一种基于关键点的时间序列聚类算法

谢福鼎¹ 李迎² 孙岩² 张永²

(辽宁师范大学城市与环境学院 大连 116029)¹ (辽宁师范大学计算机与信息技术学院 大连 116081)²

摘要 基于关键点技术,提出了一种新的时间序列聚类方法。算法首先寻找时间序列的关键点,将关键点序列进行等维处理后,通过计算关键点序列的相似性构造复杂网络,最后通过复杂网络的社团划分,实现时间序列的聚类。实验结果表明,在时间序列聚类过程中,本方法不仅可以有效降低时间序列的维数,加快聚类的速度,而且可以得到理想的聚类结果。

关键词 时间序列,降维,关键点,复杂网络,聚类

中图分类号 TP311 **文献标识码** A

Cluster Algorithm for Time Series Based on Key Points

XIE Fu-ding¹ LI Ying² SUN Yan² ZHANG Yong²

(School of Urban and Environmental Science, Liaoning Normal University, Dalian 116029, China)¹

(Department of Computer and Science Technology, Liaoning Normal University, Dalian 116081, China)²

Abstract Based on key point technology, a new method for time series cluster was proposed. The key points for each time series were first found, and then the complex network was constructed by calculating the similarity between key point series after they were equidimensional. At last, the clustering time series were implemented by partitioning the complex network into communities. The experimental results show that the dimensions of time series and the consumption of computing time can be effectively reduced by the proposal. Furthermore, the desired cluster result is obtained when applying this method to cluster some practical data.

Keywords Time series, Reduction dimension, Key point, Complex network, Cluster

1 引言

时间序列是一类重要的数据对象,在经济、气象等许多领域都大量存在。对这些数据进行分析,可以揭示事物变化和发展的规律,为科学决策提供依据。如何分析和处理这些海量的时间序列数据,并从中发现一些事先未知的、有价值的信息,正受到越来越多研究者的关注和重视。由于这些海量数据序列具有短期波动频繁、大量噪声干扰以及非稳态等特点,因此直接在原始时间序列上进行相似性查询^[1,2]、分类和聚类^[3]、模式挖掘^[4,5]等工作,不但会造成存储和计算效率低下,而且会影响算法的准确性和可靠性,从而难以获得满意的结果。近年来,时间序列的数据挖掘、知识发现、预测、相似性搜索等问题已经成为一个研究热点,主要研究工作包括降维、特征提取、相似性度量、相似性搜索等。

关键点选择算法 KPSegmentation(key points segmentation)^[6]是一种有效的时间序列降维方法。该方法将时间序列中重要的信息点全部提取出来,有效地去除了时间序列中的噪声与重要性小的数据点。这样不仅加快了对时间序列处理的速度,节省了处理时间与存储空间,而且处理问题的准确率还有所提高。但时间序列降维之后,也带来了另外一些问

题,如关键点提取之后,时间序列由原来的等长序列变成了不等长序列,在度量时间序列相似性时则受到了约束,因为此时无法利用计算等长序列相似性与聚类的方法。

本文提出了一种新的基于关键点技术的时间序列聚类方法,它对降维后得到的不等维关键点序列进行局部等维处理,能够实现非等长关键点序列的相似性度量与聚类。算法首先将时间序列利用关键点提取方法进行降维,利用本文方法将不等维的关键点序列进行局部等维,再用欧氏距离进行相似性度量。其次,利用所得到的相似度矩阵构造复杂网络;最后利用网络的划分结果,实现时间序列的聚类。本文通过两组实验,证明此方法在序列降维、相似性度量和聚类准确性上都有比较好的效果。此方法的另外一个优点是:在关键点序列局部等维之后,序列的相似性度量和聚类不再受距离公式和聚类方法的约束,拓展了基于关键点技术的时间序列聚类方法。比较试验结果说明,本文方法在计算准确率和空间存储上也有较大的改善。

2 相关知识

2.1 关键点提取

作为时间序列趋势变化的分界点,关键点对数据序列的

到稿日期:2011-04-11 返修日期:2011-06-30 本文受国家自然科学基金(10771092),辽宁省博士启动基金(20081079)资助。

谢福鼎(1965-),男,博士,教授,CCF会员,主要研究方向为人工智能、数据挖掘;李迎(1986-),女,硕士,主要研究方向为人工智能、数据挖掘;孙岩(1974-),女,博士,副教授,主要研究方向为数据挖掘;张永(1975-),男,博士,副教授,主要研究方向为智能计算、可信计算。

