软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2020,31(3):763−777 [doi: 10.13328/j.cnki.jos.005912]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

融合选择提取与子类聚类的快速 Shapelet 发现算法*

赵 超[1], 王腾江[2], 刘士军[1], 潘 丽[1], 嵇 存[3]

[1](山东大学 软件学院,山东 济南 250101)

[2](浪潮通用软件有限公司,山东 济南 250101)

[3](山东师范大学 信息科学与工程学院,山东 济南 250014)

通讯作者: 刘士军, E-mail: lsj@sdu.edu.cn, 潘丽, E-mail:panli@sdu.edu.cn

摘 要: 基于 Shapelet 的时间序列分类算法具有可解释性,且分类准确率高、分类速度快.在这些算法中,Shapelet 学习算法不依赖于单一分类器,能够学习出不在原始时间序列中的 Shapelet,可以取得较高的分类准确率,同时还可以保证 Shapelet 发现和分类器构建同时完成;但如果产生的 Shapelet 过多,会增加依赖参数,导致训练时间太长,分类速度低,动态更新困难,且相似重复的 Shapelet 会降低分类的可解释性.提出一种选择性提取方法,用于更精准地选择 Shapelet 候选集,并改变学习方法以加速 Shapelet 学习过程;方法中提出了两个优化策略,通过对原始训练集采用时间序列聚类,可以得到原始时间序列中没有的 Shapelet,同时在选择性提取算法中加入投票机制,以解决产生 Shapelet 过多的问题.实验表明,该算法在保持较高准确率的同时,可以显著地提高训练速度.

关键词: 时间序列;分类;Shapelet;候选集;选择性提取

中图法分类号: TP18

中文引用格式: 赵超,王腾江,刘士军,潘丽,嵇存.融合选择提取与子类聚类的快速 Shapelet 发现算法.软件学报,2020,31(3):763−777. http://www.jos.org.cn/1000-9825/5912.htm

英文引用格式: Zhao C, Wang TJ, Liu SJ, Pan L, Ji C. Fast shapelet discovery algorithm combining selective extraction and subclass clustering. Ruan Jian Xue Bao/Journal of Software, 2020,31(3):763−777 (in Chinese). http://www.jos.org.cn/1000-9825/5912.htm

Fast Shapelet Discovery Algorithm Combining Selective Extraction and Subclass Clustering

ZHAO Chao[1], WANG Teng-Jiang[2], LIU Shi-Jun[1], PAN Li[1], JI Cun[3]

[1](School of Software, Shandong University, Ji'nan 250101, China)

[2](Inspur General Software Co. Ltd., Ji'nan 250101, China)

[3](School of Information Science and Engineering, Shandong Normal University, Ji'nan 250014, China)

Abstract: The time series classification algorithm based on Shapelet has the characteristics of interpretability, high classifica-tion accuracy and fast classification speed. Among these Shapelet-based algorithms, learning Shapelet algorithm does not rely on a single classifier, and Shapelet that is not in the original time series can be learned, which can achieve a high classification accuracy and ensure that Shapelet discovery and classifier construction are completed at the same time. However, if too many Shapelets are generated, it will increase the dependent parameters, resulting in too long training time, low classification speed, and difficult dynamic updates. And similar redundancy Shapelets will reduce the interpretability of the classification. This study proposes a new selective extraction algorithm to select Shapelet candidate set and change the learning method to accelerate the learning process of Shapelet and puts forward two

* 基金项目: 国家自然科学基金(61872222); 山东省重点研发计划(2018GGX101019); 山东大学未来学者计划

Foundation item: National Natural Science Foundation of China (61872222); Key Research and Development Program of Shandong Province (2018GGX101019); Young Scholars Program of Shandong University

本文由人工智能赋能的数据管理、分析与系统专刊特约编辑李战怀教授、于戈教授和杨晓春教授推荐.

收稿时间: 2019-08-12; 修改时间: 2019-09-10, 2019-11-25; 采用时间: 2019-12-18; jos 在线出版时间: 2020-01-10

CNKI 网络优先出版: 2020-01-10 13:35:00, http://kns.cnki.net/kcms/detail/11.2560.TP.20200110.1334.012.html

optimization strategies. By using time series clustering for the original training set, Shapelets not in the original time series can be obtained. Meanwhile, a voting mechanism is added into the selective extraction algorithm to solve the problem of excessive Shapelet generation. Experiments show that the proposed algorithm can improve the training speed while maintaining high accuracy.

Key words: time series; classification; Shapelet; candidates; selective extraction

时间序列分类是时间序列数据挖掘的经典问题和热点问题之一,其主要内容是将一个未知类的时间序列划分到已知的类别中.传统的分类问题中,分类器的输入为训练集的特征向量,而时间序列数据与此不同,其并没有明确的特征;另一方面,时间序列数据的属性有先后次序关系,传统的分类问题则没有这方面的考虑.因此,一般的分类方法在时间序列数据上建立起的分类器往往缺乏可解释性;同时,时间序列数据维度高、数据量大的特点也导致传统分类方法在时间序列数据上执行特征选择的计算开销很大.

针对现有时间序列分类算法存在的分类时间长、分类结果可解释性弱这两个问题,Ye 和 Keogh 提出了 Shapelet[1]的概念.Shapelet 是时间序列中一段能够最大程度定义类别信息的子序列,这些子序列可以出现在时间序列的任何位置,且一般长度较短.与其他分类算法相比,由于 Shapelet 可以直观地表示出类别之间的差异性,通过整个时间序列中是否存在一个或多个 Shapelet 就可以区分所属的不同类,所以基于 Shapelet 的分类算法具备更强的可解释性.如图 1 所示,箭头所指为一个可能的 Shapelet,因为其具有明显的区分度,可以将两条时间序列显著区分开来.

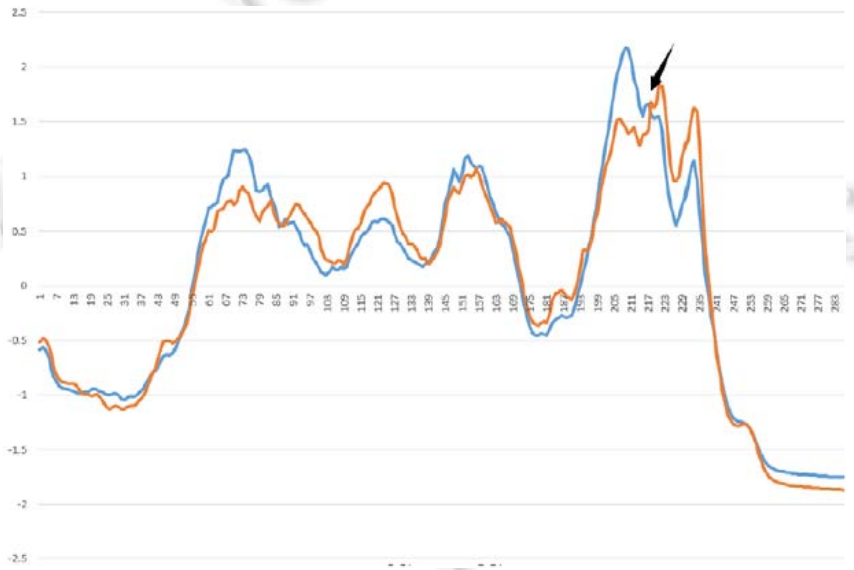

Fig.1 Shapelet sketch

图 1 Shapelet 示意图

最初提出的 Shapelet 发现算法通过枚举所有可能的候选对象来找到 Shapelet,该算法的时间复杂度为 $O(n^2m^4)$(n 和 m 分别是时间序列的数量和长度)[2,3].显然,这个时间复杂度非常高.针对这个问题,研究者提出了很多加速策略,如子序列距离早放弃技术和允许熵剪枝技术等,其中,快速 Shapelet 算法(fast Shapelets)[4]是最快的.

