博 士 学 位 论 文

时间序列分类算法研究

Research on Time Series Classification

作者：原继东

导师：王志海

北京交通大学

2016 年 9 月

学位论文版权使用授权书

本学位论文作者完全了解北京交通大学有关保留、使用学位论文的规定。特授权北京交通大学可以将学位论文的全部或部分内容编入有关数据库进行检索，提供阅览服务，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。学校可以为存在馆际合作关系的兄弟高校用户提供文献传递服务和交换服务。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：　　导师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

学校代码：10004 密级：公开

北京交通大学

博士学位论文

时间序列分类算法研究

Research on Time Series Classification

作者姓名：原继东 学 号：12112078

导师姓名：王志海 职 称：教 授

学位类别：工学 学位级别：博士

学科专业：计算机科学与技术 研究方向：数据挖掘

北京交通大学

2016 年 9 月

致谢

本文是我博士期间的工作总结，在此谨向所有支持和帮助过我的老师、同学、朋友和家人致以诚挚的感谢！

感谢我的导师王志海教授！读博士期间，王老师在学习，生活和工作上都给了我很大的帮助。王老师为我创建了良好的学习环境和优良的学术氛围，并提供了赴国外交流的机会，帮我建立了与同行间的学术交流，拓宽了我的视野，为我的学术研究奠定了扎实的基础。能得到王老师的辛勤指导是我人生的宝贵财富，也必将激励我继续在科研的道路上探索。

感谢我在法国的导师 Ahlame Douzal 教授，在法国访问期间，Ahlame 严谨细致的治学精神给我留下了深刻的影响，同时也感谢她在学术上对我的无私帮助和指导。

感谢实验室的瞿有利老师以及付彬师兄，你们在学习和生活上给了我很多有帮助的建议！

感谢和我一起成长的同学和朋友们：张启伟，白凤伟，苗润华，刘海洋，韩萌，孙艳歌，杜超，鞠卓亚，马宾，熊昊，Saeid，Bikash 等等，是你们让我的博士生活更加充实多彩！

我还要特别感谢我的每一个家人。回首二十年一来的寒窗苦读，满是你们爱护与鼓励的身影，你们的支持，理解与帮助永远是我前进的动力！也愿我们的明天会更好！

最后，特别感谢我的妻子，感谢你一直以来对我的陪伴，支持与鼓励！感谢你在我一无所有时仍毅然选择和我走在一起！感谢你见证我的学生生涯！感谢你愿与我携手，共度此生！

摘要

时间序列数据广泛存在于日常生活中的几乎每一个应用领域。它们是实值型的序列数据，具有数据量大，数据维度高，以及数据不断更新等特点。时间序列分类问题与传统分类问题之间的主要区别在于，时间序列数据的各个变量之间具有次序关系，而传统分类问题认为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。因此，时间序列分类问题已成为数据挖掘领域的特殊挑战之一。

时间序列分类问题主要面临着三个方面的挑战。首先，对于传统分类器而言，输入数据为特征向量，然而时间序列数据并没有明确的特征；其次，尽管可以在时间序列数据上进行特征选择，但由于其特征空间维度非常大，特征选择的过程会花费很大的计算量；最后，在某些应用中，除了精确的分类结果之外，我们还希望得到具有可解释性的分类器。但由于其没有明确的特征，建立一个可解释性的分类器是非常困难的。本文围绕以上三个问题深入研究如何建立具有可解释性的时间序列分类器，主要贡献如下：

（1）研究得出一种基于逻辑 shapelets 转换的时间序列分类算法。时间序列 shapelets 是时间序列中最具有辨别性的子序列。首先，针对时间序列规范化过程缓慢的问题，通过应用一种基于智能存储和计算重用的技术，将发现 shapelets 的时间复杂度降低一个数量级；其次，为提升 shapelet 的辨别性，提出一种基于合取或析取的逻辑 shapelets 的转换方法。通过逻辑 shapelets 转换，将初始时间序列转换成新的非序列数据，同时也把时间序列的分类问题转化成了经典的分类问题。此方法在保持 shapelets 辨别性的同时提升了分类的准确性。

（2）研究得出一种简单有效的 shapelet 剪枝和覆盖方法。首先，针对 shapelets 转换时相似 shapelets 过多的问题，提出一种基于 shapelet 分裂阈值的剪枝方法，用于过滤掉相似的 shapelets，并大幅度减少候选 shapelets 样本的数量；其次，提出一种基于 shapelets 覆盖的方法来确定数据转换时 shapelets 的数量，并保证 shapelets 对实例的覆盖；最后，阐述如何将所提出算法扩展到逻辑 shapelets 转换中，并将所提出的算法和其他基于 shapelets 的时间序列分类算法，以及基于不同距离度量的 1-NN 基准分类器作对比，阐明所提出算法的分类准确性和可解释性。

（3）首次将关联式分类器应用于普遍的时间序列分类问题中，阐述了基于 SAX（Symbolic Aggregate approXimation）表示的关联式分类器在时间序列数据上的可解释性。首先，针对传统关联规则主要应用于符号型事务数据而无法应用于数值型时间序列的问题，采用 SAX 表示方法离散化并符号化时间序列；其次提出一

种改进的 CBA（Classification Based on Associations）算法，用于发现类序列规则并分类预测。在此基础上，提出一种懒惰式的关联式分类算法，避免产生过量规则，并保证规则对测试实例的覆盖。另外也评估了四种不同的类序列规则评价方式。

（4）研究得出一种具有可解释性的基于动态时间弯曲（Dynamic Time Warping，DTW）的 k 近邻（k nearest neighbours，k-NN）分类器。k-NN 分类器被认为是当前解决时间序列分类问题的基准分类器。针对其可解释性的不足，首先，提出了一种新的有效的时间序列加权模型，为每一条时间序列的每一个特征提供权值；其次，提出了两种不同的 DTW 加权方式来发现辨别性子序列，通过和其他基于非相似性度量的 k-NN 分类器相比较，展示了其可解释性；最后，将所提出模型扩展至多变量时间序列分类问题，并讨论其特殊情况，即加权欧式距离在时间序列分类问题上的应用；通过在多个公共数据集上与多个方法的对比，展现所提出模型在时间序列分类问题上的有效性与可解释性。

上述结论从多种角度论述了时间序列分类器的构造和分类过程，展示了各个分类器在寻找辨别性子序列方面的高效性，提升了时间序列分类方法的可解释性，也为实际应用问题奠定了良好的基础。

图 62 幅，表 26 个，参考文献 147 篇。

关键词：时间序列；时间序列分类；Shapelets 转换；关联式分类；可解释性；k 近邻

ABSTRACT

Time series data exists widely in almost every field of our daily life. It is real-valued sequence data, which is also high dimensional, large volumed, and updated continually. The main difference between time series classification problem with traditional classification issue is that, the former one's variables are ordered in timestamp, while for the latter one, the order of each variable is unimportant, and the correlation between each other is independent of their relative positions. Therefore, time series classification problem has become one of the greatest challenges of data mining.

There are three major challenges for time series classification. First, for traditional classifiers, input data is considered as a feature vector, while there are no explicit features in time series data. Second, although feature selection methods could be applied on time series, it is time consuming since the high dimensionality of time series. Third, besides accurate classifiers, we may also want to build an interpretable classifier. But it is difficult for time series since there are no explicit features. In order to solve these three problems, this dissertation mainly focuses on building interpretable classifier for time series, the main contributions are as follows.

(1) A logical shapelets based transformation method is studied. Time series shapelets is considered as the most discriminative subsequence of time series. First, the process of discovery shapelet is time consuming, even though shapelets are computed offline. This 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. Second, in order to improve the interpretability of shapelet transformation, a novel transformation that is based on conjunctive or disjunctive of shapelets is proposed. This method tranformes original time series data into traditional data, which could be treat by classical classifiers. Experimental results have shown the efficiency of logical shapelets transformation on classic benchmark datasets used for these problems, which can improve classification accuracy, whilst retaining their interpretability.

(2) A simple but effective shapelet pruning and coverage method is proposed. First, previous algorithms often inevitably result in similar shapelets among the selected shapelets. This work addresses this problem by introducing an efficient and effective shapelet pruning technique to filter similar shapelets and decrease the number of candidate shapelets at the same time. Second, on this basis, a novel shapelet coverage

method is proposed for selecting the number of shapelets for a given dataset, which ensures the coverage of original dataset. Experiments on the classic benchmark datasets for time series classification, comparing with distance metrics based 1-NN and other shapelets based methods, demonstrate that the proposed transformation is interpretable and improve classification accuracy as well.

(3) To the best of our knowledge, we proposed the first work that discovery association rules on time series datasets. The interpretability of SAX-based associative classifier is represented and experimental results show that classifiers built this way are competitive. First, a SAX (Symbolic Aggregate approXimation) representation that discretizes original time series into symbolic string is adopted since traditional association rules can only handle transcation dataset. Second, a modified eager CBA (Classification Based on Associations) algorithm is proposed to discover Class Sequential Rules and make the final prediction firstly; On this basis, a lazy associative classification is proposed, which is in contrast to the eager one that generates excessive number of rules, but still unable to cover some test data with the discovered rules. In addition, four different methods that select the mined rules are also proposed for carrying out associative classification.

(4) An interpretable DTW (Dynamic Time Warping) based robout *k*-NN (*k* Nearest Neighbours) classifier is studied. *k*-NN is considered as the bench mark classifier for time series classification, but it is not interpretable. For that, a novel and effective time series weighting model is proposed to provide corresponding weight for each time series alignment firstly; Then, a weighted DTW dissimilarity measure, based on two different function, is proposed to discover discriminative subsequence of time series; Compared with other dissimilarity measure based *k*-NN classifiers, our method shows the ability of interpretable; Last but not least, we propose an extension of our weighting model to multivariate time series classification, discussing its special case, weighted Euclidean distance at the same time; We also evaluate the proposed method on sets of univariate/multivariate time series, demonstrating the utility of our discriminative local weighting model.

In conclusion, the achievements of this dissertation have demonstrated the process of building time sereis classifiers and classifying instances on several facets. Experimental results showed their efficiency of discovery time series shapelets or discriminative subsequences, improving the interpretability at the same time. Moreover, this dissertation has laid a sound foundation for real applications.

KEYWORDS： Time series; Time series classification; Shapelets transformation; Associative classification; Interpretable; *k*-nearest neighbours

目录

图目录

表目录

1 绪论

1.1 研究背景与意义

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。现实生活中，通过一系列时间点上的观测来获取数据是司空见惯的活动[1]。比如，在商业上，我们会观测交易日股票开闭盘价、银行周利率、商品月价格指数、年销售量等等，图 1.1 展示了某股票 2011 至 2014 年的日股票闭盘价所组成的时间序列。在气象上，我们会观测太阳黑子的活动情况、记录每天的最高或最低温度、年降水量、每小时的风速等等，图 1.2 展示了 1770 年到 1869 年间太阳黑子的活动情况（横坐标为年份）。在生物科学上，我们会观测每毫秒心电或脑电活动的状况，图 1.3 给出了某病人的心电图活动状况。另外，在农业上，我们会记录不同农作物每年的产量、土壤侵蚀情况、出入口农产品销售量等方面的数字。在生态学上，我们会记录动物种群数量的变动情况等等。目前，时间序列数据正以不可预测的速度产生于日常生活中的几乎每一个应用领域。

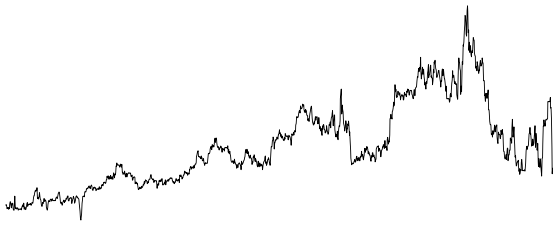

图 1.1 某股票 2011-2014 年的交易数据

Figure 1.1 Stock transaction dataset between 2011 and 2014

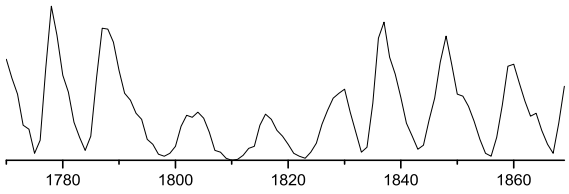

图 1.2 1770 年到 1869 年间太阳黑子的活动情况

Figure 1.2 Activities of macula from 1770 to 1869

如图 1.1-1.3 所示，时间序列数据是实值型的序列数据，具有数据量大，数据维度高，以及数据是不断更新的等特点。时间序列分类问题作为序列分类问题的一

个分支[2]，已经在时间序列挖掘领域引起了广泛地关注。时间序列分类的目标是首先从标定类标的训练集中学习到能够区分不同序列的鉴别性特征；然后，当一条未标定的时间序列到来时，它能够自动决定该时间序列的类标。它与传统分类问题之间的差别在于，对于后者而言，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，变量的次序在寻找最佳的辨别性特征时起着至关重要的作用[3]，因此，时间序列分类问题已成为数据挖掘领域的特殊挑战之一。

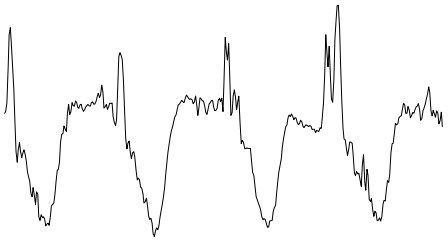

图 1.3 心电图示例

Figure 1.3 An example of ECG dataset

时间序列分类问题亦广泛地存在于现实生活中的诸多领域：如健康信息处理中的心电图（Electrocardiograph，ECG）或脑电图（Electroencephalograph，EEG）分类[4]，气象中天气状况预测[5]，根据传感数据来区分不同的行为动作[6-8]，根据用电量来区分不同的家用电器[9]等等，Keogh 等人还专门收集了用于时间序列分类/聚类的 UCR 数据集[10]。