变化形态影响很大。人们普遍认为,找到序列的关键点的同时也找到了数据序列中的主要变化特征,可以用关键点的线性拟合表示数据序列,这样既保证了数据序列形态基本不变,又降低了对数据存储的要求,提高了计算效率。为了能够在较高压缩率的情况下更好地线性拟合数据序列,KPSegmentation算法将极值法与夹角法相结合,实现优势互补。关键点提取是一种很好的数据降维方法,在时间序列挖掘处理中发挥了很重要的作用。

KPSegmentation方法^[6]利用中线距离阈值和非单调序列中的极值点保持时间段阈值两个约束条件,仅需一次扫描序列数据集,就可以保留非单调序列中重要的极值点,并根据单调序列中3个连续数据点形成的三角形中线长度判断中间数据点是否为需要保留的关键点。此方法极大地减少了数据存储量,提高了计算速度。

定理 1^[6] 在由3个顺序数据点 x_{i-1}, x_i, x_{i+1} 构成的最小序列模式中, x_i 为转折点的可能性与 $\left| x_i - \frac{x_{i+1} + x_{i-1}}{2} \right|$ 相关; $\left| x_i - \frac{x_{i+1} + x_{i-1}}{2} \right|$ 越小, x_i 为转折点的可能性越小;反之成立。

2.2 相似性度量

DTW(data time warping)方法^[3,7]可以计算非等长时间序列的相似性,允许时间序列在时间轴上扭曲,鲁棒性比较好。但是DTW算法的计算复杂度高,计算时间长。

DTW公式^[3]如下:

设时间序列 $Q = \{q_1, q_2, \dots, q_n\}, C = \{c_1, c_2, \dots, c_m\}$,其长度分别为 n 和 m 。它们之间的DTW距离定义为

$$D_{tw}(\langle \rangle, \langle \rangle) = 0; D_{tw}(Q, \langle \rangle) = D_{tw}(\langle \rangle, C) = \infty$$

$$D_{tw}(Q, C) = D_{use}(head(Q), head(C)) + \min\{D_{tw}(Q, rest(C)), D_{tw}(rest(Q), C), D_{tw}(rest(Q), rest(C))\}$$

其中, $\langle \rangle$ 为空序列。

欧氏距离(Euclidean distance)是计算相似性时最常用的一种度量方法,具有运算速度快、复杂度低的优点。因此本文在实现了关键点序列等长之后,采用欧氏距离来计算关键点序列之间的相似性。对于 n 维空间的两个点,它们之间的欧氏距离为

$$d = \sqrt{(x_{11} - x_{21})^2 + (x_{12} - x_{22})^2 + \dots + (x_{1n} - x_{2n})^2}$$

3 基于关键点的时间序列聚类算法

时间序列的高维和高噪音的特点,为时间序列聚类带来了许多困难。关键点算法找到了数据序列中的主要变化特征,并用关键点的线性拟合表示数据序列,较好地解决了这一问题。

设 $X_1 = \langle (t_1, x_1), (t_2, x_2), \dots, (t_i, x_i), \dots, (t_k, x_k) \rangle, X_2 = \langle (t_1, x_1), (t_2, x_2), \dots, (t_i, x_i), \dots, (t_k, x_k) \rangle$ 是等长的两条时间序列,经过关键点提取之后得到的两个关键点序列分别为

$$X_1' = \langle (t_{m1}, x_{m1}), (t_{m2}, x_{m2}), \dots, (t_{ms}, x_{ms}) \rangle$$

$$X_2' = \langle (t_{n1}, x_{n1}), (t_{n2}, x_{n2}), \dots, (t_{np}, x_{np}) \rangle$$

其中, $t_{m1} = t_{n1} = t_1, t_{ms} = t_{np} = t_k$ 。一般来说,它们是非等长的序列。显然,可以用DTW算法计算它们之间的相似度,但计算复杂度较高。