由于直接将 Shapelet 嵌入决策树中无法生成特别精确的分类器,Lines 等人提出了一种 Shapelet 变换算法,通过将 Shapelet 作为特征,可以将时间序列变换成一个特征向量,时间序列分类的问题也就转变为一般的分类问题[5,6].Grabocka 等人则提出了一种 Shapelet 学习算法[7],通过将逻辑回归应用到 Shapelet 训练里,达到了非常高的准确率.事实上,除了 Shapelet 学习算法外,所有其他 Shapelet 发现算法都需要从时间序列数据集的大量子序列中搜索 Shapelets,但分类效果好的 Shapelet 可能并不存在于原始的时间序列中.例如,有两个子序列都不是特征很显著的 Shapelet,但将它们按时间轴对齐,并对每个点求平均值后,得到的结果序列可能会是一个更显著

的 Shapelet.Shapelet 学习算法正是能够学习出不存在于原始的时间序列中的 Shapelet,从而能够获得较高的分类准确率,但它同时也带来了可解释性降低、训练速度慢的弊端.

针对这一问题,我们在研究中提出了一种选择性提取算法 S4(selecting superior candidates from a suitable set)去选择 Shapelet 候选集,并改变学习方法以加速 Shapelet 学习过程;算法在保持 Shapelet 学习算法高准确率的同时,最大程度地解决了可解释性低、训练时间长这两个严重缺陷.为了保留 Shapelet 学习算法可以学习出原始时间序列中不存在的 Shapelet 这一优势,并解决产生的结果 Shapelet 过多这一问题,又对 S4 提出了两个优化策略:通过对原始训练集采用时间序列聚类,可以得到原始时间序列中没有的 Shapelet;同时,在选择性提取算法中加入投票机制,以解决产生 Shapelet 过多的问题.优化后的 S4 算法称为 ES4(enhanced-S4).实验结果表明,ES4 算法的分类准确率与准确度最高的 Shapelet 学习算法基本一致,但训练速度提升 100 倍以上.

1 相关工作

针对 Shapelet 发现的加速方法.Ye 和 Keogh 提出了子序列距离早放弃和允许熵剪枝两种加速方法,Mueen 等人使用矩阵来缓存距离,并使用三角不等式对 Shapelet 候选集进行修剪[8].然而即便如此,训练的时间复杂度仍然达到了 $O(n^2m^3)$.Chang 等人则是在 GPU 上实现了 Shapelet 的发现过程[9],此方法减少了时间,但依赖于硬件条件.He 等人提出了一个假设:辨识度较高的子序列往往比其他的平凡子序列出现次数少得多.他利用这些子序列进行计算,加快了运行时间.但他的这一假设并不适用于某些特定的数据集[10].2013 年,Rakthanmanon 和 Keogh 提出了一种名为 Fast Shapelet(FS)的算法[4]:首先,将时间序列进行符号聚合近似表示;然后,通过在符号化表示的时间序列上投影来查找 Shapelet,将时间复杂度降低到了 $O(nm^2)$,然而却导致了预测准确率的降低.

由于决策树是一种较弱的分类器,它限制了基于 Shapelet 算法的分类精度[5,6].于是,Lines 等人提出了一种 Shapelet 变换方法[5,6].他们首先使用其他质量评估方法(如 F 统计量或 Kruskal-Wallis 检验)找到最好的 k 个 Shapelet;然后使用这些 Shapelet 作为时间序列的特征,每个特征对应的值是该 Shapelet 与时间序列之间的欧式距离.通过这个过程,时间序列分类就变成了一个普通的分类问题.然而,这个方法将 Shapelet 的发现过程与训练分类器的过程分离开来,因此可能无法找到最佳的 Shapelet 组合.

最近几年,Grabocka 等人提出了一种名为 LTS 的新算法[7],在训练分类器的同时直接学习出最终的目标 Shapelet,而不是像从前那样从所有子序列中去搜索.LTS 的分类精度达到了相当高的水平,但机器学习的训练时间则很慢.后来,Wistuba 等人提出了一种基于随机抽样的 Shapelet 发现算法 UFS[11].他们认为 Shapelet 应当是频繁出现,因此采用随机抽样的方法.然而,随机方法将导致生成非常多的 Shapelet,降低了方法的可解释性,且分类精度不高.Hou 等人利用广义特征向量和稀疏建模技术,通过确定 Shapelet 位置的方式直接从原始时间序列中学习最终 Shapelet[12],但其分类准确率比 LTS 下降较为明显.Yang 等人也尝试使用自组织增量神经网络去学习 Shapelet,速度明显加快,但分类准确率同样达不到 LTS 的水平[13].

2 问题描述

2.1 相关定义

定义 1(时间序列). 时间序列 T 是一条连续实值序列,即 $T=\{t_1,t_2,...,t_m\}$,其中:$t_1,t_2,...,t_m$ 是按照时间序列排列的观测值,且相邻观测值之间具有相同的时间间隔;m 为 T 的长度.

定义 2(子序列). 时间序列 T 的子序列 $T_{i,l}=\{t_i,t_{i+1},...,t_{i+l-1}\}$ 是一条 T 中从第 i 个位置开始、长度为 l 的一段连续子集.这里,$1\leqslant i\leqslant m-l+1$.

定义 3(数据集). 时间序列数据集 D 是若干条个时间序列组成的集合,D 中包含的时间序列序列数目用符号 n 表示,在分类问题中,每条时间序列属于一个具体的类,数据集中类的数量表示为 C.

定义 4(数据集子类). 时间序列中数据集 D 中属于某个特定类别的子序列构成了数据集子类.一个时间序列数据集中包含了 C 个数据集子类,其中,C 为时间序列数据集中类的数量.

定义 5(时间序列间的距离). 时间序列之间的距离 $dist(T,R)$是一个距离函数,用于度量两个时间序列间的相似性.本文使用欧式距离,对于两个长度为 m 的时间序列,其计算方法如公式(1)所示:

$$dist(T,R)=\sqrt{\sum_{i=1}^{m}(t_i-r_i)^2} \tag{1}$$

定义 6(时间序列与子序列的距离). 时间序列与子序列的距离 $sdist(T,S)$是一个距离函数,输入长度为 m 的时间序列和长度为 l 的时间序列子序列,两者距离的计算方法如公式(2)所示:

$$sdist(T,S)=\min_{1\leqslant p\leqslant m-l+1} dist(S,T_p^l) \tag{2}$$

定义 7(Shapelet). 时间序列 shapelet 是时间序列中能够最大限度地表示一个类别的子序列,它本质上也是时间序列的一段子序列,所以 Shapelet 与时间序列的距离也可以由公式(1)计算.

