基于逻辑 shapelets 转换的时间序列分类算法

原继东[1),2)] 王志海[1),2)] 韩 萌[1),2)] 游 洋[3)]

[1)](北京交通大学计算机与信息技术学院 北京 100044)

[2)](交通数据分析与挖掘北京市重点实验室 北京 100044)

[3)](西北大学电子工程与计算机系 埃文斯顿 美国 60201)

摘 要 时间序列 shapelets 是序列之中最具有辨别性的子序列. 解决时间序列分类问题的有效途径之一是通过 shapelets 转换技术,将其发现与分类器的构建相分离,其主要优点是优化了 shapelets 的选择过程并能够灵活应用不同的分类策略. 但此方法也存在不足,仅仅简单地应用这些 shapelets 而忽略它们之间的逻辑组合关系,有可能降低分类的效果;另外,离线式的发现 shapelets 的过程是相当耗时的. 文中针对后一个问题,采用了一种基于智能缓存的计算重用技术,将发现 shapelets 的时间复杂度降低了一个数量级. 在此基础上,作者提出了一种基于合取或析取的逻辑 shapelets 转换方法,并通过在多个经典的基准数据集上测试,表明了该方法能够在提升分类准确性的同时保持 shapelets 所具有的解释力.

关键词 时间序列;分类;shapelets;逻辑 shapelets

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2015.01448

A Logical Shapelets Transformation for Time Series Classification

YUAN Ji-Dong[1),2)] WANG Zhi-Hai[1),2)] HAN Meng[1),2)] YOU Yang[3)]

[1)](*School of Computer and Information Technology, Beijing Jiaotong University, Beijing* 100044)

[2)](*Beijing Key Laboratory of Traffic Data Analysis and Mining, Beijing* 100044)

[3)](*Department of Electrical Engineering and Computer Science, Northwestern University, Evanston* 60201, *USA*)

Abstract The shapelets of time series are subsequences of time series that are representative of class members. One of the most promising approaches to solve problems of time series classification is to separate the process of finding shapelets from classification algorithms by applying a shapelet transformation. The main advantages of this approach are that it could optimize the process of shapelets selection, and then various classification strategies might be adopted. However, there also exist some limitations in the process. First, only directly employing these shapelets, while ignoring logical relationship between them, the classification accuracy may be cut down. Second, the process is much more time consuming significantly, even though shapelets are computed offline. In this paper, the latter problem is addressed by using an intelligent caching based and reusable skill, which reduces the time complexity of finding shapelets by an order of magnitude. On this basis, a novel transformation that is based on conjunctive or disjunctive of shapelets is proposed. Experimental results have shown the efficiency of logical shapelets transformation on classic benchmark datasets used for these problems, which can improve classification accuracy, and whilst retaining their interpretability.

Keywords time series; classification; shapelets; logical shapelets

收稿日期:2013-10-20;最终修改稿收到日期:2014-12-09. 本课题得到中央高校基本科研基金(2015YJS049)资助.**原继东**,男,1989 年生,博士研究生,主要研究方向为数据挖掘和模式识别. E-mail: 12112078@bjtu.edu.cn. **王志海**,男,1963 年生,博士,教授,博士生导师,主要研究领域为数据挖掘和机器学习. **韩 萌**,女,1982 年生,博士研究生,讲师,主要研究方向为数据挖掘和机器学习. **游 洋**,男,1991 年生,硕士研究生,主要研究方向为下一代互联网技术与数据处理方法.

1 引　言

一条时间序列是一组序列数据，它通常是在相等间隔的时间段内，依照给定的采样率，对某种潜在过程进行观测的结果. 时间序列数据广泛地存在于商业、农业、气象、生物科学等诸多领域. 在时间序列分类问题中，我们将任意实值型有次序的数据看作一条时间序列[1]. 它与传统分类问题之间的差别在于，对于传统分类问题而言，属性次序是不重要的，并且变量之间的相互关系独立于它们的相对位置；而对于时间序列数据而言，在寻找最佳的辨别性特征时，变量的次序起着至关重要的作用[2]. 因此，时间序列分类问题已成为机器学习领域的特殊挑战之一.

在过去的十多年中，针对时间序列分类问题的研究主要集中在基于不同距离度量方式的最近邻(1-NN)算法上，如基于欧氏距离或者动态时间规整(Dynamic Time Wrapping，DTW)的 1-NN 等[3-4]. 近年来，基于时间序列 shapelets 的分类算法引起了相关研究者极大的兴趣. 所谓的时间序列 shapelets 是序列之中最具有辨别性的子序列[5]. 最初的基于 shapelets 的分类算法是由 Ye 等人[5-6]所提出的，其采用信息增益度量数据的分裂点，并通过递归搜索最具有辨别性的 shapelets 来构建决策树. 如图 1 所示，对于经典的 *Gun/NoGun*(手中有枪/没枪)问题[5]，最佳 shapelet 如粗黑线标注部分所示，它能够对 *Gun/NoGun* 数据集进行准确的分类. Mueen 等人[7]提出了使用逻辑 shapelets 来构建决策树的思想，并采用加速技术和剪枝策略来提升文献[5]中算法的可解释性和运行效率. Rakthanmanon 等人[8]针对原有的 shapelets 发现算法在时间效率上的不足，提出了一种基于 SAX(Symbolic Aggregate approXimation)表示的快速 shapelets 发现算法，此算法在提升发现 shapelets 速率的同时保证了一定的分类准确性.

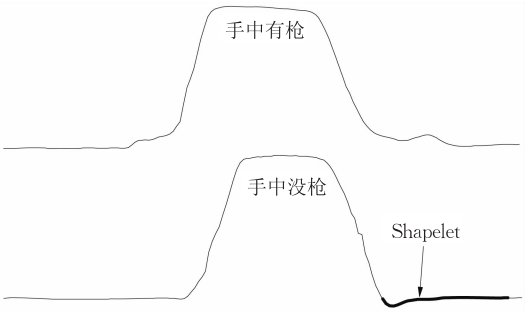

图 1　Shapelet 示例

相比较于具有较高分类准确率且易于实现的 1-NN 分类器，时间序列 shapelets 的优点主要在于两个方面.

(1)分类结果的可解释性更强. 原因是 1-NN 分类器只能说明被分类对象与所分到的类别间具有较大的相似性，但没有指出它与其他类别之间具体的不同点，而 shapelets 可以较为充分地解释各类之间的差异(如图 1 所示)，即具有较强的辨别性.

(2)分类速度更快. 首先，由于 shapelets 比较的是子序列(一般情况下比原始的时间序列要短)之间的相似性，所以不像基于实例的方法那样需要对比整个数据集. 其次，shapelets 能够非常简洁紧凑地表示一个类的概念，有时我们仅需要一个 shapelet 就可以出色地完成分类任务，此简洁性意味着分类所需要的时间和空间大量地减少，所以 shapelets 具有较快的分类速度.

上述的几种方法都是在发现 shapelets 的同时构建分类器. 而 Bagnall 等人[2]强调了数据转换在时间序列分类中的重要性，其为解决数据转换后可能引起的准确率大幅度差异问题，提出了一种集成学习技术来提升分类性能.

Lines 等人[1]提出了一种基于 shapelets 转换的时间序列分类方法，即将 shapelets 的发现与分类器的构建过程相分离，其主要优点是优化了 shapelets 的选择过程并能够灵活应用不同的分类策略. 另外，shapelets 也被广泛应用于时间序列聚类[9]、早期分类问题[10]和姿势辨别问题[11]等等.