另外，在时间序列分类问题中，任意实值型有次序的数据被当作一条时间序列[11]。也就是说，数据不需要是在时间上有序的，任何逻辑上有序的实值型数据都可用于时间序列分类问题中。因此，一张图像的轮廓或手写字符可以被转变为一条单变量或多变量时间序列，然后应用时间序列分类方法处理图像分类问题或手写字符识别问题，比如，研究者们已也将时间序列分类方法用于植物叶片的识别，古文物中的箭头识别，人脸分类，音乐片段比对等等[6-8, 12]。鉴于时间序列数据存在的广泛性和时间序列分类方法应用的普遍性，本文决定在现有时间序列分类方法的基础上展开深入的研究工作。

1.2 国内外发展现状

近十年来，针对时间序列数据的挖掘研究工作引起了越来越多研究者的关注，也取得了许多长足的进步。本节将在给出本文通用的时间序列符号的基础上（第 1.2.1 节），主要从时间序列的表示（第 1.2.2 节）和分类（第 1.2.3 节）两个方面阐

述国内外的研究现状。

1.2.1 定义与符号

本节用于介绍文中涉及到的基本术语和相关符号。

表 1.1 基本符号表

Table 1.1 Basic symbols

符号	释义
$\boldsymbol{X}$	时间序列数据集
$\boldsymbol{x}, \boldsymbol{x}_i$	一条时间序列
x_i, a_i^q	时间序列中的数据点
$\boldsymbol{s}$	时间序列的子序列
l	子序列长度
T	时间序列的长度
μ_x, σ_x	均值与方差
N	时间序列的个数
$\boldsymbol{Y}$	类标值集合
y	单个类标
q	多变量时间序列中变量个数

定义 1.1：时间序列数据集

$\boldsymbol{X}$ 为一个时间序列数据集，其中包含有 N 条时间序列，即 $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$。

定义 1.2：单变量时间序列

按照某一时间点上观测变量的多少，时间序列可划分为单变量时间序列（univariate time series）和多变量时间序列（multivariate time series）。在单变量时间序列中，时间序列 $\boldsymbol{x}$ 是一条长度为 T 实值的序列，可表示为 $\boldsymbol{x} = \{x_1, x_2, \ldots, x_T\}$。一般情况下，数据点 $x_1, x_2, \ldots, x_T$ 是按照时间顺序排列的，两两之间具有相同的时间间隔，此时的时间序列为离散型时间序列。同样，连续型时间序列的观测值是在连续的时间点上取得的，本文将专注于处理单变量离散型时间序列。

定义 1.3：多变量时间序列

对于多变量时间序列，每一条时间序列 $\boldsymbol{x} = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_T\}$ 包含 T 个观测值，其中每一个观测值有 q 个变量，即 $\boldsymbol{a}_i \in \mathbf{R}^q$ 并且 $\boldsymbol{a}_i = \{a_i^1, \ldots, a_i^q\}^{\mathrm{T}}$。注意，对于多变量时间序列，每条时间序列的长度 T 可能不同。

定义 1.4：时间序列子序列

给定一条时间序列 $\boldsymbol{x} = \{x_1, x_2, \ldots, x_T\}$，其子序列 $\boldsymbol{s}_{i,l} = x_i, x_{i+1}, \ldots, x_{i+l-1}$ 是一条 $\boldsymbol{x}$ 中从位置 i 开始长度为 l（$l \le T$）的连续的子序列，此处 $1 \le i \le T-l+1$。时间序列的子序列可以当作是时间序列的局部特征。假设一个长度为 T 的时间序列的子序列的最小长度为 1，最大长度为 T，那么它能有 $T(T+1)/2$ 个子序列。

定义 1.5：时间序列的表示

给定一条长度为 T 的时间序列 $\boldsymbol{x} = \{x_1, x_2, \ldots, x_T\}$，时间序列 $\boldsymbol{x}$ 的表示为维

度是 d（$d << T$）的序列 $\boldsymbol{x}_d$，并且 $\boldsymbol{x}_d$ 与 $\boldsymbol{x}$ 非常接近。

定义 1.6：时间序列分类

假设一个时间序列数据集中有 N 条时间序列，$\boldsymbol{X}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_N\}$，每一条时间序列有 T 个观测值和一个类值 y，即 $\boldsymbol{x}=<x_1，x_2，\ldots，x_T，y>$。时间序列分类问题也可以表述为一个将时间序列的观测值映射到类值的函数 $y \leftarrow f(\boldsymbol{x})$或 $p(y|\boldsymbol{x})$。

定义 1.7：时间序列间的非相似性度量

为获得两条时间序列间的有效比对并保证缩放和偏移的不变性，在处理时间序列之前，须使用 z-规范化方法（z-normalization，如公式（1.1）所示，使得时间序列的值规范在[0,1]之间）对每个时间序列进行规范化处理[4]。因为度量单位的不同或信号采集的偏移都有可能对相似性比对造成巨大的影响，所以这是至关重要的一步。另外，为允许不同长度时间序列间的比对，需要通过除以时间序列的长度来规范化距离，并称之为长度规范化（如公式（1.2）所示）。

$$x_{norm}=\frac{x-\bar{X}}{\sigma_x} \tag{1.1}$$

$$dist(\boldsymbol{x},\boldsymbol{x}')=\sqrt{\frac{1}{T}\sum_{i=1}^{T}(x_i-x'_i)^2} \tag{1.2}$$

公式（1）中时间序列的均值与方差分别为

$$\bar{X}=\mu_x=\frac{1}{T}\sum_{i=1}^{T}x_i \tag{1.3}$$

$$\sigma_x=\sqrt{\frac{1}{T}\sum_{i=1}^{T}(x_i-\mu_x)^2}=\sqrt{\frac{1}{T}\sum_{i=1}^{T}x_i^2-\mu_x^2} \tag{1.4}$$

注意，为方便起见，将公式（1.2）中的规范化距离表示为欧氏距离（Euclidean Distance，ED），而其他距离度量方式，如曼哈顿距离，动态时间弯曲（Dynamic Time Wrapping, DTW）等皆可应用于时间序列间的非相似性度量。

1.2.2 时间序列表示方法

怎样表示一条时间序列是时间序列挖掘的基础问题[13]。如前所述，直接在初始时间序列数据上进行挖掘工作是非常耗时的，为有效的存储和加快时间序列的处理过程，需采用一种简洁的方式来表示时间序列数据，此表示方法需要从时间序列的形状出发并在降低时间序列维度的同时保持其重要的特征[14]。一个有效的时间序列表示方法不仅能够允许序列间的相似性或非相似性比对，也可以较好的应用于不同的数据挖掘任务中。时间序列表示方法的基本特征包括：有效的降低数据

维度；强调局部的和全局的形状特征；较低的计算消耗；能够根据约减后的表示较好地重构原数据；对噪音不敏感或者能够隐式的处理噪音等[14]。

每一种时间序列表示方法都从不同的侧面强调了上述的多个基本特征，根据不同的转换方式，不同的时间序列表示方法被分类为非数据适应性的，数据适应性的和基于模型的三种[15]，下面将一一阐述。

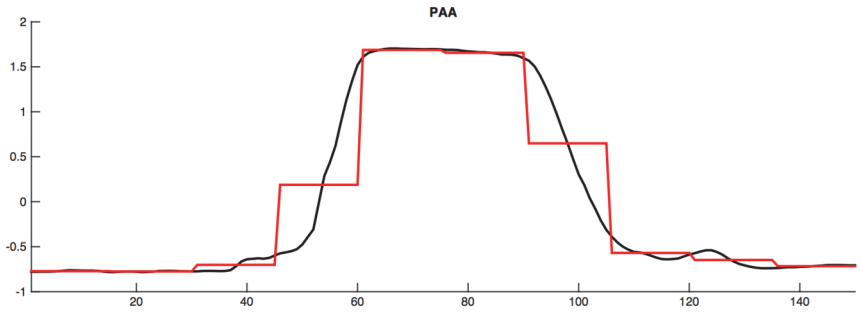

图 1.4 PAA 表示方法示例

Figure 1.4 An example of PAA representation

（1）非数据适应性表示方法

在非数据适应性表示方法中，每一条时间序列的转换参数是一致的。频谱分析是一种比较常见的非数据适应性表示方法。比如，采用离散傅里叶变换（Discrete Fourier Transform, DFT）将时间序列映射到频度域[16, 17]，用基于 Harr 或 Daubechies 的离散小波变换（Discrete Wavelet Transform，DWT）来同时表示时间序列中的时域和频域信息[18-20]等。除频谱分析之外，研究者还提出了其他专门用于时间序列表示的方法。比如，采用基于逐段线性分割（Piecewise Linear Segments）的方法来表示时间序列的形状[21]以及采用 PAA（Piecewise Aggregate Approximation）[22]逐段分割表示时间序列。如图 1.4 所示，PAA 表示方法首先将原始时间序列分割成等长的片段，然后用片段的平均值表示每一个片段。

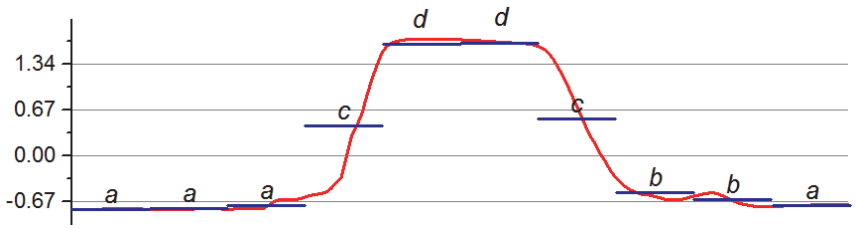

图 1.5 SAX 表示方法示例

Figure 1.5 An example of SAX representation

（2）数据适应性表示方法

数据适应性表示方法在数据转换时，转换参数随着时间序列数据的变化而变

化。非数据适应性表示方法可以转化为数据适应性表示方法，比如 Keogh 等人[23]在 PAA 的基础上提出了一种自适应的维度约减技术 APCA（Adaptive Piecewise Constant Approximation）用于表示时间序列。

众所周知，数值型的高维时间序列数据是难以掌控的，但当把它们表示为符号序列而不是实值序列时，从中挖掘和发现有趣的模式或规则将变得更加容易。Lin 等人[24]提出了一种符号聚集近似（Symbolic Aggregate approXimation，SAX）表示方法，可以将初始的实值型的高维数据转换成离散的低维数据。如图 1.5 所示，一条维度为 150 的时间序列经过 SAX 表示，转换成了维度为 10 的符号序列“*aaacddcbba*”。在低维的离散型的数据上进行搜索是更有效的。更重要的是，研究者可以更加容易地处理 SAX 转换后的数据，比如可以从文本处理或者是生物信息处理中借鉴相应的处理方法（如随机映射等），也可以将哈希表，后缀树等数据结构应用到时间序列数据中。研究者也提出了 SAX 的变种，比如钟等人[25]针对 SAX 符号化时间序列时丢失较多信息的缺陷，引入两个统计特征分量将原来的 SAX 标量符号转化为矢量符号，使其能够比 SAX 提供更多的描述信息。Bondu 等人提出了一种基于数据驱动的能够自适应数据的 SAXO 表示算法[26]。针对 SAX 未能考虑序列趋势信息的缺点，Malinowski 等人[27]提出了一种包含趋势信息的 1d-SAX 表示方法。另外，SAX 表示方法也可以用于时间序列的索引查询[28, 29]。

（3）**基于模型的表示方法**

基于模型的表示方法假设一条时间序列是对某潜在模型的观察结果。常用的模型包括用于模型化时间序列或临床诊断的隐马尔科夫模型[30, 31]（Hidden Markov Model，HMM），用于 EEG 分类的距离耦合隐马尔可夫模型[32]（Distance Coupled HMM，CHMM），用于发现时间序列的内部动态的基于模式的隐马尔科夫模型[33]（*p*HMM），以及用于处理时空序列数据的时空隐马尔科夫模型[34]（STHMM）等等。另外，相应模型也包括 Kalpakis 等人[35]提出的求和自回归移动平均模型（AutoRegression Integrated Moving Average, ARIMA）和 Nanopoulos 等人[36]提出的基于统计的模型（如均值，方差）等等。

1.2.3 时间序列分类方法

所有的分类问题都依赖于数据间的相似性或非相似性度量，时间序列分类问题也不例外。对于时间序列数据来说，时间序列间的相似性有以下三种形式[3]：

（1）时域相似性（Similarity in Time）：同一类别的时间序列都是在时间维度上对某一潜在相同曲线的观察结果，它们之间的不同可能是由噪音和相位漂移所引起的。最近邻（1-Nearest Neighbor，1-NN）分类器最适合处理此类问题，而 DTW

度量可缓解噪音等带来的影响。

（2）形状相似性（Similarity in Shape）：同一类别的时间序列是通过一些相同的子序列或形状来区分的，而且这些子序列可能出现在时间序列的任意位置，这是它与时域相似性的主要不同。子序列与时间的相关性越小，基于时域的 1-NN 分类器就越难处理此类问题，此时可通过使用基于时间序列特征的方法来区分不同的类别。

（3）变化相似性（Similarity in Change）：最不容易被观察到的相似性，此类相似性出现在自相关性较强的序列中。此问题可以用产生式模型，如 HMM，自回归移动平均模型（Auto Regressive Moving Average，ARMA）等等来处理。

接下来，本文将详细阐述这三种相似性以及相对应的分类算法。

（1）基于时域相似性的分类算法

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 1-NN 算法上，如基于 ED 或者 DTW 的 1-NN 等[37, 38]。Faloutsos 等人[39]首次将欧氏距离用于时间序列子序列的匹配算法中。Batista 等人[40]提出了一种复杂性不变的距离度量方式 CID（Complexity-Invariant Distance），用于缓解 1-NN 在处理复杂和简单时间序列时遇到的困境，并指出，“尽管目前有大量的分类算法应用于时间序列分类问题中，但实验表明，简单的 1-NN 分类器是很难被击败的”。另外，Buza 等人[41]还介绍了一种融合不同距离度量方式的 1-NN 分类器。