为了提高计算速度,将 X_1', X_2' 变为等长序。将 X_1', X_2' 的时间点集合 $\{t_{mi}\}$ 和 $\{t_{nj}\}$ 合并为一个有序的集合 T ,即取两条时间序列的时间点的并集,将关键点序列重新对照原时间序列取点,这样就可以得到两个新的等长的关键点序列为

$$X_1'' = \langle (t_{h1}, x_{h1}), (t_{h2}, x_{h2}), \dots, (t_{hs}, x_{hs}) \rangle$$

$$X_2'' = \langle (t_{h1}, x_{h1}), (t_{h2}, x_{h2}), \dots, (t_{hs}, x_{hs}) \rangle$$

一般地,新序列的维数远远小于原始时间数据序列维数,这同样达到了降维的目的。对于所得到的等长序列,这里采用欧式距离来计算它们之间的相似度。基于相似度的结果,给定合适阈值,构建Laplace矩阵,利用复杂网络的谱平分方法^[13]进行社团划分聚类,也可以直接在相似度结果的基础上采用FCM聚类方法对序列进行聚类。在本文的实验中,两种聚类结果一致,因此构建复杂网络的方法不再详述,下面的算法描述采用简单的FCM聚类。

算法描述如下:

输入:时间序列数据集 $TS = \langle X_1, X_2, \dots, X_n \rangle$;

输出:聚类结果;

- Step1 对 TS 中的每个 X_i 进行关键点提取,得到关键点序列 X_i' ;
- Step2 对每一对关键点序列 X_i' 和 X_j' ,取两条序列时间点的并集进行等长处理,得到等长序列 X_i'' 和 X_j'' ;
- Step3 采用欧式距离计算 X_i'' 和 X_j'' 相似度 S_{ij} ;
- Step4 调用FCM算法进行聚类,并输出结果。

该算法的优点在于,在这里进行的是局部等维,即在两个关键点序列之间进行等维计算,不是全部的关键点序列进行等维。这不但达到了降维的目的,而且在序列等长之后,可以选择熟悉的相似度计算公式,进行序列间相似度的计算,从而拓展了降维后计算不等长序列相似度的空间,也为时间序列预处理提出了新的解决方法,为相似性度量与聚类提供了新的扩展与延伸。实验结果证明了本文的方法在基于关键点序列的聚类方法上是可行的,在相似性度量准确性与聚类准确性上也有较好的提高。

4 方法的应用

本文采用Isolet5数据库和Synthetic Control数据集两组数据进行了实验。实验环境为2.66GHz CPU Pentium® 4 PC机,1G内存,操作系统为Windows XP Professional。算法实现软环境为matlab 7.0和VC++6.0。

4.1 实验1

数据来自Isolet5数据库(<http://www.ics.uci.edu/~mllearn/MLRepository.html>),该数据库具有维数较高的特点。Isolet5数据库包含了1559个序列,每个序列有617个属性,所有的序列被分成26类。

我们利用本文方法对Isolet5数据库中的1559条序列进行聚类,结果如表1所列。结果表明,通过适当的参数选取,聚类精度可以达到82.94%,关键点序列最长为77维,等长后的关键点序列长度最多为138维,压缩率为87.52%。从表1可以看出,在压缩率为90.28%时,准确率为82.55%,可以得到比较满意的结果。在压缩率很高的情况下(92.71%),聚类精度有所下降(75.18%),即压缩后所得到的关键点序列失去了较多的原始序列的信息。实验说明,我们不但可以进行降维处理,且可以得到理想的聚类结果(表1中 c 和 ϵ 为关键点提取算法中的2个参数)。

表1 Isolet5中序列的聚类结果

数据库	参数(c,ε)	准确率	关键点序列的长度	等长后点序列的长度
	(7,4)	82.94%	43~77	98~138
Isolet5	(8,4)	82.55%	36~60	71~104
	(10,6)	75.18%	18~45	50~77