尽管基于时间序列 shapelets 的方法是目前解决时间序列分类问题的有效途径之一，它仍存在着许多不足. 以 Lines 等人提出的基于 shapelets 转换的方法为例，一是仅仅简单地应用这些 shapelets 而忽略它们之间的逻辑组合关系，有可能降低分类的效果. 如图 2 所示，单个 shapelet 无法对 Sony AIBO

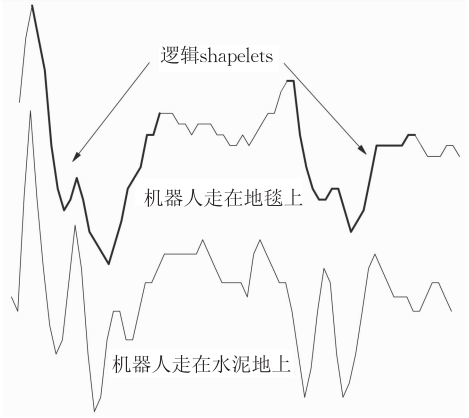

图 2　逻辑 shapelets 示例

Robot 数据集进行严格区分,此时需要联合多个 shapelets,即使用逻辑 shapelets 才能准确区分两种类别;二是尽管发现 shapelets 的过程是离线计算的,它仍然需要消耗大量的时间.

针对上述两点不足,本文的主要贡献在于两个方面.首先是应用了一种基于智能存储和计算重用的技术,将发现 shapelets 的时间复杂度降低了一个数量级.其次,我们在此基础上提出了一种基于合取或析取的逻辑 shapelets 转换方法.具体做法是首先找到 k 个最好的逻辑 shapelets,然后通过计算这些逻辑 shapelets 与原有时间序列间的距离,将原有时间序列转换成新的拥有 k 个属性的数据,同时也把时间序列分类问题转化成了传统的分类问题.此方法在保持 shapelets 的解释力的同时提升了分类的准确性.

本文第 2 节将阐述时间序列分类问题的相关定义和符号;在第 3 节,将回顾一下文献[1]中提出的算法;第 4 节将阐述本文所应用的加速技术以及基于逻辑 shapelets 转换的时间序列分类算法的基本思想;在第 5 节,本文将进行实验描述并评估所提出的算法;第 6 节进行相应的总结以及对进一步工作的展望.

2 符号与定义

将时间序列分类问题定义为从一系列标定的时间序列训练集上建造分类器的过程.假设有 n 条时间序列,$D=\{T_1,T_2,\cdots,T_n\}$,每一条时间序列有 m 个观测值和一个类值 c_i,即 $T_i=\langle t_{i1},t_{i2},\cdots,t_{im},c_i\rangle$.目标是构造一个将时间序列的观测值映射到类值的函数.为方便起见,假设每一条时间序列具有相同数目的观测值.文中涉及的主要符号如表 1 所示.

表 1 符号表

符号	释义
D	时间序列数据集
T,X,Y	时间序列
S,s	时间序列的子序列
m	时间序列的长度
$dist(x,y)$	时间序列间的规范化距离
$subdist(x,y)$	子序列距离
$C(x,y)$	相关系数
$\mu_x,\sigma_x,\mu_y,\sigma_y$	均值与方差
$N,\|D\|$	时间序列的个数
c,C	类值和类的个数
(s,t)	分裂点
E	熵
I	信息增益
G	分裂间隔

定义 1. 时间序列及其子序列.

时间序列 T 是一条长度为 m 的实值序列,可表示为 $t_1,t_2,\cdots,t_m$.一条时间序列的子序列 $S_{i,l}=t_i,t_{i+1},\cdots,t_{i+l-1}$ 是一个 T 中从位置 i 开始长度为 l 的连续的子序列,其中 $1\leq l\leq m$,$1\leq i\leq m-l+1$.

注意,数据 $t_1,t_2,\cdots,t_m$ 是按照时间顺序排列的,两两之间具有相同的时间间隔.时间序列的子序列可以当作是时间序列的局部特征.假设一条时间序列的长度为 m,其子序列的最小长度为 1,最大长度为 m,那么它能有 $m(m+1)/2$ 个子序列.

定义 2. 时间序列间的距离.

所有的分类问题都依赖于数据间的相似性度量,时间序列分类问题也不例外.对于给定的具有相同长度 m 的两条时间序列 X 和 Y,我们可以使用规范化的欧氏距离来度量它们之间的不同.如式(1)所示.

$$dist(x,y)=\sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_i-y_i)^2} \tag{1}$$

$$x_{\text{norm}}=\frac{x-\overline{X}}{\delta_X} \tag{2}$$

注意,为获得两条时间序列之间距离的有效比对并保证缩放和偏移的不变性,在计算真正的距离之前,必须使用 z-规范化方法(如式(2)所示)对每条时间序列进行规范化处理[12].因为缩放和偏移上甚至是一丁点的不同都有可能对形状的相似性比对造成巨大的影响,所以这是至关重要的一步.此处 $\overline{X}$ 为时间序列观测值的均值,δ_x 为时间序列观测值的标准差,x_{norm} 为 z-规范化后的时间序列观测值.

从式(1)可以得到,两条时间序列间距离计算的复杂度和时间序列的长度呈线性关系.相比之下,将采用从 X 和 Y 推演而得到的 5 个数据来计算 X 与 Y 之间的规范化欧氏距离,这些数据被称为充分统计量(sufficient statistics)[7,13].它们分别为 $\sum x$,$\sum y$,$\sum x^2$,$\sum y^2$ 和 $\sum xy$.通过使用此方式计算规范化的欧氏距离,可以重新利用计算的中间结果并降低计算规范化欧氏距离的时间复杂度.本文将在第 4 节进行具体的阐述,相应的计算公式如下.

一条时间序列 X 的均值和方差可以通过式(3)和(4)计算.

$$\mu_x=\overline{X}=\frac{1}{m}\sum x \tag{3}$$

$$\delta_x^2=\frac{1}{m}\sum(x-\mu_x)^2 \tag{4}$$

X 与 Y 之间的欧氏距离可以通过式(5)和(6)计算.

$$C(x,y)=\frac{\sum xy-m\mu_x\mu_y}{m\delta_x\delta_y} \tag{5}$$

$$dist(x,y)=\sqrt{2(1-C(x,y))} \tag{6}$$

许多时间序列数据挖掘算法(如 1-NN 分类、聚类算法等)只需要对比相等长度的时间序列. 相比之下,时间序列 shapelets 需要测试一个短的时间序列(即 shapelet)是否在某一特定阈值下包含于一个更长的时间序列中. 为得到此结果,短的时间序列必须在长的序列上滑动以得到它们之间的最佳比对结果,称此距离度量为子序列距离,并定义为

$$subdist(x,y)=\min(dist(x,y_{|x|})) \tag{7}$$

或者根据充分统计量可表示为

$$sufficientdist(x,y)=\sqrt{2(1-C_s(x,y))} \tag{8}$$

这里 X 与 Y 是两个长度分别为 m 和 n 的时间序列($m\leq n$),$y_{|x|}$ 表示时间序列 Y 中长度为 $|x|$ 的子序列. 相应地

$$C_s(x,y)=\min_{0\leq l\leq n-m}\frac{\sum_{i=1}^{m}x_iy_{i+l}-m\mu_x\mu_y}{m\delta_x\delta_y} \tag{9}$$

在上述定义中,μ_y 和 δ_y 分别表示 Y 中从 $l+1$ 位置开始的 m 个连续值的均值和标准差. 注意,*subdist*( ) 和 *sufficientdist*( ) 得到的是两个序列间的最短距离.