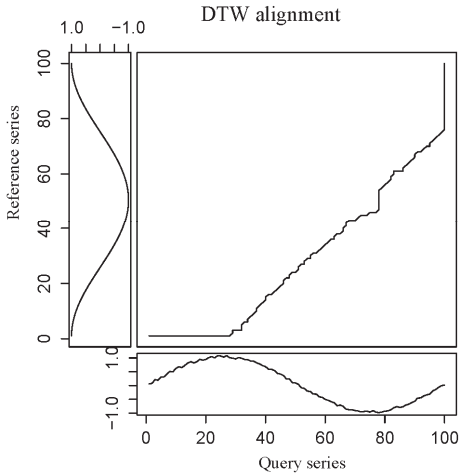

图 1.6 DTW 对齐方式示例

Figure1.6 An example of DTW alignment

欧氏距离对噪音数据和相位漂移比较敏感，而 DTW 能够较好地处理时间轴上的变形。DTW 采用了动态规划的思想处理时间序列，从而更容易发现两条时间序列间的相似性片段。如图 1.6 所示，参照序列中只有波谷，而查询序列中既有波峰

又有波谷，若采用 ED 距离，两者只能进行一一对应（即可以认为 ED 的对齐为图 1.6 方框中的对角线）。而 DTW 方法通过时间弯曲，将两条序列的波谷相对应起来。另外，DTW 还可以比对长度不同的两条序列。

Ding 和 Wang 等人[37, 42]对已有的时间序列维度约减和相似性度量方法做了统一的对比试验，并指出基于 DTW 的距离度量方法可能是当前最好的度量时间序列非相似性的方法。Berndt 等人[43]首次将之前应用于语音识别领域的 DTW 度量方法应用于时间序列的模式发现中。随后，Keogh 等人[44]将基于 DTW 的精确索引应用于时间序列挖掘中。Fu 等人[45]将 DTW 和 US（Uniform Scaling）结合起来用于时间序列查询，并采用多维索引技术来提升查询效率。由于传统的 DTW 在计算两条时间序列间的距离时，每个观测点被赋予了相同的权重，从而忽略了参考点和测试点之间的相位差，此缺陷可能引起形状相似性对比时的误分类问题。Jeong 等人[46]为解决此问题，提出了一种全局加权的 DTW 来分类时间序列。虽然基于 DTW 的 1-NN 分类器是难以击败的，但该算法需要消耗大量的运算时间，所以它并不能很好地处理需要实时反馈的应用，为缓解此问题，Xi 等人[47]提出将数据块消减（numerosity reduction）技术应用于基于 DTW 的 1-NN 分类器，使之在拥有更快分类速度的同时保持较好的分类准确率。另外，Rakthanmanon 等人[4]提出了一种基于 DTW 的快速的时间序列查询方法，并将其扩展到模式发现、聚类、分类以及时间序列数据流的挖掘中。同时，Li 等人也提出了一种支持 DTW 度量的多变量时间序列索引方法[48]。

图 1.7 解决 *Gun/NoGun* 问题的最佳 shapelet

Figure 1.7 Best Shapelet for *Gun/NoGun* problem

（2）基于形状相似性的分类算法

基于时域相似性的时间序列分类器在分类时间序列时，倾向于从整个时间序列入手。然而，在许多时间序列数据集中，同一类别的不同时间序列间往往存在着较大的差异性，基于时域相似性的分类器并不能很好的处理这种差异性，而基于形状相似性的分类算法能够很好地发现区分不同类别的最佳特征。

时间序列 shapelets 是序列之中最具有辨别性的子序列[49]。最初的基于 shapelets 的分类算法是由 Ye 等人[8, 49]所提出的，其通过递归发现当前最好的 shapelet 建立决策树。如图 1.7 所示，对于经典的 *Gun/NoGun*（手中有枪/没枪）问题，最佳 shapelet 如红色标注部分所示，它能够对 *Gun/NoGun* 数据集进行准确地分类。Mueen 等人

[7]针对单个 shapelets 可解释性的不足，提出了使用逻辑 shapelets 来构建决策树的思想。Rakthanmanon 等人[50]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[24]表示的快速 shapelets 近似发现算法。

采用决策树解决时间序列分类问题的思想可追溯到 Yamada 等人[51]提出的二叉分类树。作者提出了两种分裂策略，第一种策略通过穷举搜索得到信息增益最大的一条时间序列作为分裂节点；第二种策略通过穷举搜索得到信息增益最大的一对时间序列作为分裂节点。两种策略均采用 DTW 距离度量时间序列的相似性。Balakrishnan 等人[52]在获取分裂节点时，采用 k-means 聚类（$k=2$）来发现不同类别的代表性序列，这些代表性序列并不是原有的时间序列，而是位于类簇中心的一条抽象序列。Douzal-Chouakria 等人[53]提出了一种综合考虑趋势相似与数值相似的时间序列相似性度量方法，并采用此度量建立能够处理时间序列的决策树。Deng 等人[54]提出了一种随机森林方法[55] ，用于解决时间序列分类问题，同时提出了一种联合信息增益和距离度量的评价方式来选取分裂节点。

上述的几种方法都是在发现 shapelets 的同时构建分类器。Lines 等人[11]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。Hills 等人[56]评估了除信息增益外的其他三种不同的相似性度量方法：Kruskal-Wallis，F-statistic，Mood's median 对 shapelets 选择的影响。Yuan 等人针对 shapelets 转换时可解释性不足以及相似 shapelets 过多的问题[57, 58]，提出了基于逻辑 shapelets 转换的时间序列分类算法和相应的 shapelets 剪枝策略。

（3）基于变化相似性的分类算法

基于变化相似性的分类算法从本质上说是基于模型的分类算法，比如 Zhong 等人[32]成功将广泛应用于语音识别领域的 HMM 模型应用于时间序列中的 EEG 分类问题。在临床学习方面，为对齐和分类时间序列基因表达式，Lin 等人[31]提出了一种辨别性的 HMM 模型来提升分类准确率。Povinelli 等人[59]提出了一种基于相空间重构的高斯混合模型（Gaussian Mixture Models，GMM），用于处理时间序列分类中的信号分类问题。Deng 等人[60]提出将 ARMA 模型用于时间序列的辨别中。Nanopoulos 等人[36]提出了一种用于时间序列分类的多层传感神经网络方法（Multi-layer Perceptron，MLP）。尽管 HMM，GMM 等模型能突出时间序列间变化的相似性，但在分类准确性方面，它们中的多数被证明不如简单的基于 DTW 的 1-NN 算法[47]。

1.3 存在的问题

由于时间序列数据的特殊性，时间序列分类问题面临着三个主要方面的挑战。首先，对于多数分类器如决策树或神经网络来说，输入数据为特征向量，然而时间序列数据并没有明确的特征；其次，尽管可以在时间序列上使用特征选择的方法，但由于时间序列特征空间的维度非常大，特征选择的过程是非常繁琐的，此举会花费很大的计算量；最后，在某些应用中，除了精确的分类结果之外，还希望得到具有可解释性的分类器。但由于时间序列数据没有明确的特征，建立一个可解释性的分类器是非常困难的[2]。结合 1.2 节关于国内外发展现状的描述，时间序列分类挖掘仍存在以下亟待解决的问题。

（1）时间序列的表示技术多用于时间序列的索引和相似性查询[61]，并未能很好地与时间序列分类技术相结合，因此有必要针对不同表示方法在分类方面的适用性，分析设计出适用于时间序列分类的表示技术，并将其应用于时间序列分类中。

（2）如前文所述，时间序列数据多为离散的实值型数据，其并没有明确的特征，因此在时间序列数据上建立一个可解释性的分类器是非常困难的。而关联规则能够对知识进行简洁的、直观的描述，依据关联规则进行分类明显具有很强的可解释性。由于关联规则主要处理的是符号数据，因此鲜有将可解释性的关联规则用于时间序列挖掘的方法。

（3）由于 shapelet 的可解释性和高辨别性，使得基于 shapelets 的分类方法成为近年来的研究热点。但其仍存在一些需要提高的地方，包括 shapelets 的发现效率较低，采用 shapelets 转换时，shapelets 之间的存在着相似性；单个 shapelets 有时无法区分不同类别的时间序列，即单个 shapelets 的可解释性不足；现有的 shapelets 的发现方法未能与不同种类的数据特征相结合，即未能根据不同的应用问题设计出不同评价策略的 shapelets 发现算法；还有，各个时间序列 shapelets 之间的依赖关系仍需要进一步的探讨等等。

（4）虽然基于度量学习的 *k*-NN 分类器是目前处理时间序列分类问题的基准分类器，但 *k*-NN 是一种基于实例的懒惰式分类器，其并不像贝叶斯网络或者决策树那样具有明显的可解释性，因此在研究出具有可解释性的 k-NN 分类器的同时提升其分类准确率就显得至关重要。

（5）研究者对时间序列的研究多数集中在单变量的时间序列表示和分类算法上，尽管目前已经有一些针对多变量复杂时间序列的研究工作[5, 48, 53, 62-65]，但其多致力于多变量时间序列的索引与预测上，因此针对多变量时间序列的分类方法研究仍具有很大的发展空间。

1.4 主要研究内容

时间序列的分类方法在过去的十多年中得到了长足的发展，但现有的方法仍存在着各种各样的不足之处，这也为我们的研究提供了一定的方向。本节针对 1.3 节提出的问题，主要展开以下几方面的研究。

（1）将时间序列表示技术与分类方法相结合，研究时间序列中关联规则的发现方法，并将关联式分类应用于时间序列中。传统的关联规则挖掘算法主要致力于基于符号表示的事务数据上，将关联规则挖掘技术应用于时间序列分类中主要存在两方面的挑战。一是如何符号化或者离散化数值型时间序列。二是如何避免找出过量的无用的规则。对于第一个问题，本文在研究时间序列序列离散化表示方法的基础上，决定 SAX 来符号化表示原有时间序列；对于后一个问题，本文将提出一种懒惰式的序列关联式分类方法，其在保持原有序列间时间顺序的情况下能够进行可解释性的，准确的分类，并能够精确地针对某一条序列产生相应的序列规则。

（2）改进基于 shapelets 的时间序列分类算法。基于 shapelets 的分类方法主要有两类，第一类通过递归地发现当前数据集中最好的 shapelets 来构建决策树；第二类采用基于 Top-*k* shapelets 的空间转换思想，将时间序列分类问题转换成传统的分类问题。本文主要研究第二类方法。首先针对 shapelets 转换时，单个 shapelets 可解释性不足的问题，提出采用逻辑 shapelets 进行数据转换的思想，并采用智能缓冲技术来加快 shapelets 的选取过程；其次针对 shapelets 转换时，top-*k* shapelets 间具有较大相似性的问题，提出了一种 shapelets 剪枝策略，用于去除相似 shapelets，从而允许其他具有辨别性的子序列参与到算法中。另外还提出了用于决定 shapelets 数量的 shapelets 覆盖方法。

（3）设计具有可解释性的 *k*-NN 分类器。*k*-NN 分类器是否具有可解释性一直是具有可争议性的话题。而在时间序列分类中，*k*-NN（尤其是 1-NN）只能说明被分类对象与其最近邻之间的相似性，不能寻找出两者之间的共同点以及被分类对象与其他序列之间的不同之处。针对此问题，本文主要研究了基于 ED/DTW 度量的 *k*-NN 分类器，根据所提出的局部加权方式，通过实例分析，阐释了基于加权 ED/DTW 的 *k*-NN 分类器的可解释性和其在分类方面的准确性。

（4）多变量时间序列的分类算法研究。多变量时间序列分类方法的难点在于，多变量时间序列数据集中序列的长度可能不一样，因此经典的非相似性或相似性度量方法无法处理此类问题，而 DTW 能够处理长短不一的时间序列。为此，本文研究并提出了一种基于局部加权 DTW 度量方式的 *k*-NN 算法，并通过实验展示了其在分类多变量时间序列时的有效性。

1.5 论文组织

全文共分 6 章，第 1 章为绪论，最后一章进行全文总结与展望，第 2 到 5 章为本文的主要部分，具体安排如下：

第 1 章：介绍可解释性时间序列分类算法的研究意义和研究现状，在介绍当前时间序列分类和表示算法的基础上，分析目前时间序列分类方法的优缺点，提出本文的主要研究内容，并简要介绍各章节的安排。

第 2 章：介绍 shapelets 和逻辑 shapelets 的相关概念，以及基于 shapelets 进行数据转换的思想，为增强 shapelets 的解释性并提高分类准确率，提出基于逻辑 shapelets 转换的时间序列分类算法。

第 3 章：研究数据转换时，减少相似 shapelets 的方法，并提出 shapelets 覆盖的思想，用于确定最终 shapelets 转换时辨别性子序列的数量；将所提出方法扩展至基于逻辑 shapelets 转换的分类方法，并通过实验验证所提出方法的优越性。

第 4 章：研究如何将关联规则应用到时间序列分类问题中。首先通过采用 SAX 表示方法将数值型时间序列离散化，然后提出一种懒惰式的时序关联式分类算法，用于过滤掉冗余的序列规则，并通过实验验证所提出算法的可解释性和分类准确性。

第 5 章：基于距离度量的 *k*-NN 分类器被认为当前最好的用于时间序列的分类方法，但经典的 *k*-NN 分类器缺乏可解释性。本章研究一种基于局部加权 DTW 的 *k*-NN 分类器，用于提高分类准确率和分类器的可解释性，实验表明该算法能准确定位时间序列中具有辨别性的子序列。该章还阐释了其在多变量时间序列分类上的应用。