为了说明本文方法的有效性并和其它方法得到的结果相比较,我们又选择了 Isolet5 数据库中的第 13 类和第 14 类的数据进行聚类。其中第 13 类包含 59 个序列,第 14 类包含 60 个序列,结果如表 2 所列。其中,方法 1 是对数据直接进行 K-means^[10] 聚类;方法 2 是对数据序列利用自底向上分段算法^[11,12] 进行分段,转换成(-1,0,1)构成的齐序列后,进行 K-means 聚类;方法 3 利用 EMD 方法提取数据序列的趋势,利用自底向上分段算法对趋势序列进行分段,转换成由(-1,0,1)构成的齐序列后,进行 K-means 聚类^[8];方法 4 为本文方法。表 2 说明,利用本文方法降维效果更加明显,且误分的序列个数较少,即聚类效果较好。

表2 4种不同方法的结果比较

数据库	Isolet5 数据库中的第 13 类和第 14 类				
方法	方法 1	方法 2	方法 3	方法 4	
第一类	聚类后第一类包含元素个数	65	64	54	43
	真正属于第一类的元素个数	24	38	36	43
	由第二类误分来的元素个数	41	26	18	0
第二类	聚类后第二类包含元素个数	54	55	65	76
	真正属于第二类的元素个数	19	34	42	60
	由第一类误分来的元素个数	35	21	23	16
	参加聚类的序列长度	617	287	279	64~79

由表 2 中的结果,我们计算了 4 种不同方法的平均查准率和查全率,结果如表 3 所列。

表3 4种方法的平均查准率和查全率

	Isolet5(第 13 类与第 14 类)			
	方法 1	方法 2	方法 3	方法 4
平均查准率	0.3605	0.6060	0.6564	0.8948
平均查全率	0.3617	0.6054	0.6551	0.8644

实验 1 的结果表明,本文提出的基于关键点技术的时间序列挖掘方法,不但能够实现维度的约减,而且聚类精度也有所提高。

4.2 实验 2

实验数据来自 Keogh 博士的 Synthetic Control 数据集(http://www.cs.ucr.edu/~eamonn/time_series_data/)。Synthetic Control 实验数据集包含 300 条时间序列,每个时间序列包含 60 个属性,所有的数据被分成 6 类,每类包含 50 条。为了便于比较,仅仅抽取了前两类中的 6 条数据,每类各 3 条(1~3 为一类,4~6 为一类)。

在得到关键点序列之后,为了计算序列间的相似度,对其采用等维处理方法,而 DTW 算法的优势在于计算不等长序列的相似度。分别用 DTW 算法和本文所提出的方法对这 6 条序列进行了相似度的计算,结果如表 4 和表 5 所列。因为做了降维处理,所以对其可以进行等维计算。对一般的不等长序列,本文方法就不能处理。

表4 DTW 得到的 6 条时间序列间距离

	1	2	3	4	5	6
1	0	5.8323	5.7686	5.5175	5.4033	6.0739
2	5.8323	0	6.4217	5.4168	5.0877	5.5564
3	5.7686	6.4217	0	6.055	5.7973	6.3648
4	5.5175	5.4168	6.055	0	4.5232	5.0762
5	5.4033	5.0877	5.7973	4.5232	0	5.3402
6	6.0739	5.5564	6.3648	5.0762	5.3402	0

表5 利用本文方法得到的 6 条时间序列的距离

	1	2	3	4	5	6
1	0	9.8596	10.2475	11.2017	10.9469	10.9953
2	9.8596	0	10.9307	11.0668	11.2174	10.5754
3	10.2475	10.9307	0	11.248	10.8093	10.4868
4	11.2017	11.0668	11.248	0	5.3377	10.9338
5	10.9469	11.2174	10.8093	5.3377	0	9.0925
6	10.9953	10.5754	10.4868	10.9338	9.0925	0

从表 4 和表 5 结果可以看出,利用本文的算法计算它们之间的相似度,同一类的序列之间的距离小于不同类序列之间的距离。而采用 DTW 算法,不同类序列之间的距离有的小于同一类序列间的距离,这显然对聚类结果是不利的。

以上实验结果表明,本文提出的基于关键点技术的时间序列聚类方法,不仅对高维度的时间序列有很好的降维效果,节省了存储空间,在聚类准确性上也有较大的提高,且对低维序列也能计算出序列间比较理想的相似度。