定义 3.　熵.

假设数据集 D 中包含 N 条时间序列,类的个数为 C,类 c_i 在数据集 D 中有 n_i 个实例,并且 $n_1+n_2+\cdots+n_C=N$. 由于本文采用信息增益度量来评价 shapelets 的辨别能力,首先需要在此回顾一下熵的概念. 数据集 D 的熵定义为

$$E(D)=-\sum_{i=1}^{C}\frac{n_i}{N}\log\frac{n_i}{N} \tag{10}$$

定义 4.　分裂点.

一个分裂点为一个元组(s,t),这里 s 表示一个子序列,t 表示距离的阈值. 一个分裂点(s,t)将数据集 D 分割成两个不相交的子集,其中 $D_{\text{left}}=\{x: x\in D, subdist(s,x)\leq t\}$,$D_{\text{right}}=\{x: x\in D, subdist(s,x)>t\}$,并且 $N_1=|D_{\text{left}}|$,$N_2=|D_{\text{right}}|$.

定义 5.　信息增益.

一个分裂点(s,t)的信息增益为

$$I(s,t)=E(D)-\frac{N_1}{N}E(D_{\text{left}})-\frac{N_2}{N}E(D_{\text{right}}) \tag{11}$$

定义 6.　分裂间隔.

一个分裂点的分裂间隔为

$$G(s,t)=\frac{1}{N_2}\sum_{x\in D_{\text{right}}}subdist(s,x)-\frac{1}{N_1}\sum_{x\in D_{\text{left}}}subdist(s,x) \tag{12}$$

即为分裂点右侧实例的平均距离减去分裂点左侧实例的平均距离.

定义 7.　Shapelet.

数据集 D 中的一个 shapelet 是由 D 中某一实例 T 的一个子序列 S 和一个分裂点的阈值 t 所组成的元组(S,t),它试图将数据集 D 分割成两个不同的群组.

现在将用图 3 所示的例子来解释 shapelet 的概念. 左边方框中是一个候选的 shapelet,右边的 *orderline* 记录了该候选 shapelet 与数据集 D 中每条时间序列之间的距离,t 为最佳的分裂阈值. 在图 3 中,数据集 D 为经典的 *Gun/NoGun* 数据集,圆形表示 *NoGun* 类时间序列与候选 shapelet 之间的距离,它们与候选 shapelet 间的距离较小;矩形表示 *Gun* 类时间序列与候选 shapelet 之间的距离,它们与候选 shapelet 间的距离较大. *orderline* 创建之后,就可以计算候选 shapelet 的分裂点、分裂间隔以及信息增益等. 最终选取最优的一个或多个 shapelets 进行分类.

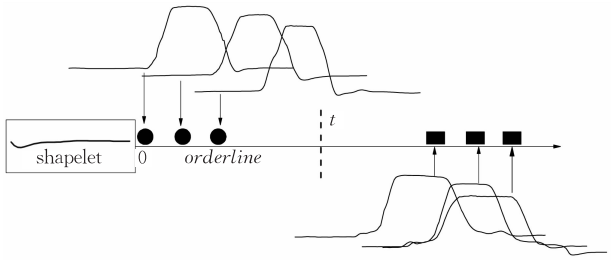

图 3　一个 shapelet 的 *orderline* 示例

需要注意的是,当我们对 shapelets 排序时,优先选择信息增益较大的 shapelet;若两个 shapelet 的信息增益相同,则优先选择分裂间隔较大的 shapelet;若两者的分裂间隔也相同,则优先选择长度较短的 shapelet;若两者长度相等,则优先选择先产生的 shapelet. 也就是说,使用分裂间隔以及 shapelet 的长度等作为打破平局的方法.

3　Shapelets 转换技术

Lines 等人[1]提出了一种基于 shapelets 转换的时间序列分类方法,此方法的主要贡献是将 shapelets 的发现与分类器的构建相分离,避免了文献[5]中发现 shapelets 时贪心的递归搜索过程,并允许转换后的数据应用于不同的分类器中. 相应的算法实现主要包括 3 个步骤:首先,通过一次扫描数据集,找到最好的 k 个 shapelets;其次,将每一条时间序列转

换成具有 k 个属性的实例.即让每个属性代表一个 shapelet,属性值为该 shapelet 与时间序列间的距离,这样就创建了一个新的转换后的数据集;最后,将转换后的数据集用于贝叶斯网络、决策树、1-NN 等分类器进行分类.具体的描述见算法 1.

算法 1. Shapelets 发现算法 $ShapeletFilter(D, min, max, k)$.

输入:时间序列数据集 D,最小长度 min,最大长度 max, shapelets 的个数 k

输出:k 个最好的 shapelets $kShapelets$

1. $kShapelets \leftarrow \varnothing$;
2. FOR $i \leftarrow 0$ to $|D|$ DO {D 中的每一条时间序列 T_i}
3. $shapelets \leftarrow \varnothing$;
4. FOR $l \leftarrow min$ to max DO {T_i 中的每一个长度}
5. FOR $u \leftarrow 0$ to $|T_i| - l + 1$ DO {每一个起始位置}
6. $S \leftarrow T_{i(u,l)}$;
7. FOR $m \leftarrow 0$ to $|D|$ DO {计算候选 shapelet S 与每一条时间序列的距离}
8. $D_s \leftarrow \textbf{subdist}(S, T_m)$;
9. $orderline \leftarrow sort(D_s)$;
10. $quality \leftarrow assessCandidate(S, orderline, D_s)$;
11. $shapelets.add(S, quality)$;
12. $sortByQuality(shapelets)$;
13. $removeSelfSimilar(shapelets)$;
14. $kShapelets \leftarrow merge(k, kShapelets, shapelets)$;
15. RETURN $kShapelets$;

算法 1 描述了从数据集中抽取 k 个最好的 shapelets 的过程.根据 min 和 max 参数设置的范围,每一条时间序列的每一种可能长度的子序列都被做了相应的评估,并最终得到最好的 k 个 shapelets.初始时 $kShapelets$ 为空(第 1 行).得到一个候选的 shapelet 之后,需要计算该 shapelet 与每一条时间序列间的距离并构建 $orderline$(第 7～9 行).如算法描述的第 9～11 行所示,得到一个 shapelet 的 $orderline$ 之后,将对其评估并得到它的最大信息增益,然后将其加入到候选 shapelets 中.在对候选 shapelets 进行排序时(第 12 行),主要根据其信息增益的大小进行排列,具体打破平局的方法参见第 2 节定义 7.若两个 shapelets 来自同一条时间序列并有重叠之处,则认为它们是自相似的,此时需要移除自相似的 shapelets(第 13 行).一旦得到一条时间序列中所有不相似的 shapelets,就将它们和现有的最好的 shapelets 相结合,并保留最好的 k 个(第 14 行).

候选 shapelets 与时间序列间距离的计算方法如算法 2 所示.Shapelets 与时间序列比对时,都进行了规范化处理(第 2,5 行).需要注意的是,在计算距离之前重复地进行规范化会消耗大量的时间,而且同一个序列可能会被规范化多次,而本文将在 4.1 节所介绍的技术成功避免了这一点.

算法 2. 规范化子序列距离 $subdist(x, y)$.

