因为在分段之前有一步对时间序列进行规范化的过程，也可以理解成规范化一条长度为480的时间序列用时是770ms左右。

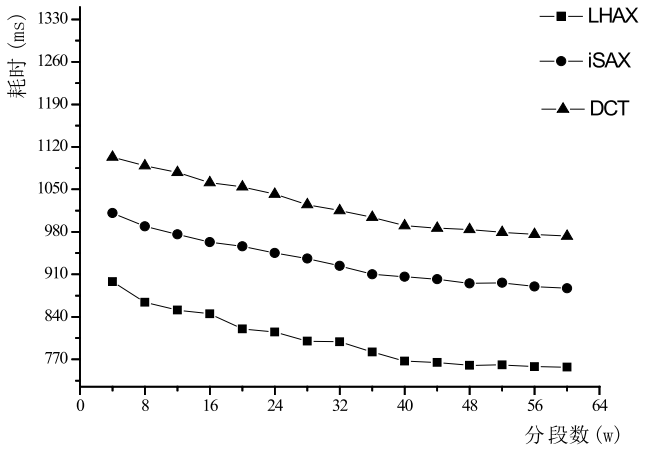

图11 分段数影响

Fig.11 The influence of the piecewise

(4)选定时间序列长度为480，基数为10，分段大小选择40。对系统的完整耗时进行测试，选择对比实验，结果如图12所示。对比实验主要分索引创建时间和查询时间两部分，其中索引创建时间是前期时间序列规范化表示，创建索引的时间总和表示。从图12中可以看出，无论哪种方法，索引创建和预处理耗时占完整耗时中的大部分，而查询时间占完整耗时中的小部分。在索引创建方面的性能提升与iSAX相比不明显；主要是查询方面的提升，从而验证了下界距离的有效性。

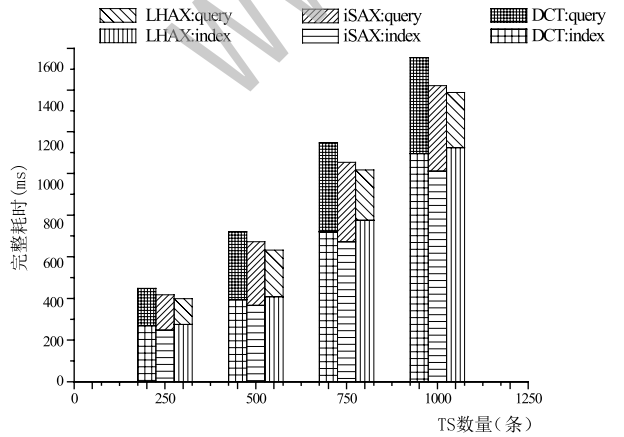

图12 完整耗时

Fig.12 Total time consuming

7 结论(Conclusion)

本文基于已有的时间序列表示方法上，提出了关于时间序列规范化方法，并尝试使用了线性散列创建时间序列的索引，在时间序列的相似性查询方面提出了一种新的下界距离来衡量时间序列之间的相似度。经过实验得出了时间序列规范化方法的有效性，验证了线性散列在索引时间序列时，索引创建时间并没有很大的提升，但是降低了时间序列查询的时间。对提出的下界距离方法做了讨论，效果比较理想。

参考文献(References)

[1] Keogh E,et al.Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases[J].Knowledge & Information Systems,2001,3(3):263-286.

[2] 刘芬,郭躬德.基于符号化聚合近似的时间序列相似性复合度量方法[J].计算机应用, 2013,33(01):192-198.

[3] Chan K P,Fu W C.Efficient Time Series Matching by Wavelets[C].IEEE International Conference on Data Engineering.IEEE,1999:126-133.

[4] Zhou H A,Wang X M,Mei Y L.Theoretical Analysis of the Sound Absorption Characteristics of Periodically Stiffened Micro-perforated Plates[J].Acta Mechanica Sinica,2014,30(5):714-726.

[5] Agrawal R,Faloutsos C,Swami A.Efficient Similarity Search in Sequence Databases[C].International Conference on Foundations of Data Organization and Algorithms.Springer-Verlag,1993:69-84.

[6] Korn F,Jagadish H V,Faloutsos C.Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequences[J].Acm Sigmod Record,1999,26(2):289-300.