定义 8(Shapelet 变换). Shapelet 变换指将时间序列 Shapelet 变换到特征空间.Shapelet 变换以每个 Shapelet 为特征,Shapelet 与时间序列的距离作为特征值.设所选 Shapelet 集合为 S,时间序列 T 经 S 变换后的形式 $T_{transformed}$ 如公式(3)所示,其中,$|S|$为 Shapelet 集合中 Shapelet 的数量:

$$T_{transformed}=\{sdist(S_1,T),sdist(S_2,T),\ldots,sdist(S_{|S|},T)\} \tag{3}$$

2.2 问题分析

2.2.1 Shapelet 学习算法的问题

目前的 Shapelet 学习算法的基本思路如下:

(1) 对原始数据集中时间序列的所有子序列进行聚类,将得到的 k 个聚类中心作为 Shapelet 候选集;

(2) 使用这些候选集对训练集中的所有时间序列进行变换,得到若干个 k 维向量;

(3) 迭代训练 Shapelet,将 Shapelet 训练和分类器构建同时完成.

这个思路的特点是可以学习出不在原始时间序列中的 Shapelet,准确率高,同时还可以保证 Shapelet 发现和分类器构建同时完成.但它同时也面临着可解释性弱和训练速度慢的问题.

- 第一,使用时间序列子序列的聚类结果作为 Shapelet 候选集的做法降低了可解释性,如果多个类的时间序列中都含有同样一段无意义的子序列,如设备刚启动时的一段传感数据子序列,这段序列往往会被聚类为一个中心并作为 Shapelet 候选.显而易见,这段子序列并不符合 Shapelet 选取的初衷,即对类别的辨识度.同时,选取的结果 Shapelet 过多,特别是相似 Shapelet 过多,也会降低方法的可解释性;
- 第二,参数依赖过多,包括 Shapelet 长度、Shapelet 数量等,这导致训练过程异常复杂、训练速度缓慢.我们在实验中发现,即使各项参数已经按算法建议设定好,部分数据集上的 Shapelet 训练时间仍长达数小时之久.

2.2.2 解决方案

对于问题 1,即聚类结果作为 Shapelet 候选集的可解释性弱的问题,可以看出,选择更具解释性的 Shapelet 候选集提取方法是非常有必要的.在文献调研时,我们发现了 Matrix Profile 方法,它是为了解决时间序列相似性搜索而提出的,但其思路对 Shapelet 发现也具有意义[14,15].通过对其进行优化,并在提取之前加入时间序列采样技术,可以使其更适合用来做 Shapelet 发现.我们将优化后的算法称为 S4[16],在本文中,使用它进行 Shapelet 候选集提取,可以达到去除无意义子序列的目的,同时得到具有类别辨识度的子序列.使用这些子序列作为 Shapelet 候选集合更具有可解释性.

S4 算法虽然使 Shapelet 学习算法的训练过程加速了很多,但它只能从原始序列中提取 Shapelet,丢失了 Shapelet 学习算法可以得到不在原始数据集中出现的 Shapelet 这一优势.并且,虽然通过精简候选一定程度上抑制了生成 Shapelet 数目过多的问题,但实验过程中仍然发现有大量相似的 Shapelet 出现,这使算法整体解释性降低且分类速度下降.除此之外,时间序列采样的方法并不适用于所有数据集,尤其对训练集数目较少、每条时间序列均有各自特征的数据集效果不好.对此,我们提出了两个优化策略,即子类聚类和投票法.对训练集中每一类时间序列进行聚类,以每个类别得到的若干聚类中心去做选择性抽取,聚类可以使抽取的 Shapelet 不局限于

原时间序列中.同时,在选择性提取算法中加入投票机制,统计每个子序列的得票数量,去掉相互之间重叠过多的子序列,可使生成的 Shapelet 重复数量减少.优化后的 S4 算法称为 ES4.

对于问题 2,则是采用改变学习策略的方式.Lines 等人在 Shapelet 变换算法中已经证明,支持向量机对转换后的时间序列数据得分最高[5,6].我们在此基础上将属性选择技术 L1 范式添加到 SVM 中,可以保证 Shapelet 选择过程和分类器训练能够同时完成.众所周知,L1 范式可以避免过拟合,生成稀疏解.通过最小化损失函数后可以得到一个稀疏权矩阵,去除掉对应权重均为零的 Shapelet 候选,候选集中剩余的 Shapelet 候选即是最终所求的 Shapelet.

3 选择性 Shapelet 提取算法 S4

3.1 时间序列采样

在时间序列训练集中,每个时间序列都有一个确定的类.例如,数据集“Coffee”[17]中有 28 个时间序列,其中 14 个属于“0 类”,另外 14 个属于“1 类”.这意味着可以从整个训练集中采样一部分时间序列,以达到减少数量的目的.

研究已经证明,对每一类直接采样一些时间序列是不明智的,因为这不能充分概括训练集的特性,从而导致降低分类准确性[18].这其中一个重要的原因就是大多数数据集中都有子类的存在.为了解决这个问题,本文将使用与文献[18]中相同的方法对原始数据集进行子类划分,并从每个子类中采样一些时间序列.时间序列采样示意图如图 2 所示.

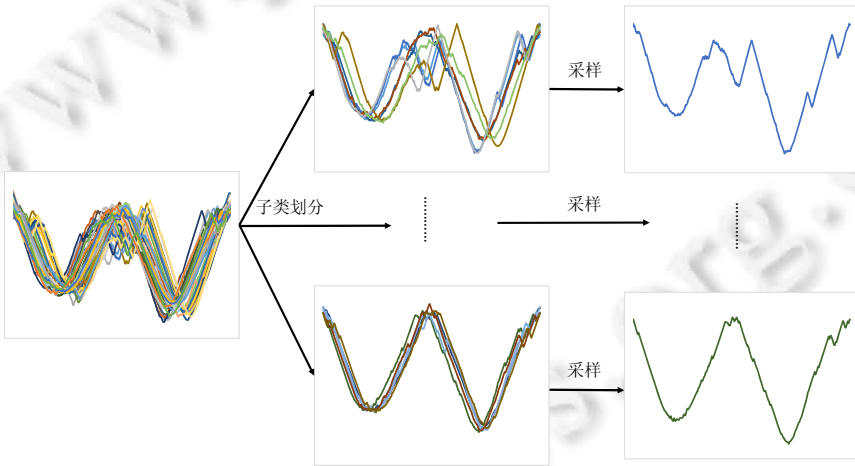

Fig.2 Process of time series sampling

图 2 时间序列采样的过程

下面通过一个简单的例子来解释这个方法.

假设有 n 个属于同一类的时间序列,每个时间序列的长度为 m,这些时间序列的集合用 D 来表示.则采样顺序应当按以下步骤进行.

(1) 需要找出一个标准时间序列.标准是指此时间序列的和值最为接近集合 D 中所有时间序列和值的平均值,和值即一个时间序列中所有元素之和.通过计算每个时间序列的和值和集合 D 的和值均值,可以在此集合中找到标准时间序列.如上所述,标准时间序列的和值与均值的差值最小.

(2) 对这个时间序列集合进行子类划分.首先计算标准时间序列与该集合的其他时间序列之间的欧式距离,并根据距离值对这些序列进行排序并计算相邻距离值的差值.然后计算这些差值的标准差.最后,可以将这些时间序列划分为几个子类:若相邻的两个距离的差值大于标准差,则它们将会分别被划分放入到不同的子类

中;若相邻的两个距离的插值小于标准差,则它们放入到同一个子类中.

(3) 从每个子类中抽取一个时间序列.在每个子类中,到其他时间序列的距离之和最小的时间序列会被采样来用.

3.2 基于重要点的选择性提取

在之前的工作中,我们定义了重要数据点(important data points,简称 IDPs)来做时间序列分段表示,并使用重要点对 Fast Shapelet 算法和 Shapelet 变换算法进行了加速[19−21].在 S4 算法中,重要点仍然扮演着一个重要的角色.