第 6 章：全面综合对比本文所提出的四种算法，总结它们的优缺点和相应的使用范围。对全文进行总结，并对今后的研究内容做进一步的展望。

2 基于逻辑 Shapelets 转换的时间序列分类算法

时间序列 shapelet 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。Shapelets 转换的本质是将时间序列从原始的输入空间映射到新的特征空间（类似于基于核方法的支持向量机[67]），从而忽略原有时间序列不同变量间的时间顺序，然后在新的特征空间上建立不同的分类模型。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，发现 shapelets 的过程相当耗时。本章将针对这些问题，描述一种加速方法和基于逻辑 shapelets 转换的时间序列分类算法。

2.1 引言

时间序列分类问题将任意实值型，有次序的数据看作一条时间序列[11]。它与传统分类问题之间的差别在于，对于后者而言，属性的次序是不重要的，并且变量之间的相互关系独立于它们的相对位置；而对于前者而言，变量的次序有着至关重要的作用（其重要性尤其体现在寻找辨别性特征时[3]）。因此，时间序列分类问题成为了数据挖掘领域的特殊挑战之一。

1-NN 由于具有分类准确率较高且易于实现等优点，传统时间序列分类算法的研究主要集中在基于不同距离度量方式（如 ED，DTW 等）的 1-NN 算法上[37, 38, 68-71]。近年来，基于时间序列 shapelets 的时间序列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 是指时间序列中能够最大限度地表示一个类别的子序列[49]。如图 2.1 所示，对于经典的 *Gun/NoGun* 问题，最佳 shapelet 为红色标注部分所示，它能够对 *Gun/NoGun* 数据集进行准确的分类。最初的基于 shapelets 的分类算法是由 Ye 等人所提出的[8, 49]，作者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 来构建决策树分类器。Mueen 等人[7]针对 Ye 等人所提出的算法在时间效率以及可解释性上的不足，提出使用逻辑 shapelets 的思想来构建决策树。Rakthanmanon 等人[50]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 表示的快速 shapelets 发现算法，此算法在提升发现 shapelets 速率的同时保持了一定的分类准确性。为将时间序列 shapelets 扩展至多变量时间序列，Wistuba 等人[64]提出了一种快速 shapelets 发现算法。上述几种 shapelets 发现方法都是从子序列的预测能力入手进行评估，而 Grabocka 等人[72]

定义了发现 shapelets 的分类目标函数公式，并通过随机梯度下降方法[73]选取“真正”的 Top-*k* shapelets。

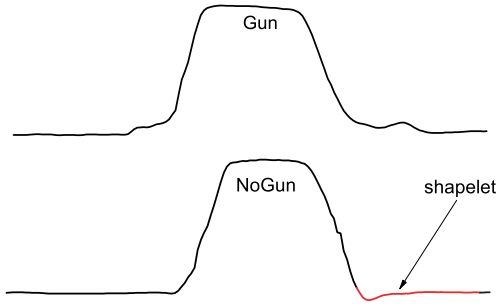

图 2.1 shapelet 示例

Figure 2.1 An example of shapelet

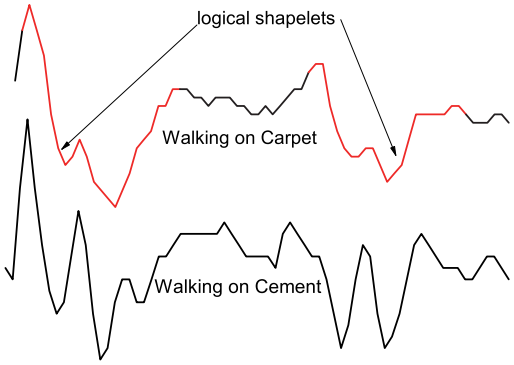

图 2.2 逻辑 shapelets 示例

Figure 2.2 An example of logical shapelets

上述方法均是在发现 shapelets 的同时构建分类器。而 Bagnall 等人[3]强调了数据转换在时间序列分类中的重要性，并提出解决时间序列分类问题的最简单方式是将原始时间序列映射到另一个辨别性特征比较容易被检测到的空间。文章为解决数据转换后可能引起的准确率大幅度差异问题，提出了一种基于分类器融合的方式来提升分类性能。Lines 等人[11]提出了一种基于 shapelet 转换的时间序列分类方法，即将发现 shapelets 与构建分类器的过程相分离。此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。另外，时间序列 shapelets 还广泛地被应用于时间序列聚类[74]、姿势识别[6]、天气预测[5]、早期分类[75, 76]等多个方面。

相比较于传统的时间序列分类方法，如基于 ED 或 DTW 的 1-NN 分类器，采用时间序列 shapelets 的优点主要在于两个方面。

（1）可解释性更强。虽然 1-NN 分类器是否具有可解释性是一个有争议的话

题，但是 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，其并没有指出与各个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区别（图 2.1 所示），即具有较强的辨别性；

（2）分类速度更快。首先，1-NN 算法是基于实例的方法，它需要对比整个时间序列数据集。而 shapelets 比较的是子序列（多数情况下比原始的时间序列要短）之间的相似性；其次，shapelets 能够非常简洁紧凑地表示一个类别，有时仅需要一个 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 shapelet 具有较快的分类速度。

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最具有前景的方式之一，它仍存在许多不足。以基于 shapelets 转换的方法为例，首先单个 shapelet 有时不足以区分不同的类别，此时需要将多个 shapelets 联合起来才能达到预期的效果。如图 2.2 所示，单个 shapelet 无法对 SonyAIBORobotSurface 数据集进行严格区分，此时需要联合多个 shapelets，从而组合为“逻辑 shapelets”才能准确区分图中的两种类别；其次，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，本章的贡献主要在两个方面。首先应用了一种基于智能存储和计算重用的技术，将发现 shapelets 的时间复杂度降低一个数量级。其次，在此基础上提出了一种基于合取或析取的逻辑 shapelets 的转换方法。具体做法是，首先找到 k 个最好的逻辑 shapelets（合取或析取），然后通过计算这些逻辑 shapelets 与原有时间序列的距离，将初始时间序列转换成新的非序列数据，同时也把时间序列的分类问题转化成了经典的分类问题。此方法在保持 shapelets 解释力的同时提升了分类的准确性。

本章主要结构如下。第 2.2 节阐述本章将用到的时间序列分类问题的背景知识。第 2.3 节将回顾 shapelets 转换算法。第 2.4 节将阐述本章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。第 2.5 节进行实验描述并评估所提出的算法。第 2.6 节进行相应的总结以及对以后的工作进行展望。

2.2 背景知识

时间序列分类问题被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。为方便起见，假设每一条时间序列具有相同数目的观测值。另外，本章主要处理单变量时间序列分类问题，除第 1 章提供的基本符号外，本章中涉及的其他符号如表 2.1 所示。

从公式（1.1）中可以得到，计算两个时间序列间距离的复杂度和时间序列的

长度呈线性关系。相比之下，本章将采用从时间序列 $\boldsymbol{x}$ 和 $\boldsymbol{x}'$推演而得到的五组数据来计算 $\boldsymbol{x}$ 和 $\boldsymbol{x}'$之间的规范化欧式距离，这些数据被称为充分统计量（sufficient statistics）[77] 。它们分别为$\sum x$，$\sum x'$，$\sum x^2$，$\sum (x')^2$ 和$\sum xx'$。通过此方式计算规范化的欧式距离，可以重复利用计算的中间结果并将原有时间复杂度从线性降低到常数级。本章将在第 2.4 节进行具体地阐述，相应的计算公式如下。

表 2.1 符号表

Table 2.1 Symbols table

符号	释义
$C(\boldsymbol{x}, \boldsymbol{x}')$	相关系数
$(\boldsymbol{s}, \tau)$	分裂点
$E(\boldsymbol{X})$	熵
$I(\boldsymbol{s}, \tau)$	信息增益
$G(\boldsymbol{X}, \tau)$	分裂间隔
L	orderline
min	shapelet 最小长度
max	shapelet 最大长度

$\boldsymbol{x}$ 和 $\boldsymbol{x}'$之间的欧式距离可以通过以下公式计算。

$$C(\boldsymbol{x}, \boldsymbol{x}') = \frac{\sum xx' - T\mu_x\mu_{x'}}{T\sigma_x\sigma_{x'}} \tag{2.1}$$

$$dist(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{2(1 - C(\boldsymbol{x}, \boldsymbol{x}'))} \tag{2.2}$$

公式（2.1）为 $\boldsymbol{x}$ 和 $\boldsymbol{x}'$之间的皮尔森相关系数（Pearson correlation coefficient）。当 $\boldsymbol{x}$ 和 $\boldsymbol{x}'$都经过规范化处理之后，它们的均值为 0，方差为 1，并且$\sum x^2 = \sum (x')^2 = T$，由此再根据公式（1.2），就可推导出公式（2.2）。

许多时间序列数据挖掘算法（如基于 ED 的 1-NN 分类，k-means 聚类，密度估计等）只需要对比相等长度的时间序列。相比之下，寻找时间序列 shapelets 需要测试一条较短的时间序列（或者说子序列，shapelet 等）是否在某一特定阈值下包含于某条更长的序列中。为得到此结果，较短时间序列须在较长序列上滑动以得到它们之间的最佳比对结果，此距离度量被称为子序列距离并定义为

$$subdist(\boldsymbol{x}, \boldsymbol{x}') = min(dist(\boldsymbol{x}, \boldsymbol{s}_{i,|x|})) \tag{2.3}$$

其中 $\boldsymbol{s}_{i,|x|}$表示时间序列 x'中从位置 i 开始长度为$|\boldsymbol{x}|$的子序列。或者根据充分统计量可表示为

$$sufficientdist(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{2(1 - C_s(\boldsymbol{x}, \boldsymbol{x}'))} \tag{2.4}$$

相应的

$$C_s(\boldsymbol{x}, \boldsymbol{x}') = \max_{0 \le l \le T - |x|} \frac{\sum_{i=1}^{|x|} x_i x'_{i+l} - |x| \mu_x \mu_{x'}}{|x| \sigma_x \sigma_{x'}} \tag{2.5}$$

在上述定义中，$\mu_{x'}$和 $\sigma_{x'}$分别表示 $\boldsymbol{x}'$中从 l+1 位置开始的$|\boldsymbol{x}|$个离散值的均值和标准差，之所以取最大值，是因为皮尔森相关系数是一种相似性度量，取值范围是[-1, 1],值越大越相关，而 ED 是一种非相似性度量，值越小越相似。注意，*subdist*()和 *sufficentdist*()函数得到的是两个序列间的最短距离。

定义 2.1：熵

假设数据集 $\boldsymbol{X}$ 中包含 N 条时间序列，类值的个数为$|\boldsymbol{Y}|$，类 y_i 在数据集 $\boldsymbol{X}$ 有 n_i 个实例，其中（$n_1+n_2+\ldots+n_{|Y|}=N$）。由于本章采用信息增益度量来评价一个 shapelet 的辨别能力，在此需要首先回顾一下熵的概念。

熵是随机变量不确定性的度量，数据集 $\boldsymbol{X}$ 的熵定义为

$$E(X) = -\sum_{i=1}^{|Y|} \frac{n_i}{N} \log(\frac{n_i}{N}) \qquad (2.6)$$

熵越大，数据集 $\boldsymbol{X}$ 中类值 $\boldsymbol{Y}$ 分布的不确定性也就越大。

定义 2.2：分裂点

一个分裂点（split）为一个元组$(\boldsymbol{s}, \tau)$，这里 $\boldsymbol{s}$ 表示一个子序列，τ表示一个距离的阈值。一个分裂点会将数据集 $\boldsymbol{X}$ 分割成两个不相交的子集，其中 $\boldsymbol{X}_{left}=\{\boldsymbol{x}:\boldsymbol{x}\in\boldsymbol{X}, subdist(\boldsymbol{s},\boldsymbol{x})\leqslant\tau\}$，$\boldsymbol{X}_{right}=\{x:x\in\boldsymbol{X}, subdist(\boldsymbol{s},\boldsymbol{x})>\tau\}$，并且 $N_1=|\boldsymbol{X}_{left}|$，$N_2=|\boldsymbol{X}_{right}|$。

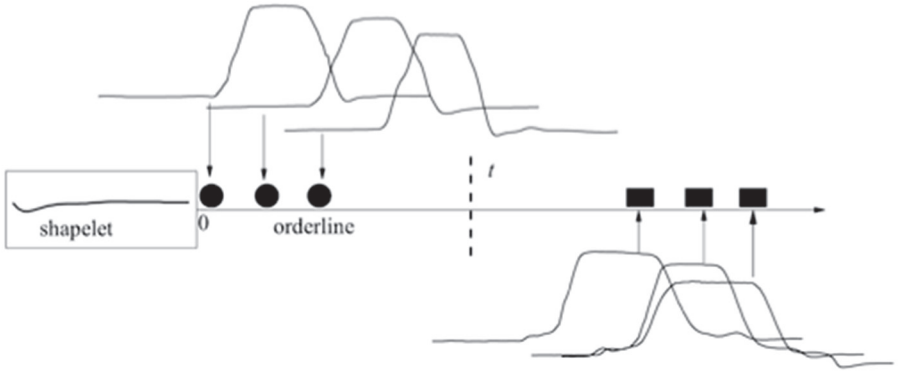

图 2.3 一个 shapelet 的 orderline 示例

Figure 2.3 An example of shapelet based orderline

定义 2.3：信息增益

分裂点的信息增益为

$$I(s,\tau) = E(X) - \frac{N_1}{N} E(X_{left}) - \frac{N_2}{N} E(X_{right}) \qquad (2.7)$$

该分裂点的信息增益度量的是，得知分裂点$(\boldsymbol{s}, \tau)$的信息之后使得时间序列数据集类值 $\boldsymbol{Y}$ 的信息的不确定性减少的程度。当然，信息增益越大，表明类值分布的不确定性减少的越多，该子序列 $\boldsymbol{s}$ 与 $\boldsymbol{Y}$ 就具有较强的互信息。

定义 2.4：分裂间隔

一个分裂点的分裂间隔的定义为

$$G(s,\tau)=\frac{1}{N_2}\sum_{x\in X_{right}} subdist(s,x)-\frac{1}{N_1}\sum_{x\in X_{left}} subdist(s,x) \qquad (2.8)$$