输入:时间序列 x 和 y,$|x| \leq |y|$

输出:x 与 y 的规范化距离

1. $bestSum \leftarrow MAX\_VALUE$;
2. $x \leftarrow zNorm(x)$;
3. FOR $i \leftarrow 0$ to $|y| - |x| + 1$ DO {y 中的每一个起始位置}
4. $sum \leftarrow 0$;
5. $z \leftarrow zNorm(y_{i,|x|})$;
6. FOR $j \leftarrow 0$ to $|x|$ DO{计算欧氏距离}
7. $sum \leftarrow sum + (z_i - x_i)^2$;
8. $bestSum \leftarrow min(bestSum, sum)$;
9. RETURN $(bestSum/|x|)^{1/2}$;

得到 k 个最好的 shapelets 之后,利用它们将原有的时间序列转换成新的数据集的算法描述如算法 3 所示.

算法 3. Shapelets 数据转换 $TransformData(S, D)$.

输入:最好的 k 个 shapelets S,数据集 D

输出:转换后的数据集 $output$

1. $output \leftarrow \varnothing$;
2. FOR $i \leftarrow 0$ to $|D|$ DO {D 中的每一条时间序列 T_i}
3. $transformed \leftarrow \varnothing$;
4. FOR $j \leftarrow 0$ to $|S|$ DO {S 中的每一个 shapelet}
5. $dist \leftarrow \textbf{subdist}(S_j, T_i)$;
6. $transformed.add(dist)$;
7. $output.add(transformed)$;
8. RETURN $output$;

对于 D 中的每一条时间序列 T_i,它的子序列距离通过与 S_j ($j = 0, \cdots, k-1$) 计算得出(第 2～5 行),得到的 k 个距离用于构建转换后数据集,即新的数据集中每一条实例具有 k 个属性,每一个属性值都对应于相应的 shapelet 到原始时间序列的距离(第 6～7 行).

4 加速技术与逻辑 shapelets

下面,将首先介绍一种更加有效率的时间序列间距离的计算方法[7,13],然后阐述逻辑 shapelets 转换的基本思想,并给出具体的算法描述.

4.1 充分统计量

在第 3 节所描述的算法中,对于数据集 D 中的任意一条时间序列 D_x,D_x 中起始于任何位置的任意长度的子序列 S_i 都可能是一个候选的 shapelet.

当计算 S_i 与其他时间序列如 D_y 之间的距离时，需要将 S_i 在 D_y 上逐步滑动来找到最小的距离. 如图 4 所示，对于数据集 D 中的任意两条实例 D_x 和 D_y，S_1 和 S_2 均为长度为 l 的子序列，若要计算 S_1 与 S_2 之间的距离，首先需要对 S_1 与 S_2 进行规范化处理，然后计算 S_1 与 S_2 中每一对实值间的距离并取和. 显然，当 S_1 与 S_2 的长度和起始位置不断变化时(图 4 中的虚线所示)，会造成许多重复的冗余的计算，若能够将这些计算的中间结果存储起来，等需要的时候直接提取出来，运算效率会得到明显的提升，充分统计量方法恰恰做到了这一点.

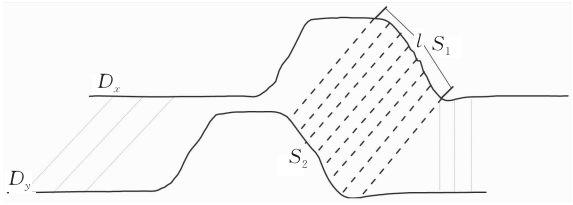

图 4　不同时间序列间的子序列距离

针对每一对时间序列 (D_x, D_y)，可以计算 5 个数组，M, S_x, S_y, S_{x^2} 和 S_{y^2}. 其中数组 S_x, S_y 用于存储时间序列 D_x 和 D_y 中观测值的累加和，即 $\sum x$，$\sum y$；数组 S_{x^2} 和 S_{y^2} 用于存储 D_x 和 D_y 中观测值平方的累加和，即 $\sum x^2$，$\sum y^2$. M 是一个用于存储 D_x 和 D_y 中不同子序列的乘积和 $\sum xy$ 的二维数组. 初始时，所有的数组都被初始化为 0，具体表示如下：

$$S_x[u]=\sum_{i=0}^{u} x_i \tag{13}$$

$$S_y[v]=\sum_{i=0}^{v} y_i \tag{14}$$

$$S_{x^2}[u]=\sum_{i=0}^{u} x_i^2 \tag{15}$$

$$S_{y^2}[v]=\sum_{i=0}^{v} y_i^2 \tag{16}$$

$$M[u,v]=\begin{cases}\sum\limits_{i=0}^{v} x_{i+t}y_i, & u>v\\ \sum\limits_{i=0}^{u} x_i y_{i+t}, & u\le v\end{cases} \tag{17}$$

在式(17)中，$t=|u-v|$，u 和 v 分别为 D_x 和 D_y 中子序列的起始位置. 这 5 个数组并称为 $Stats_{x,y}$. 当计算过 $Stats_{x,y}$ 之后，首先根据它得到序列的均值与方差，然后根据式(5)和(6)计算得到规范化的欧氏距离. 具体的算法描述如算法 4 所示，均值、方差以及任意长度子序列乘积和的计算如下所示.

$$u_x=\frac{S_x[u+l-1]-S_x[u-1]}{l} \tag{18}$$

$$u_y=\frac{S_y[v+l-1]-S_y[v-1]}{l} \tag{19}$$

$$\delta_{x^2}=\frac{S_{x^2}[u+l-1]-S_{x^2}[u-1]}{l}-u_x^2 \tag{20}$$

$$\delta_{y^2}=\frac{S_{y^2}[v+l-1]-S_{y^2}[v-1]}{l}-u_y^2 \tag{21}$$

$$\sum_{i=0}^{l-1} x_{u+i}y_{v+i}=M[u+l-1,v+l-1]-M[u-1,v-1] \tag{22}$$

算法 4.　规范化子序列距离 $sufficientdist(u, l, Stats_{x,y})$.

输入：起始位置 u，长度 l，x 与 y 的充分统计量 $Stats_{x,y}$

输出：$x_{u,l}$ 与 y 的规范化子序列距离

1. $bestSum \leftarrow MAX\_VALUE$;
2. $\{M, S_x, S_y, S_x^2, S_y^2\} \leftarrow Stats_{x,y}$;
3. FOR $i \leftarrow 0$ to $|y|-|x|+1$ DO {y 中的每一个起始位置}
4. $d \leftarrow$ 由式(5)和(6)所得到的距离;
5. $bestSum \leftarrow min(bestSum, d)$;
6. RETURN $(bestSum/|x|)^{1/2}$;

在算法 $sufficientdist(u, l, Stats_{x,y})$ 中，输入为时间序列 D_x 中某一子序列的起始位置 u、长度 l 以及事先计算好的 $Stats_{x,y}$，输出为子序列通过不断地滑动而得到的与 D_y 间的最小距离. 通过与算法 2 比较可以得出，$subdist(x,y)$ 在计算规范化的子序列距离时，最坏情况下的时间复杂度为 $O(m^2)$，而充分统计量的计算方法省略了内部的 for 循环，即只需要 $O(m)$ 的时间复杂度. 所以，依据 $sufficientdist(u, l, Stats_{x,y})$，将获取子序列距离的时间复杂度降低了一个数量级. 由于发现 shapelets 时需要计算每一个候选 shapelet S 与每一条时间序列 T 之间的子序列的距离，所以同时也将发现 shapelets 的时间复杂度降低了一个数量级.