重要点可以被定义为满足以下条件的点.

(1) 该点所在的子序列在时间序列中具有最大权值,其中,子序列的权值可以用公式(4)计算,其中,$dist_{sum}$ 表示序列中每个点到拟合直线的距离之和,$dist_{\max}$ 表示这些距离中的最大值:

$$weight=\max(dist_{sum},2\times dist_{\max}) \tag{4}$$

(2) 该点到其所在的子序列的拟合直线的距离最大.

对一个时间序列标识重要点的过程如图 3 所示.

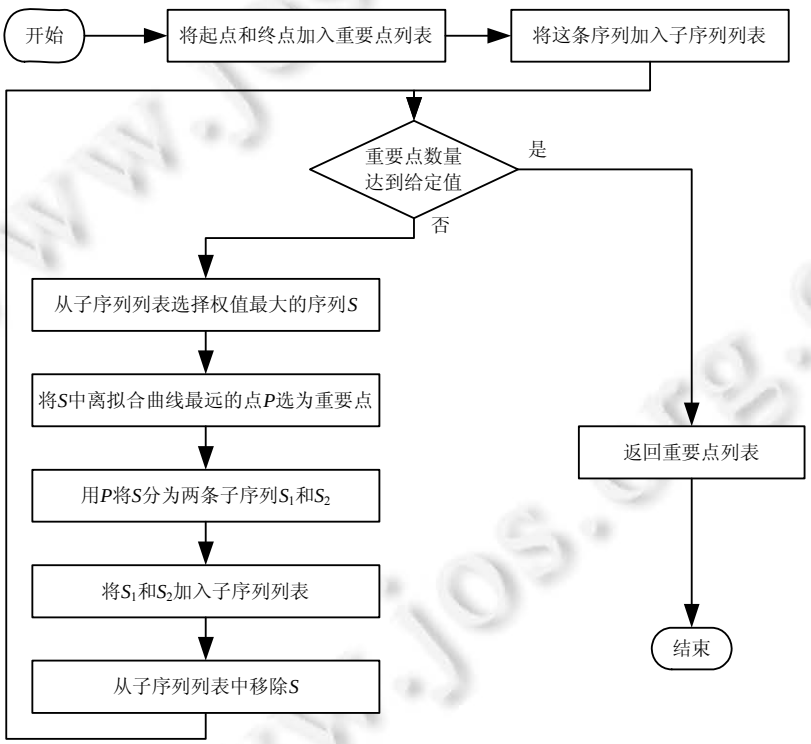

Fig.3 Process of identifying important points

图 3 标识重要点的过程

在 S4 中,需要对上一节获得的采样后训练集标识其中每个时间序列的重要点.在标识完重要点之后,将包含一个或多个重要点的子序列的索引作为重要子序列列表(important subsequence list,简称 ISL).将是否包含重要点作为选取候选 Shapelet 的依据,如图 4 所示:子序列 S_2 不包含重要点,其不能候选为 Shapelet;子序列 S_1 包含重要点,可以作为 Shapelet 候选.基于重要点将极大地较少 Shapelet 候选的数目.

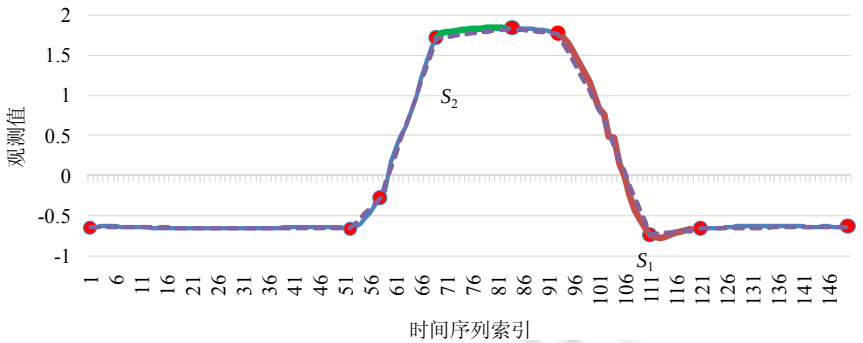

Fig.4 An example of shapelets filtering based on important points

图 4 基于重要点过滤 Shapelet 的一个例子

我们之前的工作中给出了基于重要点过滤提取 Shapelet 的算法[16],具体过程是只对 ISL 中的子序列计算相似性向量 ISSV(important subsequence similarity vector),并计算两个 ISSV 间的差异 *Diff*,如果 *Diff*(*i*)的值大于阈值,并且呈峰状,则相应的 ISL 中的子序列可被选为一个候选 Shapelet.其中阈值的计算公式如(5)所示:

$$threshold=mean(Diff)+\alpha\times std(Diff) \tag{5}$$

mean(*Diff*)和 *std*(*Diff*)分别表示 *Diff* 的平均值和标准差.变量α控制提取 Shapelet 的严格程度,α越小,产生 Shapelet 候选越多;α越大,则产生候选越少.通常情况下,α取值在 0~5 之间,通过对训练集五折交叉验证获取.

3.3 选择最终Shapelet

提取算法生成 Shapelet 候选集后,可以利用它对训练集中的原始时间序列进行 Shapelet 变换.对于变换后的特征向量,使用最小二乘支持向量机作为分类器去学习最终的 Shapelet.方法是通过最小化损失函数得到一个稀疏权重矩阵;然后,对于每个 Shapelet 候选,若与之相关的所有权重均为 0,则将其剔除出去;最终,候选集中剩余的 Shapelet 候选将作为最终的 Shapelet.

4 基于子类聚类的改进算法 ES4

由于 S4 改变了传统 Shapelet 学习策略与候选集生成方式,使得无法再学习出不在原始时间序列中的 Shapelet,并且 Shapelet 数目过多的问题仍然存在[22].本文提出了两种优化策略,通过对训练集进行子类聚类而不再是时间序列采样,并在选择性提取算法中加入投票机制去掉相互之间重叠过多的子序列.改进后的算法称之为 ES4.

4.1 子类聚类

自组织增量神经网络(self-organized incremental neural network,简称 SOINN)是一种无标签数据的在线无监督学习算法,它可以学习出无监督在线数据的拓扑结构,并报告出一个合理的聚簇数量,还能在没有先验条件的情况下给出每个聚簇的典型原型模式[23,24].

输入一个数据集,SOINN 会生成一个由节点集合和节点之间的连线集合组成的拓扑结构.在本文的工作中只使用节点集,每个节点也称为一个原型,SOINN 学习的每个原型都是相似数据的平均值,这可以保证更加有效地利用数据集信息.因此,首先使用 SOINN 对输入的时间序列训练集进行子类聚类.

首先,将时间序列数据集 *D* 按类别分成若干个集合,共 *C* 个;然后使用 SOINN 对每个集合聚类,对每个集合可以学习出若干原型;最后,将每个集合学习出的原型组合在一起作为选择性提取的对象.

在选择性提取 Shapelet 候选之前对训练集进行子类聚类并学习原型是有意义的,这可以保证选择性提取方法得到的结果 Shapelet 候选不拘泥于训练集中的原始时间序列子序列,丰富了 Shapelet 的种类;并且可以在保证原数据集特征的同时,对选择性提取的输入数据剪枝.

4.2 子类聚类

ES4 使用 SOINN 聚类得到的结果之间可以保证区分度,因此 S4 算法中,不同时间序列之间采样出相似 Shapelet 的问题可以得到缓解.接着再加入一项投票机制,可以完全杜绝同一时间序列中的子序列被多次提取.