即为分裂点右侧实例与分裂子序列之间的平均距离减去分裂点左侧与分裂子序列间的平均距离。

定义 2.5：shapelet

数据集 $\boldsymbol{X}$ 中的一个 shapelet 是由 $\boldsymbol{X}$ 中某一实例的一个子序列 $\boldsymbol{s}$ 和一个分裂点的阈值 τ 所组成的元组（$\boldsymbol{s}$, τ），它试图将数据集 $\boldsymbol{X}$ 分割成两个不同的群组。从另一个角度来说，一个 shapelet 是所有分裂点中能够最好区分不同类别的子序列。即

$$shapelets=\arg\max_{(s,\tau)} I(s,\tau) \qquad (2.9)$$

定义 2.6：Orderline

一个 orderline L 是一个数组的一维表示形式，它按照递增的顺序记录了候选 shapelet 与数据集 $\boldsymbol{X}$ 中所有时间序列间的距离。

图 2.3 所示的例子阐释了 shapelet 和 orderline 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 $\boldsymbol{X}$ 中每条时间序列间的距离，τ 为分裂点的阈值。其中，数据集 $\boldsymbol{X}$ 为 *Gun/NoGun* 数据集，圆形表示 *NoGun* 序列与候选 shapelet 间的距离，它们间的距离较小；矩形表示 *Gun* 序列与候选 shapelet 间的距离，它们间的距离较大。Orderline 创建之后，就可以计算分裂点、分裂间隔以及候选的 shapelet 的信息增益等。最终选取最优的一个或多个 shapelets 进行分类。

需要注意的是，不同 shapelets 的信息增益可能出现相同的情况。当对 shapelets 排序时，信息增益较大的 shapelet 被优先选取；若两个 shapelets 的信息增益相同，则选择分裂间隔较大者；若两者的分裂间隔也相同，则优先选择长度较短者；若两者长度也相等，则认为此两个 shapelets 的大小是一致的。也就是说，分裂间隔以及 shapelet 的长度被定义为打破平局的方法。

2.3 Shapelets 转换技术

为合适地阐述本章的贡献，此处将首先描述 Lines 等人[11]提出的基于 shapelets 转换的时间序列分类方法。此方法的主要动机是将 shapelets 的发现过程与分类器的构建相分离，从而允许基于 shapelets 转换后的数据应用于传统分类器中。相应的算法实现主要包括三个步骤。首先，通过扫描一次数据集，找到最好的 k 个

shapelets；其次，使用 5 折交叉验证方法评估能够得到最好分类结果的 shapelets 的个数 k，然后将每一条时间序列转换成具有 k 个不具有时序关系属性的实例。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，从而创建了新的数据集；最后，将朴素贝叶斯（Naïve Bayes），贝叶斯网络，决策树，1-NN 等分类器应用于转换后的数据集进行分类。具体算法描述如图 2.4 所示。

```
算法: ShapeletFilter(X, min, max, k)
输入: 时间序列 T, 最小长度 min, 最大长度 max, shapelet 个数 k
输出: kShapelets
 1: kShapelets← ∅;
 2: for i←0 to |X| do {X中的每一条时间序列}
 3:   shapelets← ∅;
 4:   for l←min to max do {x_i中的每一个长度}
 5:     for u←0 to |X_i|-l+1 do {每一个起始位置}
 6:       S←X_i(u, l);
 7:       for m←0 to |X| do {候选 shapelet s 与每一条时间序
列的距离}
 8:         D_s←subdist(s, X_m);
 9:       orderline←sort(D_s);
10:       quality←assessCandidate(s, orderline, D_s);
11:       shapelets.add(s, quality);
12:   sortByQuality(shapelets);
13:   removeSelfSimilar(shapelets);
14:   kShapelets←merge(k, kShapelets, shapelets);
15: return kShapelets;
```

图 2.4 发现 k 个 shapelets

Figure 2.4 Find k best shapelets

图 2.4 中描述了从数据集中抽取 k 个最好 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）。每得到一个候选 shapelet，都需要计算其与每一条时间序列间的距离并构建 orderline（第 2-9 行）。如第 9-11 行所示，得到一个候选 shapelet 的 orderline 之后，寻找能得到最大信息增益的分裂点并将其加入到候选 shapelets 中。得到所有候选 shapelets 之后，根据它们信息增益的大小进行降序排列（第 12 行），具体打破平局的方法参见 2.2 节。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，此时需要保留较好的 shapelet 并移除与之自相似的 shapelet（第 13 行）。一旦得到一条时间序列中所有非自相似的 shapelets，就将它们和现有最好的 k 个 shapelets 相结合，并保留当前最好的 k 个（第 14 行）。

```
算法：TransformData(s,X)
输入：最好的 k 个 shapelets s，数据集 X
输出：转换后的数据集 output
1: output←∅;
2: for i←0 to |X| do {X 中的每一条时间序列}
3:   transformed←∅;
4:   for j←0 to |s| do {s 中的每一个 shapelet}
5:     dist←subdist(s_j, x_i);
6:     transformed.add(dist);
7:   output.add(transformed);
8: return output;
```

图 2.5 数据转换图

Figure 2.5 Data transformation

得到 k 个最好 shapelets 之后，利用它们将原有时间序列数据集转换成新数据集的方法如图 2.5 所示。数据转换时使用的子序列距离计算方法如图 2.6 所示（第 5 行）。Shapelets 与时间序列比对时，都进行了规范化处理（图 2.6 第 2，5 行）。对于 $\boldsymbol{X}$ 中的每一条时间序列 $\boldsymbol{x}_i$，它的子序列距离通过与 $\boldsymbol{s}_j$（$j=0$，...，k-1）计算得出，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）。

```
算法：subdist(x, x′)
输入：时间序列 x 和 x′, |x|≤|x′|
输出：x 与 x′的规范化距离
1: bestSum←MAX_VALUE;
2: x←zNorm(x);
3: for i←0 to |x′|-|x|+1 do {x′中的每一个起始位置}
4:   sum←0;
5:   z←zNorm(x′_{i,|x|});
6:   for j←0 to |x| do {计算欧式距离}
7:     sum←sum+(z_i-x_i)^2;
8:   bestSum←min(bestSum, sum);
9: return (bestSum/|x|)^{1/2};
```

图 2.6 规范化欧式距离

Figure 2.6 Normalized Euclidean distance

2.4 加速技术与逻辑 shapelets

本节将首先介绍一种更加有效率的距离计算方法，然后阐述逻辑 shapelets 转换的基本思想，并给出具体的算法描述。

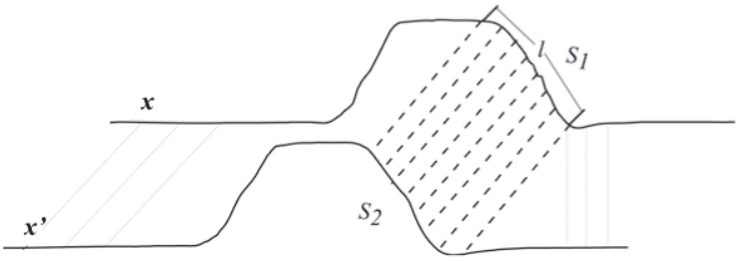

图 2.7 不同时间序列间的子序列距离

Figure 2.7 Subsequence distance for two different time series

算法： ***sufficentdist***$(u, l, Stats_{x,x'})$
输入： 起始位置 u，长度 l，$\boldsymbol{x}$ 与 $\boldsymbol{x}'$的充分统计量 $Stats_{x,x'}$
输出： $\boldsymbol{x}_{u,l}$ 与 $\boldsymbol{x}'$的规范化距离
1: $bestSum \leftarrow MAX\_VALUE$;
2: $\{\boldsymbol{M}, \boldsymbol{S}_x, \boldsymbol{S}_{x'}, \boldsymbol{S}_x^2, \boldsymbol{S}_{x'}^2\} \leftarrow Stats_{x,x'}$;
3: **for** $i \leftarrow 0$ to $|\boldsymbol{x}'|-|\boldsymbol{x}|+1$ **do** {$\boldsymbol{x}'$中的每一个起始位置}
4: $d \leftarrow$由公式（2.4）和（2.5）所得到的距离;
5: $bestSum \leftarrow min(bestSum, d)$;
6: **return** $(bestSum/|\boldsymbol{x}|)^{1/2}$;

图 2.8 根据充分统计量计算规范化欧几里得距离

Figure 2.8 Euclidean distance according to sufficient statistics

2.4.1 加速技术

在 2.3 节所描述的算法中，数据集 $\boldsymbol{X}$ 中的任意一条时间序列 $\boldsymbol{x}$ 中的起始于任何位置的任意长度的子序列 $\boldsymbol{s}_i$ 都可能是一个候选的 shapelet。当计算 $\boldsymbol{s}_i$ 与其他时间序列如 $\boldsymbol{x}'$等之间的距离时，需要将 $\boldsymbol{s}_i$ 在 $\boldsymbol{x}'$上逐步的滑动以找到两者间的子序列距离。如图 2.7 所示，对于数据集 $\boldsymbol{X}$ 中的任意两个实例 $\boldsymbol{x}$ 和 $\boldsymbol{x}'$，$\boldsymbol{s}_1$ 和 $\boldsymbol{s}_2$ 分别为长度为 l 的子序列，若要计算 $\boldsymbol{s}_1$ 和 $\boldsymbol{s}_2$ 之间的距离，需要计算 $\boldsymbol{s}_1$ 和 $\boldsymbol{s}_2$ 中每一对实值间的距离并取和。显然，当 $\boldsymbol{s}_1$ 和 $\boldsymbol{s}_2$ 的长度和起始位置不断变化时（图 2.7 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，计算效率会得到明显得提升，充分统计量（sufficient statistics）方法恰恰做到了这一点。

对每一对时间序列（$\boldsymbol{x}, \boldsymbol{x'}$），可以计算五个数组，$\boldsymbol{M}, \boldsymbol{S}_x, \boldsymbol{S}_{x'}, \boldsymbol{S}_{x^2}$ 和 $\boldsymbol{S}_{x'^2}$。其中数组 $\boldsymbol{S}_x, \boldsymbol{S}_{x'}$用于存储时间序列 $\boldsymbol{x}$ 和 $\boldsymbol{x'}$中观测值的累加和，即$\sum \boldsymbol{x}$，$\sum \boldsymbol{x'}$；数组 $\boldsymbol{S}_{x^2}$和 $\boldsymbol{S}_{x'^2}$ 用于存储 $\boldsymbol{x}$ 和 $\boldsymbol{x'}$中观测值平方的累加和，即$\sum x^2$，$\sum x'^2$。$\boldsymbol{M}$ 是一个用于存储 $\boldsymbol{x}$ 和 $\boldsymbol{x'}$ 中不同子序列乘积和的二维矩阵，$\sum xx'$。初始时，所有的数组或矩阵都被初始化为 0，并且数组或矩阵的索引从 0 开始。具体表示如下。

$$S_x[u] = \sum_{i=0}^{u} x_i \tag{2.10}$$

$$S_{x'}[v] = \sum_{i=0}^{v} x'_i \tag{2.11}$$

$$S_{x^2}[u] = \sum_{i=0}^{u} x_i^2 \tag{2.12}$$

$$S_{x'^2}[v] = \sum_{i=0}^{v} x'^2_i \tag{2.13}$$

$$M[u,v] = \begin{cases} \sum_{i=0}^{v} x_{i+t} x'_i & if\ u > v, \\ \sum_{i=0}^{u} x_i x'_{i+t} & if\ u \le v \end{cases} \tag{2.14}$$

在公式（2.14）中，$t=abs(u-v)$，u 和 v 分别为 $\boldsymbol{x}$ 和 $\boldsymbol{x'}$中子序列的的起始位置。这五个数组并称为 $Stats_{x,x'}$。计算过 $Stats_{x,x'}$之后，首先根据它们得到序列的均值与方差，然后根据公式（2.4）和（2.5）计算得到规范化的欧式距离。具体的算法描述如图 2.8 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。

$$u_x = \frac{S_x[u+l-1] - S_x[u-1]}{l} \tag{2.15}$$

$$u_{x'} = \frac{S_{x'}[v+l-1] - S_{x'}[v-1]}{l} \tag{2.16}$$

$$\sigma_x^{\ 2} = \frac{S_{x^2}[u+l-1] - S_{x^2}[u-1]}{l} - u_x^2 \tag{2.17}$$

$$\sigma_{x'}^{\ 2} = \frac{S_{x'^2}[v+l-1] - S_{x'^2}[v-1]}{l} - u_{x'}^2 \tag{2.18}$$

$$\sum_{i=0}^{l-1} x_{u+i} x'_{v+i} = M[u+l-1, v+l-1] - M[u-1, v-1] \tag{2.19}$$

在如图 2.8 所示的算法 $sufficentdist(u, l, Stats_{x,x'})$中，输入为时间序列 $\boldsymbol{x}$ 中某一子序列的起始位置 u、长度 l 以及事先计算好的 $Stats_{x,x'}$，子序列通过不断地滑动来寻找此子序列与 $\boldsymbol{x'}$间的最小距离。通过与算法 $subdist(x, x')$的比较可以得出，充分统计量的计算方式至少减少了一次 for 循环，即节省了 $O(m)$的时间复杂度。所以，

sufficentdist(u, l, $Stats_{x,x'}$)可以在常数级的时间复杂度内得到规范化的子序列距离。

2.4.2 逻辑 shapelets

Shapelet 是由数据集 $\boldsymbol{X}$ 中某一实例的一个子序列 $\boldsymbol{s}$ 和一个分裂点阈值 τ 所组成的元组($\boldsymbol{s}$, τ)，它是 $\boldsymbol{X}$ 中信息增益最大的子序列。很容易想到单个 shapelet 不足以区分不同的类别的情景。图 2.9 采用了一个简单的人造数据来进行说明。数据集 $\boldsymbol{X}$ 中包含两类时间序列，A 类时间序列中既包含有波峰又包含波谷，B 类时间序列中只包含波峰或者波谷。假设 shapelet $\boldsymbol{s}_1$，$\boldsymbol{s}_2$，$\boldsymbol{s}_3$ 的分裂阈值为 τ_1，τ_2，τ_3，从图中可以观察到，（$\boldsymbol{s}_1$，τ_1），（$\boldsymbol{s}_2$，τ_2），（$\boldsymbol{s}_3$，τ_3）都不能很好的将 A，B 两类区分开，此时需要将多个 shapelet 联合起来才能达到预期的效果。即将（$\boldsymbol{s}_1$，τ_1）和（$\boldsymbol{s}_2$，τ_2）联合起来时，可以发现，A 类和 B 类被明显的区分了。因此，为达到最好的区分效果，考虑将逻辑操作增加到 shapelet 的转换中。