4.2　逻辑 shapalets

一个 shapelet 是由数据集 D 中某一实例的一个子序列 S 和一个分裂点阈值 t 所组成的元组 (S,t). 很容易想到单个 shapelet 不足以区分不同的类别的情景. 如图 5 所示，采用一个简单的人造的例子来进行说明. 数据集 D 中包含两类时间序列，A 类时间序列既包含波峰又包含波谷，B 类时间序列只包含波峰或波谷. 假设 shapelets S_1, S_2, S_3 的分裂阈值分别为 t_1, t_2, t_3，从而可以观察到，(S_1, t_1)，(S_2, t_2)，(S_3, t_3) 都不能很好地将 A, B 类区分开. 当将 (S_1, t_1) 和 (S_2, t_2) 联合起来时，可以发现，A 类和 B 类被明

显地区分了.因此,为达到最好的区分效果,考虑将逻辑操作增加到 shapelets 的转换中.

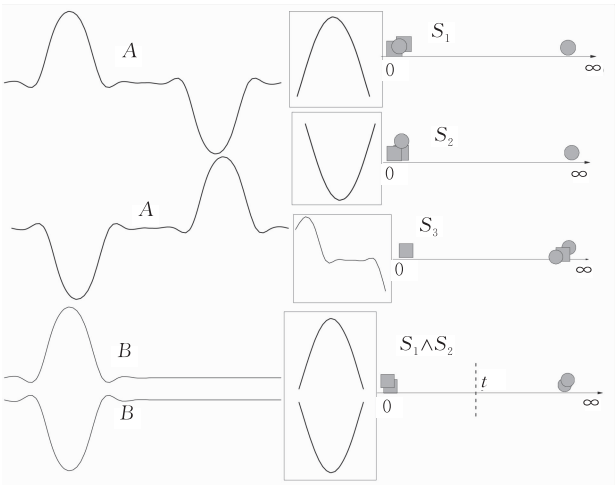

图 5 单个 shapelet 无法区分不同类别的例子

图 5 中所示的逻辑操作是合取操作,用符号"∧"表示,即$(S_1,t_1)\wedge(S_2,t_2)$,同样析取操作用符号"∨"表示.当将逻辑操作加入到 shapelets 的发现过程中之后,有可能出现$(S_1,t_1)\vee(S_2,t_2)\wedge(S_3,t_3)$这样一种析取操作与合取操作同时出现的情况,为保持逻辑操作的简洁性,本文只考虑两种情况,一种情况是只有合取操作,另一种是只有析取操作.具体定义如下.

定义 8. 逻辑 shapelets.

数据集 D 中的一个逻辑 shapelets 是一个元组,由 D 中一个或多个实例的多个子序列所组成,同时包含一个分裂点阈值 t 以及一个逻辑操作标识 opt(opt 大于等于 0 为合取操作,小于 0 为析取操作),它试图将数据集 D 分割成两个不同的群组.其可表示为$(\langle S_1,S_2,\cdots,S_n\rangle,t,opt)$,$n(n\geq 2)$为逻辑 shapelets 中所包含经典 shapelet 的个数,并且 S_1,S_2,…,S_n相互之间不重叠.

4.3 发现逻辑 shapelets

发现 k 个最好的逻辑 shapelets 的算法如算法 5 所示.与算法 1 相比较可以发现,两者的前半部分基本相同,但本文在寻找逻辑 shapelets 时采用了充分统计量技术(第 5~12 行).注意,文献[7]中使用充分统计量来递归地发现当前最好的 shapelets,而本文直接用其来发现 k 个最好的逻辑 shapelets.另外,如第 13~16 行所示,每找到一个候选 shapelet,都对它进行评估,并调用 *LogicalShapeletsCache*() 函数来寻找是否有其他 shapelet 能够与当前 shapelet 相联合,从而提高它的信息增益以及辨别力.当一条时间序列中的所有逻辑 shapelets 都被评估之后,调用 *sortByQuality*()函数进行排序(第 18 行),排序方法与定义 7 中介绍的相一致,然后移除自相似的逻辑 shapelets(第 19 行).若两个逻辑 shapelets 中存在来自同一条时间序列并有重叠之处的 shapelets,则认为它们是自相似的.自相似的逻辑 shapelets 中排序靠后者将被移除掉.一旦得到一条时间序列中所有非自相似的逻辑 shapelets,就将它们和当前最好的逻辑 shapelets 相结合,并保留当前最好的 k 个(第 20 行).注意当调用 *mergelogical*()函数时,仍需要进行自相似性检测,因为不同时间序列的子序列有可能合取或析取同一子序列.

算法 5. 发现逻辑 shapelets *LogicalShapeletFilter*(D,min,max,k).

输入:时间序列数据集 D,最小长度 min,最大长度 max,逻辑 shapelets 的个数 k

输出:k 个最好的逻辑 shapelets *kLGShapelets*

```
1. kLGShapelets←∅;
2. FOR i←0 to |D| DO {D 中的每一条时间序列 T_i}
3.   logicalshapelets←∅;
4.   tempLGShapelets←∅;
5.   x←T_i;
6.   FOR j←0 to |D| DO {计算 x 与 T_j 的充分统计量}
7.     y←T_j;
8.     Stats_{x,y}←{M,S_x,S_y,S_x^2,S_y^2};
9.   FOR l←min to max DO {每一个可能的长度}
10.    FOR u←0 to |T_i|−l+1 DO {每一个起始位置}
11.      FOR m←0 to |D| DO {计算候选 shapelet S 与每一条时间序列的距离}
12.        D_s←sufficientdist(u,l,Stats_{x,y});
13.      orderline←sort(D_s);
14.      quality←assessCandidate(S,orderline,D_s);
15.      tempLGShapelets.add(S);
16.      tempLGShapelets←LogicalShapeletsCache(D,
           min,max,num,tempLGShapelets,orderline);
17.      logicalshapelets.add(tempLGShapelets);
18.  sortByQuality(logicalshapelets);
19.  removeSelfSimilarlogical(logicalshapelets);
20.  kLGShapelets ← mergelogical(k, kLGShapelets,
       logicalshapelets);
21.RETURN kLGShapelets;
```

获取逻辑 shapelets 的算法如算法 6 所示.此算法是一个递归算法,当逻辑 shapelets 中 shapelet 的数目超过阈值 n 或者找不到比当前 shapelet/逻辑 shapelets 更好的逻辑 shapelets 时,递归循环终止(第 1~3 行,18~25 行),所以此算法得到的是合取或析取后能够使当前 shapelet/逻辑 shapelets 的信息增益最大的子序列.本文仍采用充分统计量来计

算子序列距离(第 6～8 行). 若一个候选 shapelet S 与 *LogicalShapelets* 是自相似的，则不再访问此 shapelet(第 9～12 行). 当得到一个候选 shapelet 的 *orderline* 之后，需要将此候选 shapelet 的 *orderline* 与当前 shapelet/逻辑 shapelets 的 *orderline* 相结合，构建逻辑 shapelets 的 *orderline*(第 15～16 行). 但对于逻辑 shapelets 中的合取与析取操作，怎么定义它的 *orderline* 呢？如第 16 行所示，这里采用一种比较简单的方式来联合各个 shapelet 的 *orderline*，从而组成逻辑 shapelets 的 *orderline*. 对于“∧”操作，各个 shapelet 的 *orderline* 上相应实例的最大距离将被当做新的 *orderline* 上的距离；而对于“∨”操作，则选择最小的距离. 注意，这些操作并不影响熵和信息增益的计算.