投票过滤针对的是 Shapelet 选择过程,也就是从子类聚类中所得到的原型中选择合适的子序列作为 Shapelet 的过程.假设经过 SOINN 子类聚类后得到的结果聚类中心集合为 Ds,其中共 C 个类,我们以提取类 1 中的一个时间序列 T_1 的子序列为例说明改进后的提取过程.

给定子序列长度为 l,首先将 T_1 中所有子序列赋予一个初始得票数为 0,对 T_1 和类 2 中的所有时间序列求 ISSV;然后,对于每个 ISSV 向量,若其中存在可能的 Shapelet,则给 T_1 中对应位置的子序列得票数加 1;对 T_1 和类 2 中所有时间序列的 ISSV 向量处理完毕后,统计 T_1 中各子序列的得票数,对于得票数超过类 2 中时间序列数量一半的子序列选择为 Shapelet 候选.以此类推,以相同的方式对其他类进行处理.加入投票机制后的选择性提取算法描述如下.

算法. 加入投票机制后的选择性提取过程.

输入:经 SOINN 聚类之后得到的聚类中心集合,Ds,$Ds(i)$是属于类 i 的集合,每个时间序列长度为 m;

D 中的重要点的索引集合,IDP;

D 中的类别集合,C;

时间序列子序列长度,l;

控制阈值参数,α;

输出:shapelet 候选集,*candidates*;

```
candidates=∅
for each c1∈C do
  for each Ta∈Ds(c1) do
      ISL=∅, HashSet=∅,VOTE=zeros(m–l+1,1)
      将 Ta 中包含重要点 IDP 集合中一个或多个点的子序列(重要子序列)的索引按顺序放入 ISL
      ISSVa,a=CalculateISSV(Ta,Ta,l,ISL)
      for each c2∈C do
          if c2!=c1
              for each Tb∈Ds(c2) do
                  ISSVa,b=Calculate ISSV(Ta,Tb,l,ISL)
                  DIFF=ISSVa,b–ISSVa,a
                  threshold=mean(DIFF)+α*std(DIFF)
                  for i=0; i<DIFF.length; i++ do
                      if DIFF[i]>threshold then
                          VOTE[ISL(i)]=VOTE[ISL(i)]+1
          for i=0; i<VOTE.length; i++ do
              if VOTE[i]>=Ds(c2).length/2 then
                  HashSet.ad(i)
      for each index∈HashSet do
          将 Ta 中 index 处子序列加入 candidates 中
return candidates;
```

需要指出的是,聚类与投票针对的是不同的过程,由于通常一条时间序列中包含的 Shapelet 候选(子序列)数量都远远超过最终分类所需要的 shapelet 的数量,无法通过直接减小聚类个数替换代替投票过滤过程.

4.3 算法时间复杂度分析

在 ES4 中,我们使用 SOINN 子类聚类代替了 S4 中的子类划分采样法,并在选择性提取过程中加入了投票机制.最终,ES4 总体过程包括以下 3 个过程步骤:1) SOINN 聚类;2) 标识重要点;3) 加入投票机制后的选择性提取.3 个过程串行方式进行.假设训练集中时间序列的个数为 n,每条时间序列的长度为 m,训练过程各个过程时间消耗如下.

(1) SOINN 聚类.在 ES4 中,我们使用 SOINN 子类聚类替换了 S4 中的子类划分采样法,SOINN 子类聚类的时间复杂度是 $O(knm)$[22],其中,k 代表聚类得到的结果中原型的个数.通过聚类最终获取的时间序列数量为 $O(k)$;

(2) 标识重要点.对于重要点,采用的是自顶向下分段方式标识.标识一条时间序列中重要点的时间复杂度为 $O(pm)$[25,26](p 为重要点的数量).在 SOINN 聚类过程中保留了 $O(k)$条时间序列用于获取 Shapelet 候选.因而,标识重要点的时间复杂度为 $O(pkm)$;

(3) 加入投票机制后的选择性提取方法.在方法 S4,假设有 c 个时间序列被采样,则用于生成 shapelet 候选集的时间序列从 $O(n)$减少到 $O(c)$,其时间复杂度为 $O(mpc^2\log m)$[16],其中,p 为重要点的数量.在 ES4 中,我们采用 SOINN 子类聚类代替了 S4 中的子类划分采样法,获取的时间序列数量为 $O(k)$.对于 ES4 的选择提取部分,输入时间序列为 $O(k)$个,投票机制的加入对时间复杂度的规模没有影响,最终选择性提取方法的复杂度为 $O(mpk^2\log m)$.

在 ES4 中,这 3 个过程是是串行进行的,因而 ES4 的总体时间复杂度为 3 个过程中时间复杂度的最大值 $O(mpk^2\log m)$.这个时间复杂度较对比算法为低.我们将在第 5.4 节的训练耗时对比实验中做进一步验证.

5 实验验证

5.1 数据集描述

本实验使用了 UCR 时间序列分类仓库中的部分数据集.UCR 时间序列分类仓库有大量可公开访问的数据集,它涉及领域广泛,包括环境监测和医疗诊断等等.我们从每种类型中选取了部分数据集.数据集详细信息见表 1.之选择这些数据集是基于以下考虑:(1) 这些数据集是可公开访问的;(2) 其他对比方法在这些数据集上的结果可以在相关文献中找到.实验中使用到的所有数据集都可以在文献[17]中下载.

Table 1 Dataset

表 1 数据集

数据集	训练/测试集数目	长度	类别数目
Adiac	390/391	176	37
Beef	30/30	470	5
Chlorine.	467/3840	166	3
Coffee	28/28	286	2
Diatom.	16/306	345	4
ECGFiveDays	23/861	136	2
FaceFour	24/88	350	4
GunPoint	50/150	150	2
ItalyPower.	67/1029	24	2
Lightning7	70/73	319	7
MoteStrain	20/1252	84	2
Sony.	20/601	70	2
Symbols	25/995	398	6
Synthetic.	300/300	60	6
Trace	100/100	275	4
TwoLeadECG	23/1139	80	2

5.2 实验设定

本文使用以下 6 种基于 Shapelet 的方法作为对比方法,对提出的算法 S4 和 ES4 进行评估.

(1) Shapelet 发现算法:原始的 Shapelet Tree 算法(IG),通过搜索训练集中的所有子序列,并使用信息增益

评估它们来发现 Shapelet[1].Fast Shapelets(FS)算法使用 SAX 加速 Shapelet 发现过程[4];

(2) Shapelet 变换(ST)算法:利用时间序列与每个 Shapelet 之间的距离对原始时间序列进行变换后选择分类器分类[5,6];

(3) 其他基于 Shapelet 的算法:LTS 直接通过最小化分类损失函数来学习 Shapelets[7].UFS 使用随机 Shapelets 作为候选,对时间序列进行变换[11].FLAG 使用 GEM 和 Fused-Lasso 直接学习[12].

这些算法的代码(除了 UFS 和 FLAG 之外)是由文献[27,28]提供的,参数也是按照各算法的建议值设置.实验运行软件环境是 Java 1.8,硬件环境是一台 Intel i5-3470 CPU 3.20GHz 和 8GB 内存的 Windows 主机.

5.3 准确率对比

我们对选取的算法进行了大量分类准确率的对比实验.准确率见表 2,其中,括号中的数字表示在该数据集下算法准确率的排名.每个数据集下准确率最高的算法被标为粗体.