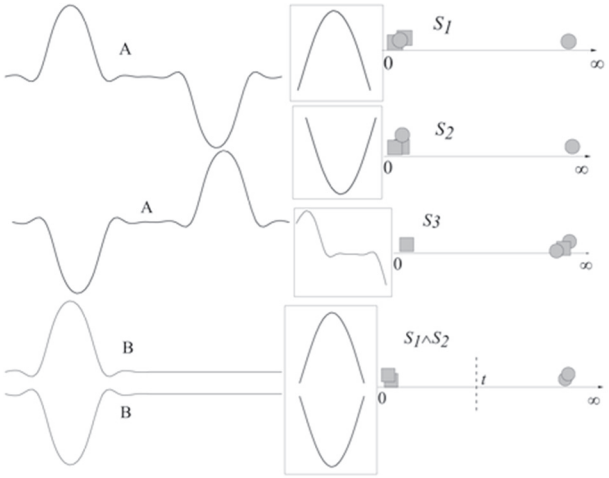

图 2.9 单个 shapelet 无法区分类别的例子

Figure 2.9 An example that shapelets cannot classify well

图 2.9 中所示的逻辑操作是合取操作，用符号“∧”表示，即（$\boldsymbol{s}_1$，τ_1）∧（$\boldsymbol{s}_2$，τ_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（$\boldsymbol{s}_1$，τ_1）∨（$\boldsymbol{s}_2$，τ_2）∧（$\boldsymbol{s}_3$，τ_3）这样一种析取操作和合取操作同时出现的现象，为保持逻辑操作的简洁性，本章只考虑两种情况，一种是只有合取操作，另一种只有析取操作。

定义 2.7：逻辑 shapelets

数据集 $\boldsymbol{X}$ 中的一个逻辑 shapelets 是一个元组（$<\boldsymbol{s}_1,\ldots,\boldsymbol{s}_n>$，$\tau$，$opt$），由 $\boldsymbol{X}$ 中一个或多个实例的一个或多个子序列所组成的，同时包含一个分裂点阈值 τ 以及一个逻辑操作标识 opt（opt 大于 0 为合取操作，小于 0 为析取操作，等于 0 表示一

个经典的 shapelet），它试图将数据集 $\boldsymbol{X}$ 分割成两个不同的群组。n 为逻辑 shapelets 中所包含经典 shapelet 的个数。

2.4.3 发现逻辑 shapelets

```
算法: LogicalShapeletFilter(X, min, max, k)
输入: 时间序列 X, 最小长度 min, 最大长度 max, 逻辑 shapelets 的个数 k
输出: kLGShapelets
 1: kLGShapelets← ∅;
 2: for i←0 to |X| do {X中的每一个时间序列}
 3:   logicalshapelets← ∅; tempLGShapelets← ∅;
 4:   for j←0 to |X| do {计算 xi 与 xj 的充分统计量}
 5:     Stats_xi,xj ←{M, S_xi, S_xj, S_xi^2, S_xj^2};
 6:   for l←min to max do {每一个可能的长度}
 7:     for u←0 to |Xi|-l+1 do {每一个起始位置}
 8:      for m←0 to |X| do {计算候选 shapelet S 与每一个时间序列的距离}
 9:        Ds←sufficentdist (u, l, Stats_xi,xj);
10:      orderline←sort(Ds);
11:      quality←assessCandidate(S, orderline, Ds);
12:      tempLGShapelets.add(S);
13:       tempLGShapelets←LogicalShapeletsCache(num, X, min, max,
tempLGShapelets, orderline);
14:      logicalshapelets.add(tempLGShapelets);
15:   sortByQuality(logicalshapelets);
16:   removeSelfSimilarlogical(logicalshapelets);
17: kLGShapelets←mergelogical(k, kLGShapelets, logicalshapelets);
18: return kLGShapelets;
```

图 2.10 发现 k 个逻辑 shapelets

Figure 2.10 Discovery k best logical shapelets

发现 k 个最好逻辑 shapelets 的算法如图 2.10 所示。与图 2.4 中算法相比较可以发现，两者的前半部分基本相同，但前者在寻找逻辑 shapelets 时采用了充分统计量技术（第 5 行）。另外，如第 13 行所示，每当找到一个 shapelet 并对它进行相应的评估之后，都将调用 *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。当一个时间序列中的所有逻辑 shapelets 都被评估过之后，调用 *sortByQuality*()函数进行排序（第 15 行），排序的方法与 2.2 节中介绍的相一致。然后移除相似的逻辑 shapelets（第 16 行）。若两个逻辑 shapelets 中存在来自同一个时间序列并有重叠之处的 shapelets，

则认为它们是相似的。一旦得到一个时间序列中所有不相似的逻辑 shapelets，就将它们和现有的最好的逻辑 shapelets 相结合，并保留当前最好的 k 个逻辑 shapelets（第 17 行）。注意当调用 *mergelogical*()函数时，仍需要进行相似性检测，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: LogicalShapeletsCache (X, min, max, n, lgshapelet, orderline)
输入: 时间序列 X,最小长度 min, 最大长度 max, 经典 shapelet 的个数 n, 子序列
lgshapelet, lgshapelet 与每个时间序列距离的顺序排列 orderline
输出: LogicalShapelets
1: LogicalShapelets←lgshapelet;
2: if n≤LogicalShapelets.size()
3:   return LogicalShapelets;
4: BestQuality←lgshapelet.quality;
5: tempLGShapelets← ∅;
6: for i←0 to |X| do {T 中的每一个时间序列}
7:   for j←0 to |X| do {计算 x 与 Tj 的充分统计量}
8:    Statsxi,xj ←{M, Sxi, Sxj, Sxi2,Sxj2};
9:  for l←min to max do {每一个可能的长度}
10:    for u←0 to |Xi |-l+1 do {每一个起始位置}
11:      for m←0 to |X| do {计算候选 shapelet S 与每一个时间序列的距离}
12:        Ds←sufficentdist (u, l, Statsxi,xj);
13:    lgorderline←sort(Ds);
14:    orderline←mergeTwoLines(lgorderline, orderline);
15:    quality←assessCandidate(S, orderline, Ds);
16:    if quality.betterthan(BestQuality)
17:      tempLGShapelets←lgshapelet;
18:      tempLGShapelets.add(S);
19:      BestQuality←quality;
20: LogicalShapelets←tempLGShapelets;
21: if LogicalShapelets.size() < n
22:     LogicalShapelets←LogicalShapeletsCache(T, min, max, n,
orderline);
23: return LogicalShapelets;
```

图 2.11 发现逻辑 shapelets

Figure 2.11 Discovery logical shapelets

寻找逻辑 shapelets 的算法如图 2.11 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 或者逻辑 shapelets 更好的 shapelets 时，递归循环终止（第 1-3 行，16-19 行），所以此算法得到的是合取或析取后能够使当前 shapelets 的信息增益最大的子序列。此处仍采用充分统计量来计算子序列距离（第 6-8 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 的 orderline 相结合，构建逻辑 shapelets 的 orderline（第 13-14 行）。但对于逻辑 shapelets 中的合取（∧）操作和析取（∨）操作，怎么定义它的 orderline 呢？如第 14 行所示，这里采用了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；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。

另外，采用逻辑 shapelets 进行时间序列转换时有可能发生过拟合（overfitting）现象。比如所发现的逻辑 shapelets 为（$\boldsymbol{s}_1$，τ_1）∨（$\boldsymbol{s}_2$，τ_2）∨…（$\boldsymbol{s}_n$，τ_n），其中 n 为数据集 $\boldsymbol{X}$ 中某一类时间序列（如 A）的实例个数，$\boldsymbol{s}_i$（i=1，2，…，n）为类 A 中不同实例的子序列。此时，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减少（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。

2.4.4 转换时间序列

```
算法：TransformData(s, X, 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 |X| do {X 中的每一个时间序列}
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(sufficentdist(u, l, Stats_x,x'));
8:       else if (opt≤0) {析取操作或没有逻辑操作}
9:         dist←Min(sufficentdist(u, l, Stats_x,x'));
10:   transformed.add(dist);
11: output.add(transformed);
12: return output;
```

图 2.12 基于逻辑 shapelets 的数据转换

Figure 2.12 Logical shapelets based data transformation

采用 shapelets 或逻辑 shapelets 转换的主要目的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。如图 2.12 所示，距离计算仍采用了 2.4.1 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 0 时，表示合取操作，此时取逻辑 shapelets 中与当前时间序列距离最大者作为转换后实例的属性值。*opt* 小于 0 则取最小值，等于 0 时表示当前逻辑 shapelets 是经典的 shapelet。初始的时间序列数据集被转换之后，即可应用于多种传统的分类算法进行分类。第 2.4.5 节将验证所提出算法的分类准确率。