算法 6.　获取逻辑 shapelets *LogicalShapeletsCache*(D, *min*, *max*, n, *lgshapelet*, *orderline*).

输入：时间序列数据集D，最小长度 *min*，最大长度 *max*，经典shapelet的个数n，子序列*lgshapelet*，*lgshapelet* 与每条时间序列距离的顺序排列 *orderline*

输出：逻辑 shapelets *LogicalShapelets*

1. *LogicalShapelets*←*lgshapelet*;
2. IF $n \leq$ *LogicalShapelets.size*()
3. RETURN *LogicalShapelets*;
4. *BestQuality*←*lgshapelet.quality*;
5. *tempLGShapelets*←∅;
6. FOR i←0 to |D| DO {D 中的每一条时间序列 T_i}
7. FOR j←0 to |D| DO {计算T_i与T_j的充分统计量}
8. $Stats_{x,y}$←{$M, S_x, S_y, S_x^2, S_y^2$};
9. FOR l←*min* to *max* DO {每一个可能的长度}
10. FOR u←0 to |T_i|−l+1 DO {每一个起始位置}
11. IF *selfSimilaritylogical*(*LogicalShapelets*, S)
12. CONTINUE;
13. FOR m←0 to |D| DO {计算候选 shapelet S 与每一条时间序列的距离}
14. D_s←***sufficientdist***(u, l, $Stats_{x,y}$);
15. *lgorderline*←*sort*(D_s);
16. *orderline*←*mergeTwoLines*(*lgorderline*, *orderline*);
17. *quality*←*assessCandidate*(S, *orderline*, D_s);
18. IF *quality.betterthan*(*BestQuality*)
19. *tempLGShapelets*←*lgshapelet*;
20. *tempLGShapelets.add*(S);
21. *BestQuality*←*quality*;
22. *LogicalShapelets*←*tempLGShapelets*;
23. IF *LogicalShapelets.size*()<n
24. *LogicalShapelets*←***LogicalShapeletsCache***(D, *min*, *max*, n, *LogicalShapelets*, *orderline*);
25. RETURN *LogicalShapelets*;

另外，采用逻辑 shapelets 进行时间序列转换时有可能发生过拟合现象. 若所发现的逻辑 shapelets 为$(S_1, t_1) \vee (S_2, t_2) \vee \cdots \vee (S_n, t_n)$，其中$n$为数据集$D$中某一类时间序列(如 A)的实例个数，S_i(i=1, 2, …, n)为类 A 中不同实例的子序列. 此时，在数据转换之后所有类标为 A 的实例与此逻辑 shapelets 所对应的属性值均为 0，并且抽取的逻辑 shapelets 的数量和质量也会因此而降低(小于 k)，进而降低分类的准确性. 为避免过拟合现象，一般情况下，将逻辑 shapelets 中经典 shapelet 的数量设置为 2，并根据数据集的大小进行不断调整，最大数量硬性规定为 5.

4.4　转换时间序列

采用逻辑 shapelets 转换的主要动机之一是为了将时间序列分类问题应用于传统的分类算法上. 当抽取 k 个逻辑 shapelets 之后，将对初始数据集进行基于逻辑 shapelets 的时间序列转换，数据集中的每一条时间序列被转换成拥有 k 个属性的实例，每个属性的值对应于该时间序列与逻辑 shapelets 之间的距离. 如算法 7 所示，距离的计算采用了 4.1 节所描述的充分统计量技术(第 7、9 行)，当逻辑操作标识 *opt* 大于等于 0 时，表示合取操作，此时需计算逻辑 shapelets 中每一个 shapelet 与当前时间序列间的距离，并取最大者作为转换后实例的属性值. *opt* 小于 0 则取最小值.

算法 7.　逻辑shapelets转换*TransformData*(S, D, *opt*).

输入：最好的 k 个逻辑 shapelets S，数据集 D，逻辑操作 *opt*

输出：转换后的数据集 *output*

1. *output*←∅;
2. FOR i←0 to |D| DO {D 中的每一条时间序列}
3. *transformed*←∅;
4. FOR j←0 to |S| DO {S 中的每一个逻辑 shapelets}
5. FOR m←0 to |S_j| DO {S_j 中的每一个 shapelet}
6. IF (*opt*≥0) {合取操作}
7. *dist*←*Max*(***sufficientdist***(u, l, $Stats_{x,y}$));
8. ELSE IF (*opt* <0) {析取操作}
9. *dist*←*Min*(***sufficientdist***(u, l, $Stats_{x,y}$));
10. *transformed.add*(*dist*);
11. *output.add*(*transformed*);
12. RETURN *output*;

初始的时间序列数据集被转换之后，即可应用于传统的分类算法如贝叶斯分类器、决策树、1-NN 等等. 本文将在第 5 节验证所提出算法的分类准确率.

4.5 分类与准确率计算

分类器在给定测试集上的准确率是分类器正确分类的测试集元组所占的百分比. 发现 k 个最好的逻辑 shapelets 之后,通过将初始数据集转换成新的数据集,就可以将时间序列分类问题转换成传统的分类问题,并使用传统分类器在测试集上的分类准确率来评价逻辑 shapelets 转换的表现. 如算法 8 所示,首先将训练集和测试集分别转换成新的训练集和测试集(第 1～2 行),然后根据转换后的训练集和相应的分类算法建造分类器,此处的分类器 C 可以是贝叶斯分类器、决策树、1-NN 分类器等等(第 3 行). 最后,针对转换后测试集中的每一条实例,若分类器预测的类标与真实类标相同,则计数值加 1(第 5～9 行),最终返回该分类器的分类准确率(第 10～11 行).

算法 8. 分类准确率 $Accuracy(C,D,T)$.

输入:分类器 C,训练数据集 D,测试数据集 T

输出:分类准确率 $accuracy$

1. $D \leftarrow$ ***TransformData***(S,D,opt);
2. $T \leftarrow$ ***TransformData***(S,T,opt);
3. $C' \leftarrow BuildClassifier(C,D)$;
4. $accuracy \leftarrow 0$;
5. FOR $i \leftarrow 0$ to $|T|$ DO {T 中的每一条时间序列 T_i}
6. $class\_label \leftarrow getClasslabel(T_i)$;
7. $predict\_label \leftarrow ClassifyInstance(C',T_i)$;
8. IF $predict\_label.equal(class\_label)$
9. $accuracy \leftarrow accuracy+1$;
10. $accuracy \leftarrow accuracy/|T|$;
11. RETURN $accuracy$;

5 实验评估

在此部分,将评估所应用的加速技术以及逻辑 shapelets 转换技术. 首先,将应用充分统计量技术后的 ShapeletFilter 算法[1] 命名为 ShapeletAcc 算法,由于相关的时间复杂度分析已在 4.1 节中给出,本文将从时间上对比 ShapeletAcc 算法与 Lines 等人提出的 ShapeletFilter 算法[1];然后通过实验验证使用逻辑 shapelets 转换进行分类的准确性. 所采用的数据集由 Ding 等人[3] 以及 Keogh 等人① 所提供. 所有的算法和实验都是在 Weka 框架下实现的.

5.1 参数设置

本文所提出的算法需要从时间序列数据集中抽取出 k 个逻辑 shapelets,由于 k 值仅仅表示抽取的逻辑 shapelets 的数量,它并不影响得到的逻辑 shapelets 的质量,但我们无法保证设置的 k 值能够转换出最适合分类的数据. 当 k 值设置过小时,所得到的逻辑 shapelets 可能不足以提供分类决策所需要的信息;而 k 值设置过大时,可能造成过拟合现象或降低重要逻辑 shapelets 的辨别性. 在实验中,为保持 k 值设置的一致性和简洁性,我们将 k 值设置为 $m/2$,这里 m 为时间序列中属性的个数.