结束语 本文提出了一种基于关键点技术的时间序列挖掘方法。首先利用关键点技术将时间序列降维,然后将它们进行等维处理后,计算其相似度,得到相似度矩阵后构造复杂网络,并用谱方法和 FCM 方法进行聚类。实验结果表明本方法是可行的,且得到了较好的聚类结果。算法解决了时间序列聚类过程中对噪声点和孤立点敏感的问题,但不能处理原时间序列是不等长的问题。本文在时间序列降维后,针对如何保持原序列的主要性质及计算其相似性方面进行了有益的尝试,实验证明了所提方法的合理性。

参考文献

- [1] 刘懿,鲍德沛,杨泽红,等. 新型时间序列相似性度量方法研究[J]. 计算机应用研究,2007,24(5):112-114
- [2] 董晓莉,顾成奎,王正欧. 基于形态的时间序列相似性度量研究[J]. 电子与信息学报,2007,29(5):1228-1231
- [3] 肖辉,胡运发. 基于分段时间弯曲距离的时间序列挖掘[J]. 计算机研究与发展,2005,42(1):72-78
- [4] Maharaj E A, D'Urso P. A coherence-based approach for the pattern recognition of time series[J]. Physica A, 2010, 389: 3516-3537
- [5] D'Urso P, Maharaj E A. Autocorrelation-based fuzzy clustering of time series[J]. Fuzzy Sets and Systems, 2009, 160: 3565-3589
- [6] 杜奕,卢德唐,李道伦,等. 一种快速的时间序列线性拟合算法[J]. 中国科学技术大学学报,2007,37(3):310-314
- [7] Keogh E, Ratanamahatana C A. Exact indexing of dynamic time warping[M]. Knowledge and Information Systems, Springer-Verlag London Ltd, 2005: 358-386
- [8] 刘慧婷,倪志伟. 基于 EMD 与 K-means 算法的时间序列聚类[J]. 模式识别与人工智能, 2009, 22(5): 803-808
- [9] 江克勤,施培蓓. 优化初始中心的模糊 C 均值算法[J]. 合肥工业大学学报, 2009, 32(5): 762-764

(下转第 173 页)

Growth 过程中与 Zookeeper 的交互。因为在 ParallelFP-Growth 过程中也需要用到 F-list 的信息,所以也要考虑后续的过程中与 Zookeeper 的交互。

4 实验结果

本文设计的优化算法与 Mahout 中的 PFP-Growth 算法保持了相同的时间复杂度,主要设计实验对比了算法优化前后的空间开销。实验平台为一个 16 个节点的机群,每个节点有一颗 Intel 2.5GHz 的双核处理器,内存为 2G,使用思科路由连接在同一网段。实验机群 16 个节点中 1 个节点为 master,11 个节点为 slave,3 个节点为 Zookeeper。所有节点上使用的都是 Ubuntu 10.04 Server 操作系统和 openjdk6,实验使用的 Hadoop 版本是 0.20.203.0, Mahout 的版本是 0.4, Zookeeper 的版本是 3.3.3。在分布为较小规模和较大规模的数据集上测试了本文优化策略的性能,所采用的测试数据集分别是 T40I10D100K 和 Webdocs,大小分别是 14.8Mb 和 1.38G。其中 T40I10D100K 是 IBM 数据生成器产生的模拟数据,共有 100000 条记录,而 Webdocs 是一个真实的数据集,共有 1692082 条记录。实验中最小 support 值都设为 1%。

实验结果如表 1 和表 2 所列。

表 1 优化前后的空间开销对比(在数据集 T40I10D100K 上)

Contents	Before Tuning	After Tuning
HDFS_BYTES_WRITTEN	85207	86
FILE_BYTES_WRITTEN	19001	292
REDUCE SHUFFLE BYTES	0	0
REDUCE OUTPUT RECORDS	942	0

表 2 优化前后的空间开销对比(在数据集 Webdocs 上)

Contents	Before Tuning	After Tuning
HDFS_BYTES_WRITTEN	12656116	86
FILE_BYTES_WRITTEN	284014127	17190
REDUCE SHUFFLE BYTES	37579111	308
REDUCE OUTPUT RECORDS	5267656	0

从表 1 和表 2 中可以看到,采用优化策略后的并行算法极大地减少了 HDFS 的空间开销以及 Reduce 中 Shuffle 的空间开销。这是因为通常在机群环境下,数据会被复制产生冗余,则实际带来的空间开销和网络资源开销会更大,同时该设计也减少了计算节点的内存开销,即计算节点不用下载整个 F-list 和 G-list,为计算节点提高了性能。