从表 2 中的算法排名可以看出,ES4 算法与 LTS 算法基本处于相同的级别,S4 仅次于这两种算法.具体来说,LTS 平均排名为 1.937 5,ES4 则与之非常接近,为 2.062 5,S4 为 2.562 5.与 S4 相比,ES4 在 7 个数据集上比 S4 准确率有提升,在 6 个数据集上持平,在 3 个数据集上有下降.这可能是因为 ES4 相比 S4 生成了一些不在原始时间序列上的 Shapelet.这证明:通过 SOINN 学习出的不在原始数据集中的 Shapelet 起到了作用,有效地提升了准确率.还可以看出:与 S4 和 ES4 相比,UFS 和 FLAG 的分类准确率明显不占优势.这说明为了加速 Shapelet 的发现过程,UFS 和 FLAG 牺牲了一定的准确率.

Table 2 Accuracy comparison

表 2 准确率对比

数据集	IG	FS	ST	LTS	FLAG	UFS	**S4**	**ES4**
Adiac	0.299(7)	0.558(5)	**0.783(1)**	0.542(6)	0.752(2)	0.698(3)	0.665(4)	0.698(3)
Beef	0.5(7)	0.517(6)	**0.9(1)**	0.8(4)	0.833(2)	0.667(5)	0.867(2)	0.833(3)
Chlorine.	0.588(7)	0.578(8)	0.7(6)	0.743(4)	0.76(3)	0.738(5)	0.768(2)	**0.77(1)**
Coffee	0.964(2)	0.925(3)	0.964(2)	**1(1)**	**1(1)**	0.964(2)	**1(1)**	**1(1)**
Diatom.	0.722(7)	0.869(6)	0.925(5)	0.951(4)	**0.964(1)**	0.958(2)	0.951(4)	0.954(3)
ECGFiveDays	0.775(5)	0.996(2)	0.984(3)	**1(1)**	0.92(4)	**1(1)**	**1(1)**	**1(1)**
FaceFour	0.841(5)	0.909(3)	0.852(4)	**1(1)**	0.909(3)	0.932(2)	**1(1)**	**1(1)**
GunPoint	0.893(5)	0.932(4)	**1(1)**	**1(1)**	0.967(3)	0.987(2)	0.993(2)	**1(1)**
ItalyPower.	0.892(7)	0.921(6)	0.948(3)	**0.962(1)**	0.946(4)	0.94(5)	0.95(2)	0.948(3)
Lightning7	0.493(6)	0.601(5)	0.726(3)	**0.877(1)**	0.767(2)	0.685(4)	0.767(2)	0.767(2)
MoteStrain	0.825(7)	0.785(8)	0.897(3)	0.913(2)	0.888(5)	0.872(6)	0.891(4)	**0.919(1)**
Sony.	0.845(4)	0.698(7)	0.844(5)	**0.952(1)**	0.929(3)	0.79(6)	0.94(2)	0.94(2)
Symbols	0.78(7)	0.93(3)	0.882(5)	**0.959(1)**	0.875(6)	0.888(4)	0.95(2)	0.93(3)
Synthetic.	0.943(5)	0.917(6)	0.983(3)	**1(1)**	0.997(2)	0.997(2)	0.97(5)	0.973(4)
Trace	0.98(3)	**1(1)**	**1(1)**	**1(1)**	0.99(2)	0.96(4)	**1(1)**	**1(1)**
TwoLeadECG	0.851(6)	0.922(5)	0.997(2)	**1(1)**	0.99(3)	0.836(7)	0.977(4)	0.99(3)

5.4 训练耗时对比

前面提到,本文研究工作的目标是在尽量保持较高分类准确率的同时减少算法的运行时间.第 5.3 节证明了准确率已经达到预期,为此,我们选择与 ES4 方法准确率近似的 LTS 算法、ST 算法进行对照.表 3 显示了每种算法的总训练时间,单位 s.

从表 3 中可以看出,对于原始的 ST 和 LTS 算法,S4 和 ES4 比 ST 快 100 多倍;在大多数数据集上比 LTS 快 200 倍.表明本文研究提出的算法在时间效率上有很大的提升.

表中也可以看到,算法 ES4 的训练时间略大于 S4.这是因为使用 SOINN 进行子类聚类比使用采样法更加复杂,时间消耗也更多.实验还发现:通过子类聚类得到的结果数要多于采样,因此选择性提取的工作量也有所增加.可以看到,两者训练时间相差最大的数据集是 Adiac 和 Synthetic_control.这是因为在这两个数据集上,子类聚类法比采样法的结果多出很多,使得选择性提取过程相比之前要慢很多.而在其他大多数数据集上, ES4 聚类产生的数量同样大于 S4 采样获得的结果数量,但训练时间相差却不多.这是因为 ES4 在加入投票机制之后,Shapelet 候选数量有很显著的下降,使得 Shapelet 变换和训练分类器的时间消耗相较之更少.

Table 3 Time consuming comparison

表 3 耗时对比

数据集	IG	FS	**S4**	**ES4**
Adiac	20 106.6	5 066.7	**214.4**	**250.7**
Beef	4 951.3	435.8	**9.6**	**9**
Chlorine.	23 667.8	9 627.2	**150.8**	**150.1**
Coffee	712.9	88.9	**0.4**	**1.5**
Diatom.	438.9	2 842.5	**0.4**	**1.1**
ECGFiveDays	15.7	1 261.2	**0.4**	**0.3**
FaceFour	1 228.4	606.7	**1**	**2**
GunPoint	143	160.2	**0.4**	**1**
ItalyPower.	0.5	87	**0.1**	**0.1**
Lightning7	5 995.4	500.6	**6**	**7.4**
MoteStrain	1.9	676.6	**0.1**	**0.2**
Sony.	1	238.2	**0.1**	**0.1**
Symbols	1 656.8	7 792.3	**2.1**	**3.2**
Synthetic.	153.6	145.5	**1**	**24.9**
Trace	6 946.4	234.4	**6.8**	**6.2**
TwoLeadECG	2.3	443.9	**0.1**	**0.1**

5.5 分类耗时对比

在分类时间上,ES4 相比 S4 则占绝对优势.因为 ES4 通过投票机制大幅减少了 Shapelet 数量,使得对原始数据进行 Shapelet 变换的计算量减少,且分类器去除了很多冗余的特征,所以这部分速度加快了很多.

由于分类时间的差异随着测试集越多越明显,因此选择 3 个大测试集的分类时间做对比,可以更方便地观察.如图 5 所示:横轴为所选数据集“ChlorineConcentration”,“MoteStrain”,“Symbols”,它们的测试集大小分别为 3840, 1252,995;纵轴为分类时间,单位为 s.图中可以看到,ES4 的分类速度远远快于 S4.