另外,算法中的两个长度参数 min 和 max 的设置也是一个难题. 由于它们定义了候选 shapelets/逻辑 shapelets 的长度范围,参数设置不正确时可能发现不了最具有辨别性的 shapelets/逻辑 shapelets,从而对分类器的准确率造成影响. 同样为保持最大长度和最小长度设置的一致性和简洁性,统一将最小长度设置为 $m/10$,最大长度设置为 $m/2$.

5.2 速度对比

此处将对比基于 shapelets 转换的 ShapeletFilter 算法和本文使用充分统计量之后的改进算法 ShapeletAcc. 尽管采用充分统计量之后,发现 shapelets 的速率得到了很大的提升,但由于在寻找逻辑 shapelets 时需要重复地遍历数据集,一般情况下,发现逻辑 shapelets 需要消耗较多的时间.

虽然改进了原有算法的效率,但发现 shapelets 的过程仍是一项耗时的工作,所以尽量采用实例或属性较少的数据集进行测试,并舍弃了那些无法在 24 h 之内得到结果的数据集. 所采用的数据集如表 2 所示.

表 2 发现 k 个 shapelets 的时间(s)对比

数据集	ShapeletAcc	ShapeletFilter	加速
SonyAIBORobotSurface	2.37	9.34	3.93
SonyAIBORobotSurfaceII	3.65	14.99	4.11
synthetic_control	2420.81	3217.67	1.33
TwoLeadECG	5.56	26.75	4.81
Beef	1147.72	42346.07	36.90
CBF	31.95	248.39	7.77
Coffee	217.95	5359.17	24.59
Cricket	18.00	177.83	9.88
DiatomSizeReduction	150.06	3597.34	23.97
ECG200	227.34	1071.20	4.71
ECGFiveDay	24.35	218.68	8.98
FaceFour	327.92	9402.13	28.67
Gun_Point	150.55	1467.21	9.75
ItalyPowerDemand	4.40	5.21	1.18
Motes	4.29	21.93	5.11
OliveOil	1590.67	75996.02	47.78
Symbols	436.14	15042.84	34.49

① Keogh E, Xi X, Wei L, Ratanamahatana C A. The UCR Time Series Classification/Clustering Homepage. http://www.cs.ucr.edu/～eamonn/time_series_data/

表格的第 1 列为数据的名称，发现 k 个 shapelets 的过程是在各个数据的训练集上完成的，时间的单位为 s，所得数值均为 10 次测量求平均值的结果. 从表中可以看到，ShapeletAcc 算法的运行时间明显少于 ShapeletFilter 算法，最大的加速比可达 47.78 倍，更直观的图形表示如图 6 所示.

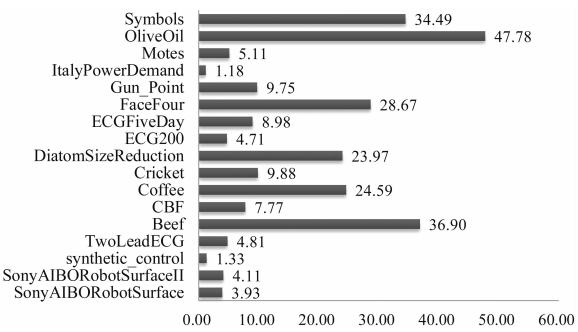

图 6　ShapeletAcc 与 ShapeletFilter 的加速比

5.3　逻辑 shapelets 转换与 shapelets 转换的对比

此部分将对比基于逻辑 shapelets 转换的时间序列分类算法和基于经典 shapelets 转换的时间序列分类算法. 在进行逻辑 shapelets 转换时，我们考虑只是合取或者只是析取这两种情况，分别将两种算法命名 LG-AND 和 LG-OR. 需要注意的是，初始时，将 k 值设定为 $m/2$，m 为时间序列中属性的个数. 为保持对比的公平性，当逻辑 shapelets 的个数小于 $m/2$ 时，将补入经典的不相似的 shapelet 以满足相应的个数. 若发现的全部 shapelets 的个数 n 小于 $m/2$，则将 k 值设置为 n.

为避免偏差，在进行实验时，所有的逻辑 shapelets 和经典 shapelets 都是从训练集中抽取出来的，然后再将训练集和测试集分别转换成新的适用于不同分类器的数据. 所有的分类器都是根据训练集建立然后在测试集上进行测试的. 由于实验是在 Weka 框架下进行的，所采用的分类器均使用了 Weka 的默认设置. 具体的分类准确率对比如表 3 所示.

表 3　逻辑 shapelets 转换与 shapelets 转换的对比

算法	NaïveBayes/%	C4.5/%	RandomForest/%	1-NN/%	RotationForest/%	BayesNet/%	数据集
LG-AND	91.35	**84.86**	**88.52**	91.02	89.85	91.68	
LG-OR	**93.34**	**84.86**	86.19	**93.51**	**93.01**	93.51	SonyAIBORobotSurface
Shapelets	92.51	**84.86**	85.69	92.35	91.35	**93.68**	
LG-AND	**88.35**	**75.97**	82.90	**89.30**	**89.09**	**87.51**	
LG-OR	87.09	**75.97**	**87.62**	81.01	87.72	84.78	SonyAIBORobotSurfaceII
Shapelets	86.88	**75.97**	84.47	83.32	87.20	84.68	
LG-AND	99.12	**91.57**	**98.95**	99.39	**99.56**	99.82	
LG-OR	**99.39**	**91.57**	97.89	**99.91**	99.21	**99.91**	TwoLeadECG
Shapelets	99.21	**91.57**	96.75	99.82	**99.56**	**99.91**	
LG-AND	**89.06**	79.79	79.87	**88.66**	**86.34**	**84.58**	
LG-OR	85.62	**80.27**	**80.51**	85.22	84.11	80.67	Motes
Shapelets	86.58	**80.27**	80.27	87.22	82.03	80.51	
LG-AND	**92.71**	**92.91**	**92.32**	93.20	92.23	92.13	
LG-OR	92.52	**92.91**	90.77	**96.02**	**94.95**	**92.23**	ItalyPowerDemand
Shapelets	92.42	**92.91**	90.77	93.49	92.91	91.74	
LG-AND	**99.54**	**98.95**	**99.42**	**100.00**	99.54	99.42	
LG-OR	**99.54**	**98.95**	**99.42**	99.54	**100.00**	**99.54**	ECGFiveDays
Shapelets	99.42	**98.95**	**99.42**	99.19	99.30	99.42	

从表 3 中可以观察到，多数情况下，基于逻辑 shapelets 转换的算法 LG-AND 和 LG-OR 在各种类型分类器上的表现优于 shapelets 转换算法[1]. 平局的情况大多出现在以信息增益为度量的决策树或决策树组合模型中，如三者在 C4.5 算法的准确率对比上出现了 5 次平局. 究其原因，是因为在选取 shapelets 或者逻辑 shapelets 时采用信息增益作为度量标准，而 C4.5 算法在选择信息增益最大的属性时，必然优先选择与信息增益最大的 shapelet 或者逻辑 shapelets 相对应的属性，若两者在决策树的某条路径上的信息增益相同（此时逻辑 shapelets 的分裂间隔较大）或相近，则会生成类似的决策规则. 在 LG-AND 和 LG-OR 的对比中，LG-AND 在 14 个分类器上优于 LG-OR，LG-OR 在 15 个分类器上占优，两者持平 7 次，所以从整体上说，LG-AND 与 LG-OR 的表现差别不大.