结束语 综上所述,本文分析了并行频繁模式增长算法,总结了之前相关算法在设计上的优缺点,详细分析了 Mahout 项目的并行频繁模式增长实现上的不足之处。并在此基础上提出了一种新的利用分布式协调系统的优化设计。本文的设计针对计数和排序部分进行改进,使得计数和排序部分可以同时进行,排序任务能够提前开始,并减少了网络开销。本文

的方法使得并行频繁模式的性能得到了提升,但是这样的设计本身也增加了一定的复杂度与实现难度,这对实现本文的设计提出了挑战。后续我们将继续本文工作,降低优化算法实现的时空复杂度,并在海量数据集上进一步验证优化后的算法性能。

参考文献

- [1] El-Hajj M, Zaiane O R. Parallel Leap: large-scale maximal pattern mining in a distributed environment [C] // Proceedings of the 12th International Conference on Parallel and Distributed Systems (ICPADS'06). 2006; 135-142
- [2] Aouad L M, Le-Khac N-A, Kechadi T M. Distributed Frequent Itemsets Mining in Heterogeneous Platforms [J]. Journal of Engineering, Computing and Architecture, 2007, 1(2)
- [3] Parthasarathy S, Zaki M J, Ogihara M, et al. Parallel Data Mining for Association Rules on Shared-Memory Systems [J]. Knowledge and Information Systems, 2001, 3(1)
- [4] Asif J, Ashfaq K. Frequent Pattern Mining on Message Passing Multiprocessor Systems [J]. Distributed and Parallel Databases, 2004, 16(3); 321-334
- [5] Iko P, Masaru K. Parallel FP-Growth on PC Cluster [C] // Proceedings of the 7th Pacific-Asia conference on Advances in knowledge discovery and data mining (PAKDD'03). 2003
- [6] Buehrer G, Parthasarathy S, Tatikonda S, et al. Toward terabyte pattern mining; an architecture-conscious solution [C] // Proceedings of the 12th ACM SIGPLAN symposium on Principles and practice of parallel programming (PPoPP '07). ACM, New York, NY, USA, 2007; 2-12
- [7] Li Hao-yuan, Wang Yi, Zhang Dong, et al. Pfp: parallel fp-growth for query recommendation [C] // Proceedings of the 2008 ACM conference on Recommender systems (RecSys '08). ACM, New York, NY, USA, 2008; 107-114
- [8] Zhang Yu-zhou, Wang Jian-yong, Zhou Li-zhu. Parallel Frequent Pattern Discovery [J]. Challenges and Methodology, Tsinghua Science & Technology, 2007, 12(6); 719-728
- [9] Hunt P, Konar M, Junqueira F P, et al. ZooKeeper: wait-free coordination for internet-scale systems [C] // Proceedings of the 2010 USENIX conference on USENIX annual technical conference (USENIXATC'10). USENIX Association, Berkeley, CA, USA, 2010
- [10] HDFS Architecture Guide [EB/OL]. http://hadoop.apache.org/common/docs/current/hdfs_design.html#N10159, 2011
- [11] Han J, Pei J, Yin Y. Mining Frequent Patterns without Candidate Generation [C] // Proceedings of the 2000 ACM SIGMOD International Conference of Management of Data. ACM press, May 2000; 1-12
- [12] Keogh E, Kasetty S. On Need for Time Series Data Mining Benchmarks; A Survey and Empirical Demonstration [C] // Proc of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Edmonton, Canada, 2002; 102-111
- [13] 汪小帆, 李翔, 陈关荣. 复杂网络理论及其应用 [M]. 北京: 清华大学出版社, 2006; 165-171

(上接第 159 页)

- [10] 张健沛, 杨悦, 杨静, 等. 基于最优划分的 K-means 初始聚类中心选取算法 [J]. 系统仿真学报, 2009, 21(9); 2586-2590
- [11] Keogh E, Chu S, Hart D, et al. An On-line Algorithm for Segmenting Time Series [C] // Proc. of the IEEE International Conference on Data Mining. San Jose, USA, 2001; 289-296