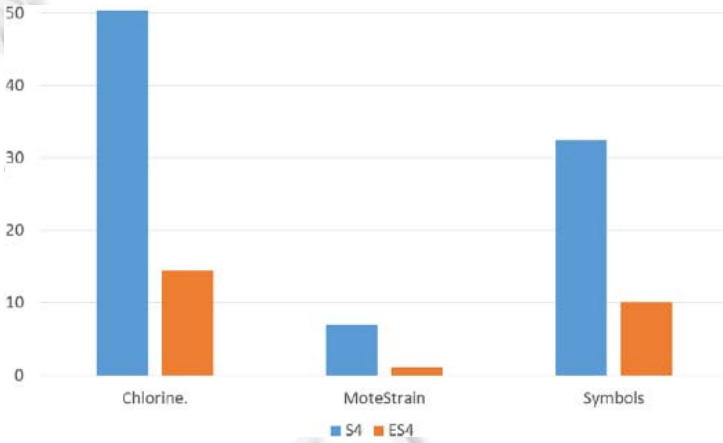

Fig.5 Classification time comparison between S4 and ES4

图 5 S4 与 ES4 的分类时间对比

5.6 ES4算法在不同长度时间序列下的性能变化

我们分别从数据集“Chlorine.”和“Trace”截取不同长度的时间序列用以验证时间序列长度对 ES4 算法性能的影响.我们分别测试了两个数据集在长度的 50%,60%,70%,80%,90%,100%下的训练时间和测试时间,实验结果如图 6、图 7 所示.图中显示,训练时间和测试时间随着时间序列长度的增长基本呈现线性增长的趋势.因而 ES4 算法可以适应较长的时间序列数据集(该组实验的硬件环境是一台 Intel i5 CPU 2.30GHz 和 8GB 内存的 MacBook Pro 主机).

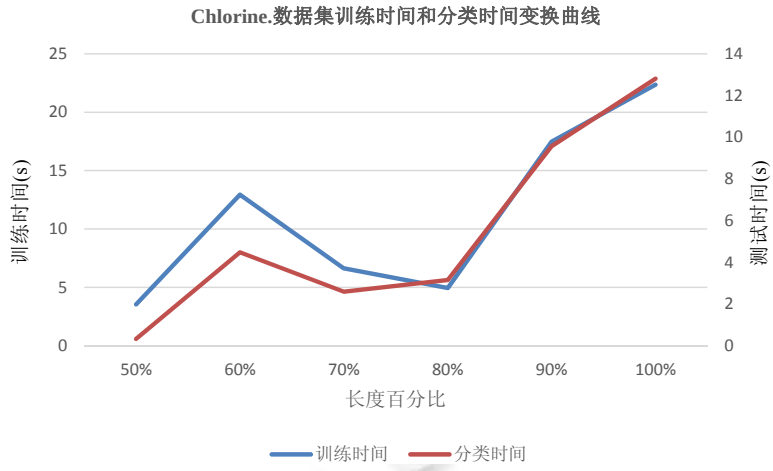

Fig.6 Trainning time and test time with difierent length for Chlorine

图 6 Chlorine 数据集不同长度下的训练时间和测试时间

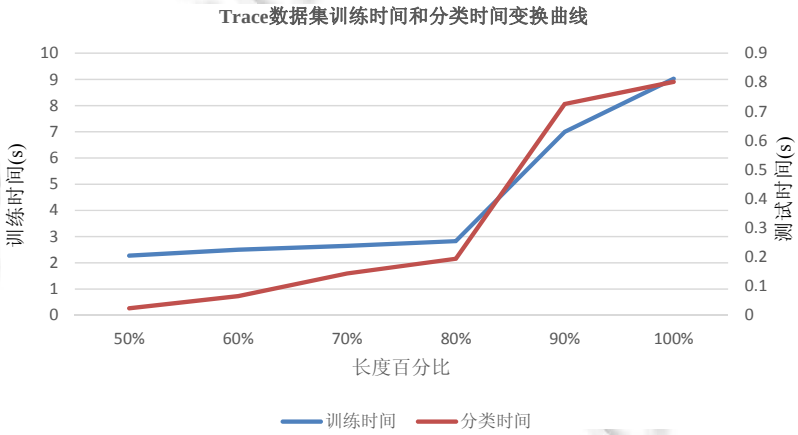

Fig.7 Trainning time and test time with difierent length for Trace

图 7 Trace 数据集不同长度下的训练时间和测试时间

5.7 ES4算法对分类可解释性的增强

Shapelet 可以看作一个可以识别的特征,当只有一个特征的时候,可以用有类似的和没有类似的直接区分开;当特征增多时,可以采用类似程度表示.由于使用时间序列子序列的聚类结果作为 Shapelet 候选集这种方法会引入大量相似 Shapelet,从而导致可解释性的降低.ES4 算法首先是对训练集中的时间序列的子类(而不是子序列)进行聚类,以每个类别得到的若干聚类中心去做选择性抽取,从而能够学习出原序列没有的(但更具有特征指向意义的)Shapelet,保证了学习方法的优势;同时,在子序列选取中,由于考虑重要点等因素,避免了无意义 shapelet 被选取,从而增强了可解释性.

5.8 ES4算法与即时分类算法训练时间比较

ES4 算法的优势在于保证准确率的前提下,大幅提高训练和分类速度.这在需要即时(just-in-time)构造、实时更新的诸如智能制造场景时间序列分类应用[26,29]尤为必要.为验证本文算法即时构造的可行性,我们对 ES4 算法和文献[29]中所采用的 FSS 算法进行了训练时间对比.我们选择的对比数据均是传感器数据,且本文算法在前述对比实验中的分类准确度也是最高的.与文献[29]的训练时间对比见表 4.从表中可以看出,ES4 和 FSS 训

练时间处于同一数量级,并且大部分情况下 ES4 效果要优于 FSS.实验结果表明了 ES4 算法可以满足智能制造等场景下的即时时间序列分类需求(该组实验的硬件环境是一台 Intel i5 CPU 2.30GHz 和 8GB 内存的 MacBook Pro 主机).

Table 4 Time consuming comparison between FSS and ES4

表 4 FSS 和 ES4 训练时间对比

数据集	FSS	ES4
Chlorine	6.66	22.32
Coffee	0.30	0.23
ECGFiveDays	0.30	0.28
Trace	20.01	9.02
TwoLeadECG	0.09	0.07

6 总 结

本文针对 Shapelet 学习算法存在的可解释性差、训练时间长这两个问题展开研究,提出了两个优化策略.通过对原始训练集采用时间序列聚类,可以得到原始时间序列中没有的 Shapelet;同时,在选择性提取算法中加入投票机制,可以解决产生 Shapelet 过多的问题.实验结果证明,本文提出的方法可以在保证 Shapelet 学习算法较高准确率不变的同时,有效加速 Shapelet 发现过程.在实验部分,我们验证了本文方法的有效性,并验证了 ES4 算法能够满足智能制造场景下的时间序列分类器即时构造、实时更新的需求.但时间序列数据分类问题面临的挑战还有很多,比如 ES4 算法的速度与数据集中类的个数有关,当类别过多时,ES4 仍然面临速度太慢的问题.未来我们将考虑引入符号化时间序列技术与 ES4 结合,以此缓解使用原始数值类型的时间序列去直接计算重要子序列相似性向量的高耗时问题;未来,我们也将以 ES4 为基础构造时间序列即时分类服务,并将其应用于工业实际应用场景.

References:

[1] Ye L, Keogh E. Time series Shapelets: A new primitive for data mining. In: Proc. of the 15th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM. 2009. 947−956. [doi: 10.1145/1557019.1557122]

[2] Yuan JD, Wang ZH, Han M. Shapelet pruning and Shapelet coverage for time series classification. Ruan Jian Xue Bao/Journal of Software, 2015,26(9):2311−2325 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4702.htm [doi: 10.13328/j.cnki.jos.004702]

[3] Yan WH, Li GL. Research on time series classification based on Shapelet. Computer Science, 2019,46(1):36−42 (in Chinese with English abstract).

[4] Rakthanmanon T, Keogh E. Fast Shapelets: A scalable algorithm for discovering time series Shapelets. In: Proc. of the 2013 SIAM Int'l Conf. on Data Mining. Philadelphia: Society for Industrial and Applied Mathematics, 2013. 668−676.