```
算法: Accuracy(C, X_train, X_test)
输入: 分类器 f, 训练数据集 X_train, 测试数据集 X_test
输出: 分类准确率 accuracy
1: D←TransformData(s, X_train, opt);
2: T←TransformData(s, X_test, opt);
3: f'←BuildClassify(f, D);
4: accuracy←0;
5: for i←0 to |X| do {T 中的每一个时间序列}
6:   class_label←getClasslabel(x_i);
7:   predict_label←ClassifyInstance(f',x_i);
8:   if predict_label.equal(class_label)
9:     accuracy←accuracy+1;
10:accuracy←accuracy/|X|;
11:return accuracy;
```

图 2.13 计算分类准确率

Figure 2.13 Compute classification accuracy

2.4.5 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。如图 2.13 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 f 可以是 Naïve Bayes、决策树分类器、k-NN、随机森林[55, 78]（Random Forest），旋转森林[79, 80]（Rotation Forest），贝叶斯网络[81]（Bayesian Network）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，则计数值加一（第 5-9 行），最终返回该分类器的分类准确率（第 10-11 行）。

2.5 实验评估

此部分将评估本章所应用的加速技术以及逻辑 shapelets 转换技术。由于相关的时间复杂度分析已经在 2.4.1 节中给出，本节将首先从时间上对比应用充分统计量技术后的 ShapeletAcc 算法与 Lines 等人提出的 ShapeletFilter 算法，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[37]以及 Mueen 等人[7]所提供。所有的算法和实验都是在 WEKA 框架[82]下实现的。

表 2.2 发现 k 个 shapelets 的时间（秒）对比

Table 2.2 Time comsuming for discovering k shapelets (seconds)

Data	ShapeletAcc	ShapeletFilter
SonyAIBORobotSurface	2.37	9.34
SonyAIBORobotSurfaceII	3.65	14.99
synthetic_control	2420.81	3217.67
TwoLeadECG	5.56	26.75
Beef	1147.72	42346.07
CBF	31.95	248.39
Coffee	217.95	5359.17
Cricket	18.00	177.83
DiatomSizeReduction	150.06	3597.34
ECG200	227.34	1071.20
ECGFiveDay	24.35	218.68
FaceFour	327.92	9402.13
Gun_Point	150.55	1467.21
ItalyPowerDemand	4.40	5.21
Motes	4.29	21.93
OliveOil	1590.67	75996.02
Symbols	436.14	15042.84

2.5.1 参数设置

本章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，k 值仅仅表示所抽取逻辑 shapelets 的数量，它并不影响每一个得到的逻辑 shapelets 的质量，但无法保证设置的 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 值过大时，可能造成过拟合现象或降低较重要逻辑 shapelet 的辨别性。在实验中，为保持 k 值设置的

一致性以及简洁性，将 k 值设置为 $T/2$，这里 T 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 $T/2$ 时，将补入经典的不相似的 shapelet 以满足相应的个数。若发现的 shapelet 的个数小于 $T/2$，此时将 k 值设置为当前 shapelet 或逻辑 shapelets 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 的长度范围，参数设置不对时可能发现不了最具有辨别性的 shapelet，从而对分类器的准确率造成影响。同样为保持最大和最小长度设置的一致性以及简洁性，统一将最小长度设置为 $T/10$，最大长度设置为 $T/2$，T 为时间序列中属性的个数。

2.5.2 速度比对

此处对比的是 ShapeletFilter 算法和使用充分统计量之后的改进算法 ShapeletAcc，并不是对比发现逻辑 shapelets 与经典的 shapelet 的快慢。尽管采用充分统计量之后，发现 shapelet 的速率得到了很大的提升，但由于在寻找逻辑 shapelets 时需要不断地重复地遍历数据集，一般情况下，发现逻辑 shapelets 需要消耗较多的时间。

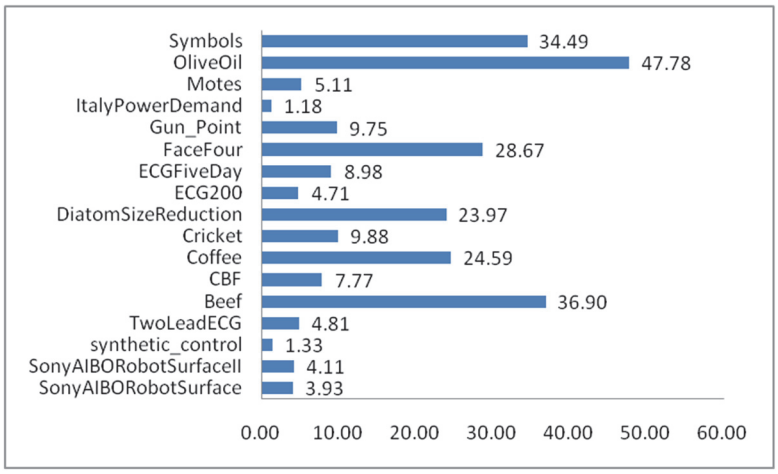

图 2.14 ShapeletAcc 与 ShapeletFilter 的加速比

Figure 2.14 Speed up ratio between ShapeletAcc and ShapeletFilter

虽说改进了原有算法的效率，但发现 shapelet 的过程仍是一项耗时的工作，所以此处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 小时之内得到结果的数据集。所采用的数据集如表 2.2 所示。发现 k 个最优 shapelets 的过程是在各个数据的训练集上完成的，计时单位为秒，所得数值均为 10 次测量求平均值所得到的结果。从表中可以明显观察到，加速后的 ShapeletsAcc 算法明显优

于 ShapeletsFilter 算法，在 OliveOil 数据集上，最大加速比近 48 倍，更直观的加速比图形如图 2.14 所示。

2.5.3 逻辑 shapelets 转换对比于 shapelets 转换

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 shapelet 转换的时间序列分类算法。在进行逻辑 shapelets 转换时，将考虑只有合取或者只有析取这两种情况，分别将两种算法命名 LG-AND 和 LG-OR。

为避免偏差，在进行实验时，所有的逻辑 shapelets 和 shapelet 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任何分类器的数据。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 2.3 所示[1]。

表 2.3 逻辑 shapelets vs. shapelets

Table 2.3 Logical shapelets vs. shapelets

Algorithm	NB	C4.5	RF	1-NN	RoF	BN	Data
LG-AND	91.35%	**84.86%**	**88.52%**	91.02%	89.85%	91.68%	SonyAIBORobotSurface
LG-OR	**93.34%**	**84.86%**	86.19%	**93.51%**	**93.01%**	93.51%	
Shapelet	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%**	SonyAIBORobotSurfaceII
LG-OR	87.09%	**75.97%**	**87.62%**	81.01%	87.72%	84.78%	
Shapelet	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%	TwoLeadECG
LG-OR	**99.39%**	**91.57%**	97.89%	99.91%	99.21%	**99.91%**	
Shapelet	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%**	MoteStrain
LG-OR	85.62%	**80.27%**	**80.51%**	85.22%	84.11%	80.67%	
Shapelet	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%**	ItalyPowerDemand
LG-OR	92.52%	**92.91%**	90.77%	**96.02%**	**94.95%**	**92.23%**	
Shapelet	92.42%	**92.91%**	90.77%	93.49%	92.91%	91.74%	
LG-AND	**99.54%**	**98.95%**	**99.42%**	**100%**	99.54%	99.42%	ECGFiveDays
LG-OR	**99.54%**	**98.95%**	**99.42%**	99.54%	**100%**	**99.54%**	
Shapelet	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。平局的情况大多

[1] Naïve Bayes: NB；Random Forest: RF；Rotation Forest: RoF；Bayesian Network: BN

出现在以信息增益为度量的决策树或决策树组合模型中，比如三者在 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。

2.5.4 逻辑 shapelets 转换对比于其他分类器

1-NN 分类器是当前解决时间序列分类问题最好的分类器之一，为进一步验证所提出的基于逻辑 shapelets 转换的时间序列分类算法的性能，此部分将对比基于欧式距离的 1-NN 分类器和基于逻辑 shapelets 转换的 1-NN 分类器。另外，此处也对比了基于逻辑 shapelets 转换后的 C4.5 分类器和直接采用 shapelet 实现的决策树的分类准确性。从表 2.4 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式距离的 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 shapelet 实现的决策树分类器。

表 2.4 准确率对比

Table 2.4 Accuracy

Data	1-NN	LG-1NN	ShapeletTree	LG-C4.5
SonyAIBORobotSurface	69.56%	**91.02%**	84.53%	84.86%
SonyAIBORobotSurfaceII	85.94%	**89.30%**	75.97%	75.97%
TwoLeadECG	74.72%	**99.39%**	85.07%	91.57%
MoteStrain	87.86%	**88.66%**	82.51%	79.79%
ItalyPowerDemand	**95.53%**	93.20%	89.21%	92.91%
ECGFiveDays	79.68%	**100%**	77.47%	98.95%
Cricket	87.76%	**91.84%**	60.20%	82.65%
Cricket_new	43.75%	**57.81%**	50%	56.25%

2.5.5 实验总结

首先，通过时间复杂度分析和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 shapelet 的发现效率；其次，通过与基于 shapelet 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试验，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基于 shapelet 转换技术的缺点，比如训练时间较长，k 值的设定以及对 shapelet

的最大最小长度把握比较困难等。

2.6 本章小结

本章介绍了一种用于加速计算规范化欧式距离的充分统计量技术，并提出了使用逻辑 shapelets 转换进行时间序列分类的思想。通过实验验证了所提出算法的高效性和准确性。但基于 shapelet 转换的分类技术的最大缺点是训练时间相对较长，一种可行的缓解方式是通过对训练集采样，仅使用原始数据的一小部分子集来发现 shapelet 或逻辑 shapelets，但此方法有可能降低分类的准确性。未来将考虑引进其他的缩短训练时间的方式。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 shapelet 来代替寻找 k 个逻辑 shapelets 进行数据转换。

3 基于 Shapelet 剪枝和覆盖的时间序列分类算法

上一章描述了基于逻辑 shapelets 进行空间转换的分类算法，尽管采用逻辑 shapelets 能够增强 shapelets 的可解释性，但其仍未解决相似性 shapelets（或逻辑 shapelets）较多，不能直接确定最终用于空间转换的 shapelets 的数量等缺点。针对这些问题，本章将描述一种普遍适用于 shapelets 和逻辑 shapelets 的剪枝和覆盖算法。

3.1 引言

如第二章所述，时间序列 shapelets 是时间序列中能够最大限度地表示一个类别的子序列[8, 49]。基于时间序列 shapelets 的分类方法具有分类准确性高，分类速度快，可解释性强等优点。该类算法可分两大类，第一类方法在发现时间序列 shapelets 的同时构造分类器，即通过递归地寻找当前最好的 shapelets 建立决策树。第二类方法将时间序列 shapelets 的选择过程当作单独的预处理步骤来处理，即通过 shapelets 转换，将 shapelets 的发现和分类器的构造相分离，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。

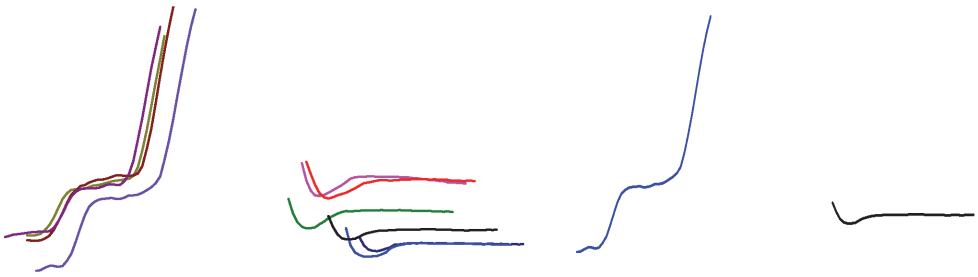

图 3.1 (a) *Gun/NoGun* 问题的 10 个最好的 shapelets; (b) *Gun/NoGun* 问题的 2 个最具有辨别性的 shapelets

Figure 3.1 (a) Best 10 shapelets for *Gun/NoGun*; (b) Best 2 discriminative shapelets for *Gun/NoGun*

虽然上一章所提出的算法提升了 shapelets 的发现效率和可解释力，但基于 shapelets 转换的时间序列分类方法仍存在两个主要问题。一是 shapelets 转换时，所选取的时间序列 shapelets 的数量影响着分类结果的好坏，而用于产生最好分类结果的 shapelets 数量是很难确定的，目前普遍采用的方法是通过交叉验证来获取，但此方法会消费大量的时间；二是被选择的 shapelets 之间往往存在着较大的相似

性。比如说，对于经典的 *Gun/NoGun* 问题，通过 ShapeletFilter 算法[11]而得到的前 10 个 shapelets 如图 3.1（a）所示。从图中可以明显地观察到，这 10 个 shapelets 自然地聚成了两簇，即它们之间存在着极大的相似性。易知，只需要找出这两簇中最具有辨别性的 2 个 shapelets（如图 3.2（b）所示）就可以涵盖这 10 个 shapelets，从而允许其它有辨别性的 shapelets 参与到数据转换中并提高分类准确率。对于数据转换时所需 shapelets 的数量，本章将借鉴属性选择中的序列覆盖思想[83]，提出了 shapelet 覆盖的概念，并根据 shapelets 对实例的覆盖情况来选取 shapelets。

本章的主要贡献有：

（1）本章提出了一种简单有效的 shapelet 剪枝方法，用于过滤掉相似的 shapelets，并大大减少候选 shapelets 样本的数量；

（2）本章提出了一种基于 shapelets 覆盖的方法来确定用于数据转换的 shapelets 的数量，并保证了 shapelets 对实例的覆盖；

（3）阐述了如何将本章所提出算法扩展到逻辑 shapelets 转换中；

（4）将所提出的算法和其他基于 shapelets 的时间序列分类算法以及基于 ED 和 DTW 的 1-NN 分类器作对比，并结合实例解析阐明了所提出算法的分类准确性和可解释性。

本章的组织结构如下。第 2 部分将描述本章所提出的 shapelet 选择和 shapelet 覆盖等方法，并阐述基于辨别性 shapelets 转换的时间序列分类算法的基本思想。第 3 部分将进行实验描述并评估所提出的算法。第 4 部分将通过一些实例分析来展示本章所提出方法的优点；第 5 部分进行相应的总结以及对进一步工作的展望。

3.2 辨别性 shapelets 转换技术

本章提出的转换算法通过四个主要步骤处理 shapelets。首先，通过扫描一次数据集，找到所有候选的 shapelets；其次，通过使用本章提出的 shapelet 剪枝方法，从候选 shapelets 中移除（除自相似 shaplets 之外的）相似的 shapelets；再次，使用本章新提出的 shapelet 覆盖方法来确定用于数据转换的 shapelets 的数目 k；最后，将每一条时间序列转换成具有 k 个相互独立属性的实例，并将多种不同的分类策略应用到时间序列分类中。

3.2.1 候选 shapelets

生成所有候选 shapelets 的过程如图 3.2 所示。此算法处理数据的过程与第二章中的 shapelet 生成算法相类似（参照图 2.4）。它们之间的主要区别在于，该算法

的目标是找到所有候选 shapelets 而不是最好的 k 个，原因是任意时间序列的任意长度的子序列都有可能是辨别性 shapelet，因此可直接省略参数 k 的设置。需要注意的是，当自相似的 shapelets 从一条时间序列中移除之后（第 13 行），需要将剩余的子序列添加到候选的 shapelets 中，并按照它们信息增益的降序排列。此过程不断迭代直至处理完所有的时间序列（第 14 行）。

```
算法: CandidateShapelets(X, min, max)
输入: 时间序列 X, 最小长度 min,最大长度 max
输出: CandidateShapelets
 1: CandidateShapelets← ∅;
 2: for i←0 to |X| do {X中的每一条时间序列}
 3:   shapelets← ∅;
 4:   for l←min to max do {x_i中的每一个可能长度}
 5:     for u←0 to |x_i|-l+1 do {每一个起始位置}
 6:       s←x_i(u, l);
 7:       for m←0 to |X| do {计算候选 shapelet s 与每一条
时间序列的距离}
 8:         D_s←subdist(s, x_m);
 9:       orderline←sort(D_s);
10:       quality←assessCandidate(s, orderline, D_s);
11:       shapelets.add(s, quality);
12:   sortByQuality(shapelets);
13:   removeSelfSimilar(shapelets);
14:   CandidateShapelets.add(shapelets);
15: return CandidateShapelets;
```

图 3.2 寻找候选 shapelets

Figure 3.2 Find candidate shapelets

3.2.2 Shapelet 剪枝

Shapelet 剪枝方法的主要动机是剪去相似 shapelets，从而允许更多有辨别性的 shapelets 参与到数据转换中。为更好的描述本章所提出的 shapelet 剪枝方法，此处将首先描述“相似 shapelets”的概念。

定义 3.1: 相似 *shapelets*

对于两个 shapelets，$(\boldsymbol{s}_1, \tau_1)$和$(\boldsymbol{s}_2, \tau_2)$，$(\boldsymbol{s}_1, \tau_1)$优于$(\boldsymbol{s}_2, \tau_2)$，即 $I(\boldsymbol{s}_1, \tau_1) > I(\boldsymbol{s}_2, \tau_2)$。若它们所在时间序列的类标相同，并且两者间的子序列距离 $subdist(\boldsymbol{s}_1, \boldsymbol{s}_2)$小于$(\boldsymbol{s}_1, \tau_1)$的分裂阈值$\tau_1$，那么两者为相似 shapelets。

此处需要注意相似 shapelets 与自相似 shapelets 间的不同。相似 shapelets 的定义主要强调了形状的相似性，作比较的两个 shapelets 往往来自不同的时间序列。而自相似 shapelets 存在于同一条时间序列中并具有重叠的部分。由于τ_1是能够获取最大信息增益的距离阈值，当 $subdist(\boldsymbol{s}_1, \boldsymbol{s}_2)$小于$\tau_1$时，表示 $\boldsymbol{s}_1$ 和 $\boldsymbol{s}_2$ 具有相似的形状并且 $\boldsymbol{s}_1$ 能够在大部分情况下替代 $\boldsymbol{s}_2$。例如，从图 3.1（a）中可以观察到，解决 *Gun/NoGun* 问题的做好的十个 shapelets 间存在着极大的形状相似性。Shapelet 剪枝之后，过滤了相似的 shapelets 并能够使用两个不相似的 shapelets 来代表最好的十个（图 3.1（b））。过滤掉相似 shapelets 的过程如图 3.3 所示。