[7] Pavlidis T,Horowitz S L.Segmentation of Plane Curves[J].IEEE Transactions on Computers,1974,C-23(8):860-870.

[8] 喻高瞻,等.时间序列数据的分段线性表示[J].计算机应用与软件,2007,24(12):17-18.

[9] Lin J,et al.Experiencing SAX:A Novel Symbolic Representation of Time Series[J].Data Mining & Knowledge Discovery,2007,15(2):107-144.

[10] 李桂玲,等.基于SAX的时间序列相似性度量方法[J].计算机应用研究,2012,29(3):893-896.

[11] Ooi B C,McDonell K J,Sacks-Davis R.Spatialkd-tree:An

Indexing Mechanism for Spatial Databases[C].IEEE COMPSAC,1987,87:85.

[12] 黄河,史忠植,郑征.基于形状特征k-d树的多维时间序列相似搜索[J].软件学报,2006,17(10):2048-2056.

[13] Tayeb J,Ulusoy Ö,Wolfson O.A Quadtree-based Dynamic Attribute Indexing Method[J].The Computer Journal,1998,41(3):185-200.

[14] Hinrichs K,Nievergelt J.The Grid File:A Data Structure Designed to Support Proximity Queries on Spatial Objects[R].Institut Fuer Informatik Zurich (SWITZERLAND),1983.

[15] Guttman A.R-trees:A Dynamic Index Structure for Spatial Searching[M].ACM,1984.

[16] Shieh J,Keogh E.I SAX:Indexing and Mining Terabyte Sized Time Series[C].ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.ACM,2008:623-631.

[17] Zoumpatianos K,Idreos S,Palpanas T.Indexing for Interactive Exploration of Big Data Series[C].Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data.ACM,2014:1555-1566.

[18] Yi B K,Jagadish H V,Faloutsos C.Efficient Retrieval of Similar Time Sequences under Time Warping[C].Proceedings of the 14th International Conference on Data Engineering.IEEE,1998:201-208.

[19] Kim S W,Park S,Chu W W.An Index-based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases[C].Proceedings of the 17th International Conference on Data Engineering.IEEE,2001:607-614.

作者简介：

戴　珂(1957-)，男，本科，工程师.研究领域：软件工程，信息检索.

(上接第18页)

同，图片分辨率不同等条件对于年龄估计的正确率有重大影响，其中侧脸对于正确率影响最大。

参考文献(References)

[1] D Zhang,JJP Tsai.Machine Learning and Software Engineering[J].Software Quality Journal,2014,11(2):87-119.

[2] BK Natarajan.Machine Learning[J].Machine Learning,2014:207-214.

[3] MI Jordan,TM Mitchell.Machine Learning:Trends,Perspectives,and Prospects[J].Science,2015,349(6245):255-260.

[4] 张建明,等.深度学习的研究与发展[J].江苏大学学报:自然科学版,2015,36(2):191-200.

[5] Abel-Rahnan E M,et al.Detecting Sirex Noctilio Grey-attacked and Lightning-struck Pine Trees Using Airborne Hyperspectral Data,Random Forest and Support Vector Machines Classifiers[J].IS-PRS Journal of Photogrammetry and Remote Sensing,2014,88:48-59.

[6] 孙志军,等.深度学习研究综述[J].计算机应用研究,2012,29(8):2806-2810.

[7] Krizhevsky A,Sutskever I,Hinton G E.Image Net Classification with Deep Concolutional Neural Networks[C].NIPS,2012,1(2):4.

[8] 叶浪.基于卷积神经网络的人脸识别研究[D].东南大学,2015.

[9] 孙宁.人脸检测综述[J].电路与系统学报,2006,11(6):104-108.

[10] 闫陈静.人脸年龄估计算法的设计与实现[D].北京交通大学,2016.

[11] Bengio Y,et al.Greedy Layerwise Training of Deep Networks[C].Proceedings of 20th Annual Conference on Neural Information Processing Systems.Vancouver:Neural Information Processing System Foundation,2007:153-160.

作者简介：

曹　橘(1995-)，男，本科生.研究领域：软件开发.

莫雅婷(1994-)，女，本科生.研究领域：软件开发.

黄　晨(1995-)，男，本科生.研究领域：软件开发.

魏子涵(1994-)，女，本科生.研究领域：软件开发.