[5] Lines J, Davis LM, Hills J, Bagnall A. A Shapelet transform for time series classification. In: Proc. of the 18th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM, 2012. 289−297. [doi: 10.1145/2339530.2339579]

[6] Hills J, Lines J, Baranauskas E, Mapp J, Bagnall A. Classification of time series by Shapelet transformation. Data Mining and Knowledge Discovery, 2014,28(4):851−881. [doi: 10.1007/s10618-013-0322-1]

[7] Grabocka J, Schilling N, Wistuba M, Schmidt-Thieme L. Learning time-series Shapelets. In: Proc. of the 20th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM, 2014. 392−401. [doi: 10.1145/2623330.2623613]

[8] Mueen A, Keogh E, Young N. Logical-Shapelets: An expressive primitive for time series classification. In: Proc. of the 17th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM, 2011. 1154−1162.

[9] Chang KW, Deka B, Hwu WMW, Roth D. Efficient pattern-based time series classification on GPU. In: Proc. of the 2012 IEEE 12th Int'l Conf. on Data Mining. New York: IEEE, 2012. 131−140. [doi: 10.1109/ICDM.2012.132]

[10] He Q, Zhi D, Zhuang F, Shang T, Shi Z. Fast time series classification based on infrequent Shapelets. In: Proc. of the 2012 11th Int'l Conf. on Machine Learning and Applications. New York: IEEE, 2012. 215−219. [doi: 10.1109/ICMLA.2012.44]

[11] Wistuba M, Grabocka J, Schmidt-Thieme L. Ultra-fast Shapelets for time series classification. ArXiv preprint arXiv: 1503.05018, 2015.

[12] Hou L, Kwok JT, Zurada JM. Efficient learning of timeseries Shapelets. In: Proc. of the 30th AAAI Conf. on Artificial Intelligence. Palo Alto: AAAI, 2016. 1209−1215.

[13] Yang Y, Deng Q, Shen F, Zhao J, Luo C. A Shapelet learning method for time series classification. In: Proc. of the 2016 IEEE 28th Int'l Conf. on Tools with Artificial Intelligence. New York: IEEE, 2016. 423−430. [doi: 10.1109/ICTAI.2016.68]

[14] Yeh CCM, Zhu Y, Ulanova L, Begum N, Ding Y. Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In: Proc. of the 2016 IEEE 16th Int'l Conf. on Data Mining. New York: IEEE, 2016. 1317−1322. [doi: 10.1109/ICDM.2016.89]

[15] Yeh CCM, Zhu Y, Ulanova L, Begum N, Ding Y, Dau HA, Zimmerman Z, Silva DF, Mueen A, Keogh E. Time series joins, motifs, discords and Shapelets: A unifying view that exploits the matrix profile. Data Mining and Knowledge Discovery, 2017,32(1): 83−123. [doi: 10.1007/s10618-017-0519-9]

[16] Zhao C, Liu S, Pan L, Ji C, Yang C. Selecting superior candidates from a suitable set: A selective extraction algorithm for accelerating Shapelet discovery in time series data. In: Proc. of the 2019 IEEE 23rd Int'l Conf. on Computer Supported Cooperative Work in Design. New York: IEEE, 2019. 404−409.

[17] Dau HA, Keogh E, Kamgar K, Yeh CCM, Zhu Y, Gharghabi S, Ratanamahatana CA, Chen Y, Hu B, Begum N, Bagnall A, Mueen A, Batista G. The UCR time series classification archive. URL, 2018. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/

[18] Sathianwiriyakhun P, Janyalikit T, Ratanamahatana CA. Fast and accurate template averaging for time series classification. In: Proc. of the 2016 8th Int'l Conf. on Knowledge and Smart Technology. New York: IEEE, 2016. 49−54.

[19] Ji C, Zhao C, Pan L, Liu S, Yang C, Wu L. A fast Shapelet discovery algorithm based on important data points. Int'l Journal of Web Services Research, 2017,14(2):67−80. [doi: 10.4018/IJWSR.2017040104]

[20] Ji C, Liu S, Yang C, Pan L, Wu L, Meng X. A Shapelet selection algorithm for time series classification: New directions. Procedia Computer Science, 2018,129:461−467. [doi: 10.1016/j.procs.2018.03.025]

[21] Ji C, Zhao C, Liu S, Yang C, Pan L, Wu L, Meang X. A fast Shapelet selection algorithm for time series classification. Computer Networks, 2019,148:231−240. [doi: 10.1016/j.comnet.2018.11.031]

[22] Zhao C. A fast time series Shapelet discovery algorithm combining selective extraction and subclass clustering [MS. Thesis]. Ji'nan: Shandong University, 2019 (in Chinese with English abstract).

[23] Furao S, Hasegawa O. An incremental network for on-line unsupervised classification and topology learning. Neural Networks, 2006,19(1):90−106. [doi: 10.1016/j.neunet.2005.04.006]

[24] Furao S, Ogura T, Hasegawa O. An enhanced self-organizing incremental neural network for online unsupervised learning. Neural Networks, 2007,20(8):893−903. [doi: 10.1016/j.neunet.2007.07.008]

[25] Keogh E, Chu S, Hart D, Pazzani M. Segmenting time series: A survey and novel approach. In: Proc. of the Data Mining in Time Series Databases. 2004. 1−21. [doi: 10.1142/9789812565402_0001]

[26] Ji C. Time series classification methods based on Shapelet [Ph.D. Thesis]. Ji'nan: Shandong University, 2017 (in Chinese with English abstract).

[27] Bagnall A, Bostrom A, Large J, Lines J. The great time series classification bake off: An experimental evaluation of recently proposed algorithms. Extended version. arXiv preprint arXiv:1602.01711, 2016.

[28] Bagnall A, Lines J, Bostrom A, Large J, Keogh E. The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery, 2017,31(3):606−660. [doi: 10.1007/s10618-016-0483-9]

[29] Ji C, Zhao C, Pan L, Liu S, Yang C, Meng X. A just-in-time shapelet selection service for online time series classification. Computer Networks, 2019,157:89−98. [doi: /10.1016/j.comnet.2019.04.020]

附中文参考文献:

[2] 原继东,王志海,韩萌.基于 Shapelet 剪枝和覆盖的时间序列分类算法.软件学报,2015,26(9):2311−2325. http://www.jos.org.cn/1000-9825/4702.htm [doi: 10.13328/j.cnki.jos.004702]

[3] 闫汶和,李桂玲.基于 shapelet 的时间序列分类研究.计算机科学,2019,46(1):36–42.
[22] 赵超.融合选择性提取与子类聚类的快速时间序列 Shapelet 发现算法[硕士学位论文].济南:山东大学,2019.
[26] 嵇存.基于 Shapelet 的时间序列分类方法研究[博士学位学位].济南:山东大学,2017.

赵超(1995－),男,陕西兴平人,硕士,主要研究领域为时间序列数据分析,云计算.

王腾江(1977－),男,硕士,主要研究领域为企业管理软件,企业大数据,移动互联网.

刘士军(1972－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为服务计算,企业服务计算和制造数据分析.

潘丽(1982－),女,博士,副教授,博士生导师,CCF 专业会员,主要研究领域为云计算,云制造,市场导向资源分配.

嵇存(1989－),男,博士,讲师,CCF 专业会员,主要研究领域为时间序列数据分析,企业服务计算,制造服务系统配.