5.4　逻辑 shapelets 转换与其他分类器的对比

1-NN 分类器是当前解决时间序列分类问题的最好分类器之一，为进一步验证本文所提出的基于逻辑 shapelets 转换的时间序列分类算法的性能，在此

部分,将对比基于欧氏距离的 1-NN 分类器和基于逻辑 shapelets 转换的 1-NN 分类器.另外,本文也对比了基于逻辑 shapelets 转换后的 C4.5 分类器和直接采用 shapelets 实现的决策树[4]的分类准确性.从表 4 中可以观察到,基于逻辑 shapelets 转换的1-NN 分类器明显优于基于欧氏距离的 1-NN 分类器,基于逻辑 shapelets 转换的 C4.5 分类器也在大部分情况下优于直接采用 shapelets 实现的决策树分类器.

表 4 准确率对比

数据集	1-NN/%	LG-1NN/%	ShapeletTree/%	LG-C4.5/%
SonyAIBORobotSurface	67.72	91.02	84.53	84.86
SonyAIBORobotSurfaceII	86.36	89.30	75.97	75.97
TwoLeadECG	72.52	99.39	85.07	91.57
Motes	85.78	88.66	82.51	79.79
ItalyPowerDemand	95.72	93.20	89.21	92.91
ECGFiveDays	80.60	100.00	77.47	98.95
Cricket	87.76	91.84	60.20	82.65
Cricket_new	43.75	57.81	50.00	56.25

5.5 实验总结

首先,通过相应的时间对比实验,验证了充分统计量技术能够大幅度提高 shapelets 的发现效率;其次,通过与基于 shapelets 转换的时间序列分类算法和其他经典时间序列分类算法的对比实验,验证了所提出的基于逻辑 shapelets 转换的时间序列分类算法的分类准确性.同时,在实验过程中也发现了一些 shapelets 转换技术的缺点,比如分类器训练时间较长,k 值以及 shapelets 最大最小长度的设定比较困难等.

6 结论与展望

在本文中,介绍了一种用于快速发现 shapelets 的充分统计量技术,并针对 shapelets 的不足,提出了一种基于逻辑 shapelets 转换的时间序列分类算法,通过实验验证了所提出算法的分类准确性.基于逻辑 shapelets 转换的缺点之一是训练时间相对较长,一种潜在的缓解方式是通过对训练集采样,即使用原始数据的一小部分子集来发现逻辑 shapelets.另外,还可以进一步考虑聚类相似的逻辑 shapelets,然后从每一簇中得到一个代表性的逻辑 shapelets 来代替寻找 k 个逻辑 shapelets 进行数据转换等.

致 谢 在此需要感谢 Jason Lines 提供关于 shapelets 转换的源代码和针对相关问题的解答.同时也感谢匿名审稿人对本文提出宝贵意见!

参 考 文 献

[1] Lines J, Davis L M, Hills J, Bagnall A. A shapelet transform for time series classification//Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2012). Beijing, China, 2012: 289-297

[2] Bagnall A, Davis L, Hills J, Lines J. Transformation based ensembles for time series classification//Proceedings of the 2012 SIAM International Conference on Data Mining (SDM 2012). Anaheim, USA, 2012: 307-318

[3] Ding H, Trajcevski G, Scheuermann P, et al. Querying and mining of time series data: Experimental comparison of representations and distance measures//Proceedings of the 34th International Conference on Very Large Data Bases (VLDB 2008). Auckland, New Zealand, 2008: 1542-1552

[4] Keogh E, Kasetty S. On the need for time series data mining benchmarks: A survey and empirical demonstration. Data Mining and Knowledge Discovery, 2003, 7(4): 349-371

[5] Ye L, Keogh E. Time series shapelets: A new primitive for data mining//Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2009). Paris, France, 2009: 947-956

[6] Ye L, Keogh E. Time series shapelets: A novel technique that allows accurate, interpretable and fast classification. Data Mining and Knowledge Discovery, 2011, 22(1-2): 149-182

[7] Mueen A, Keogh E, Young N. Logical-shapelets: An expressive primitive for time series classification//Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2011). San Diego, USA, 2011: 1154-1162

[8] Rakthanmanon T, Keogh E. Fast shapelets: A scalable algorithm for discovering time series shapelets//Proceedings of the 13th SIAM International Conference on Data Mining (SDM 2013). Austin, USA, 2013: 668-676

[9] Zakaria J, Mueen A, Keogh E. Clustering time series using unsupervised-shapelets//Proceedings of the 12th IEEE International Conference on Data Mining (ICDM 2012). Brussels, Belgium, 2012: 785-794

[10] Xing Z, Pei J, Yu P, Wang K. Extracting interpretable features for early classification on time series//Proceedings of the 11th SIAM International Conference on Data Mining (SDM 2011). Mesa, USA, 2011: 247-258

[11] Hartmann B, Link N. Gesture recognition with inertial sensors and optimized DTW prototypes//Proceedings of the IEEE International Conference on Systems Man and Cybernetics (SMC 2010). Istanbul, Turkey, 2010: 2102-2109

[12] Rakthanmanon T, Campana B, Mueen A, et al. Searching and mining trillions of time series subsequences under dynamic time warping//Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2012). Beijing, China, 2012: 262-270

[13] Sakurai Y, Papadimitriou S, Faloutsos C. Braid: Stream mining through group lag correlations//Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2005). Baltimore, USA, 2005: 599-610

YUAN Ji-Dong, born in 1989, Ph.D. candidate. His research interests include data mining and pattern recognition.

WANG Zhi-Hai, born in 1963, Ph. D., professor, Ph. D. supervisor. His research interests include data mining and machine learning.

HAN Meng, born in 1982, Ph. D. candidate, lecturer. Her research interests include data mining and machine learning.

YOU Yang, born in 1991, M. S. candidate. His research interests include next generation internet and data processing.

Background

Time series data mining is an important task in machine learning and data mining. As one of the subtypes of sequence classification, time series classification has a broad range of applications such as agriculture, chemistry, health informatics, finance, industry and so on. The goal of time series classification is first to learn what are the distinctive features distinguishing classes from each other. Then, when an unlabeled dataset entered into the system, it can automatically determine which class each series belongs.

There are three major challenges in time series classification. First, there are no explicit features in time series data, so classifiers that take input data as a vector of features, such as decision trees and neural network, cannot handle this data efficiently. Second, the dimensionality of the feature space for the time series data is very high. Though we can use various feature selection methods to transform a time series into a set of features, the computation of feature selection can be costly. Third, building an interpretable time series classifier is difficult since there are no explicit features.

Shapelets-based method is one of the most promising ways to solve the problem of time series classification. This paper is focused on building an accurate, fast and interpretable classifier. In our work, we address the time consuming work of finding shapelets by using an intelligent caching based and reuse of computations technique, which decreases the process of finding shapelets by an order of magnitude of current method. On this basis, a novel transformation that is based on conjunctive or disjunctive of logic shapelets is proposed, which can improve the classification accuracy whilst retaining the explanatory power provided by shapelets.

This work is supported by the Fundamental Research Funds for the Central Universities (No. 2015YJS049). From the viewpoint of real application, this project focuses on more efficient and accurate time series representation and classification techniques. Until now, the research team has published more than 30 papers about time series data mining, Bayesian networks learning strategies and sequential pattern mining.