```
算法: ShapeletsPrune(candidateShapelets)
输入: 候选 shapelets candidateShapelets
输出: NoSimilarShapelets
 1: NoSimilarShapelets← ∅;
 2: size←candidateShapelets.size();
 3: for i←0 to size do {每一个候选 shapelet}
 4:   selectShapelet←candidateShapelets.get(i);
 5:   NoSimilarShapelets.add(selectShapelet);
 6:   Distance←selectShapelet.splitThreshold;
 7:   for j←i+1 to size do {每一个候选 shapelet }
 8:     ToPrune←candidateShapelets.get(j);
 9:     Dist←subdist(selectShapelet, ToPrune);
10:     if (Dist <= Distance)
11:       if (selectShapelet.classLabel != ToPrune.classLabel)
12:         NoSimilarShapelets.add(ToPrune);
13:     else
14:       NoSimilarShapelets.add(ToPrune);
15:   candidateShapelets←NoSimilarShapelets
16:   size←candidateShapelets.size();
17: return NoSimilarShapelets;
```

图 3.3 Shapelet 剪枝

Figure 3.3 Shapelet pruning

在循环的开始和结束处，都需要初始化候选 shapelets 的数目，因为基本上每一次循环，候选 shapelets 的数目都将会减少（第 2 和 16 行）。候选 shapelets 中的每一个 shapelet 都将与优于它的 shapelets 作比较并决定是否剪去它（第 3-14 行）。若 *ToPrune* shapelet 和 *selectShapelet* shapelet 间的子序列距离小于等于 *selectShapelet* 的分裂阈值，将接着比较它们的类标，当它们类标也相同时剪去 *ToPrune*，否则保留 *ToPrune* 并将它应用于下一次循环中（第 10-14 行）。最终得到用于 shapelet 覆盖步骤的所有不相似 shapelets。

3.2.3 Shapelet 覆盖

如前所述，Lines 等人使用 5 折交叉验证方法来确定用于数据转换的 shapelets 的数量。但是这是一项十分耗时的工作，尤其当训练数据量比较大时它的缺点会更明显，原因是此方法需要运行多次才能估计拥有较好分类准确率的 shapelets 的数量。为了有效并简洁地获取用于最终转换的 shapelets 的数量，本章提出了一个新的概念，“shapelet 覆盖”。下面将介绍此概念。

```
算法: ShapeletsCoverage(NoSimilarShapelets, δ)
输入: 不相似的 Shapelets NoSimilarShapelets,覆盖参数 δ
输出: 辨别性 shapelets Shapelets
 1: Shapelets← ∅;
 2: Initialize(InstanceTable);
 3: size←NoSimilarShapelets.size();
 4: for i←0 to size do {每一个候选 shapelet}
 5:   Selected← false;
 6:   InstanceID←NoSimilarShapelets.get(i).InstanceID;
 7:   for j←0 to InstanceID.size() do {每一个 instance ID}
 8:     if (InstanceTable.contains(InstanceID_j))
 9:       Selected← True;
10:       if (InstanceID_j.number() < δ)
11:         if (InstanceID_j.number()+1 >= δ)
12:           InstanceTable.remove(InstanceID_j);
13:         else
14:           InstanceID_j.number()++;
15:    if (Selected)
16:      Shapelets.add(NoSimilarShapelets.get(i));
17: return Shapelets;
```

图 3.4 Shapelet 覆盖

Figure 3.4 Shapelet coverage

定义 3.2: *Shapelet* 覆盖

给定一个 shapelet (s_1, τ_1)和与它相对应的 orderline L，分裂阈值 τ_1 将 L 上的数据点或者说数据集 $\boldsymbol{X}$ 分割成两部分，左边是与 shapelet 间子序列距离小于 τ_l 的时间序列数据集 $\boldsymbol{X}_l$，右边是与 shapelet 间子序列距离大于 τ_l 的时间序列数据集 $\boldsymbol{X}_r$，此处认为 $\boldsymbol{X}_l$ 能够被 shapelet 成功覆盖，$\boldsymbol{X}_r$ 不能够被 shapelet 覆盖。参照图 2.3，根据此定义，可以认为圆形所代表的时间序列是能够被 shapelet 覆盖的数据集。

本章提出的 shapelet 覆盖的概念借鉴了属性选择算法中序列覆盖的思想[83]。这里可以将 shapelet 当做时间序列的一个特殊的特征来处理。两者间的主要区别在

于shapelet覆盖是一种近似的覆盖方法，它并不意味着某一时间序列包含此shapelet，而是它们的子序列距离小于某一给定的阈值。通过 shapelet 覆盖来选择 shapelets 的过程如图 3.4 所示。

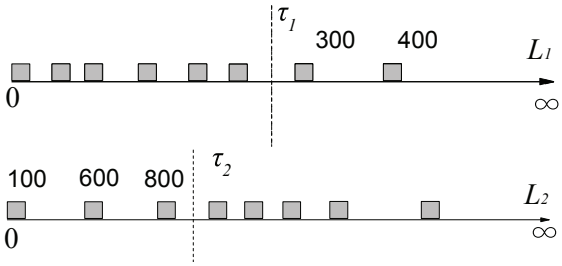

图 3.5 两个 orderline 示例

Figure 3.5 Examples of two orderlines

TSID	Count
100	1
200	1
300	0
400	0
500	1
600	1
700	1
800	1

(a)

TSID	Count
100	2
200	1
300	0
400	0
500	1
600	2
700	1
800	2

(b)

图 3.6 (a) 第一次循环时的 *InstanceTable*;

(b) 第二次循环时的 InstanceTable.

Figure 3.6 (a) *InstanceTable* for the first iteration;

(b) *InstanceTable* for the second iteration.

本章的算法通过扫描一次不相似的 shapelets 来得到最终用于转换的辨别性 shapelets（第 4 行）。当一个 shapelet 被访问之后，被此 shapelet 覆盖的时间序列将被移除掉。但是在真正的时间序列分类任务中，考虑到分类准确率，可能希望用多个 shapelets 去覆盖一个实例。为实现此想法，此处引进了一个 shapelet 覆盖参数 δ。若一条时间序列被至少 δ 个 shapelets 覆盖，这条时间序列将不会被进一步考虑（第 8-14 行）。*InstanceTale* 数组用来存储每一条时间序列的计数值。初始时，*InstanceTable* 中的每一个值被初始化为 0（第 2 行）。为更加清楚地解释此过程，此处采用了一个简单的实例进行说明，如图 3.5 和 3.6 所示。

在图 3.5 中，orderline L_1 和 L_2 展示了两个不同 shapelets 区分实例的情况。假设 $\delta=2$。第一次循环时，第一个 shapelet 的 orderline 为 L_1，*InstanceTable* 如图 3.6 (a)所示。时间序列$\{300, 400\}$与第一个 shapelet 之间的距离大于τ，所以它们未被此

shapelet 覆盖并且它们的计数值仍为 0。由于每个实例的计数值更新之后都没有达到阈值 $\delta = 2$，所以没有时间序列被移除。在第二次循环时，orderline 为 L_2，实例{100, 600, 800}与该 shapelet 间的距离小于阈值τ，所以它们的计数值都增加 1（如图 3.6 (b)所示）。按照前述的规定，时间序列{100, 600, 800}可以从 *InstanceTable* 中移除。需要注意的是，当且仅当一个 shapelet 能覆盖 *InstanceTable* 中的任何一个实例时，它才能加入最终的 shapelets 中（第 15-16 行）。

3.2.4 整合 shapelets 剪枝和覆盖方法

为强调通过 shapelet 剪枝和 shapelet 覆盖方法所得到的 shapelets 与其他 shapelets 之间的区别，本章将它们称之为“**辨别性 shapelets**”。

```
算法: PruneAndCoverage(candidateShapelets, δ)
输入: 候选 shapelets candidateShapelets, 覆盖参数 δ
输出: Shapelets
 1: Shapelets← ∅;
 2: size←candidateShapelets.size();
 3: Initialize(InstanceTable);
 4: for i←0 to size do {每一个候选 shapelet}
 5:   if(InstanceTable.isEmpty())
 6:     break;
 7:   selectShapelet←candidateShapelets.get(i);
 8:   Shapelets←UpdateCoverage(selectShapelet, δ);
 9:   for j←i+1 to size do {每一个候选 shapelet}
10:     ToPrune←candidateShapelets.get(j);
11:     PruneShapelets(selectShapelet, ToPrune);
12:   size←UpdateSize();
13: return Shapelets;
```

图 3.7 集成 shapelet 剪枝和 shapelet 覆盖

Figure 3.7 Combine shapelet pruning with shapelet coverage

在前述的 3.2.2 和 3.3.3 节中，首先通过对候选 shapelet 剪枝，获得不相似的 shapelets，然后根据 shapelet 覆盖方法，从不相似的 shapelets 中得到用于数据转换的辨别性 shapelets。为更快地获取辨别性 shapelets，可考虑将 shapelet 剪枝与 shapelet 覆盖方法相结合，即直接通过对候选 shapelets 的遍历而得到最终的 shapelets。此部分的目的是将 shapelet 剪枝和 shapelet 覆盖相整合。该方法相比较于前面分别计算 shapelet 剪枝和 shapelet 覆盖的方法，至少节省了一个时间复杂度为 $O(m)$的循环，m 为不相似 shapelets 的个数。该整合版本如图 3.7 所示。

相比较于 shapelet 剪枝和 shapelet 覆盖方法，该整合方法能够直接从候选的 shapelets 中寻找出用于最终转换的 shapelets。第 1-3 行首先初始化了 *InstanceTable* 和候选 shapelets 的数量。然后，对于每一个候选的 shapelet，在剪去与它相似的 shapelets 之前，需要检查 *InstanceTable* 是否为空。若 *InstanceTable* 为空，由于此方法将不会选择任何 shapelet，循环就可以直接中断（第 5-6 行）。此后，继续更新每一条时间序列的覆盖情况并剪去相似的 shapelets（第 7-11 行），具体方法请参照 3.3.2 和 3.3.3 节。当一个候选 shapelet 访问结束后，因为候选 shapelets 的数量可能发生了变化，所以得更新 *size* 的大小（第 12 行）。最终返回辨别性 shapelets。

3.2.5 辨别性逻辑 shapelets

是否可以将 shapelet 剪枝和 shapelet 覆盖技术应用于逻辑 shapelets 中呢？此部分将讨论定义相似逻辑 shapelets 和逻辑 shapelets 覆盖的可能性。

逻辑 shapelets 的概念是有 Mueen 等人[7]提出的。数据集 $\boldsymbol{X}$ 中的一个逻辑 shapelets 是一个元组（$<\boldsymbol{s}_1,\ldots,\boldsymbol{s}_n>$，$\tau$，$opt$），其中$<\boldsymbol{s}_1,\ldots,\boldsymbol{s}_n>$由 $\boldsymbol{X}$ 中一条或多条实例的一条或多条子序列所组成的，同时包含一个分裂点阈值 τ 以及一个逻辑操作标识 opt（opt = 1 表示逻辑合取操作∧，opt =-1 表示逻辑析取操作∨），它试图将数据集 $\boldsymbol{X}$ 分割成两个不同的群组，其中 n 为逻辑 shapelets 中所包含经典 shapelet 的个数。为保持逻辑操作的简洁性，只考虑两种情况，一是只有合取操作，一是只有析取操作。对于逻辑 shapelets 的 orderline。若是“∧”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；若是“∨”操作，则选择最小的距离。

由于 shapelet 覆盖技术主要强调了 orderline 左边的实例，所以逻辑 shapelets 覆盖的概念是比较容易定义的。对于一个逻辑 shapelets $(<\boldsymbol{s}_1,\ldots,\boldsymbol{s}_n>, \tau, opt)$，若一条时间序列被此逻辑 shapelets 覆盖，那么他们之间的子序列距离小于该逻辑 shapelets 的分裂阈值τ。然而对于两个逻辑 shapelets，$(<\boldsymbol{s}_1,\ldots,\boldsymbol{s}_n>, \tau_1, opt)$ 和 $(<\boldsymbol{s}_1,\ldots,\boldsymbol{s}_{n'}>, \tau_2, opt)$，此处 $(<\boldsymbol{s}_1,\ldots,\boldsymbol{s}_n>, \tau_1, opt)$优于 $(<\boldsymbol{s}_1,\ldots,\boldsymbol{s}_{n'}>, \tau_2, opt)$。当 opt = 1 时，若它们所在时间序列的类标相同，并且两者间的子序列距离的最小值 $subdist(\boldsymbol{s}_i, \boldsymbol{s}_j)$ 小于 $(<\boldsymbol{s}_1,\ldots,\boldsymbol{s}_n>, \tau_1, opt)$的分裂阈值$\tau_1$，那么两者为相似的逻辑 shapelets。如果 opt = -1，若两者间的子序列距离的最大值 $subdist(\boldsymbol{s}_i, \boldsymbol{s}_j)$小于$(<\boldsymbol{s}_1,\ldots,\boldsymbol{s}_n>, \tau_1, opt)$的分裂阈值$\tau_1$，那么两者为相似的逻辑 shapelets。此处 $1 \le i \le n$，$1 \le j \le n'$。为统一起见，将经过逻辑 shapelets 剪枝和逻辑 shapelets 覆盖方法过滤后的逻辑 shapelets 命名为**辨别性逻辑 shapelets**。

3.3 实验与评价

此部分将评估辨别性 shapelets（ShapeletSelection 算法）在时间序列分类问题上的表现。由于改进了原有算法，首先将讨论用于数据转换时 shapelet 覆盖参数 δ 的最优值；其次将提出的算法与 ShapeletFilter 算法, ClusterShapelet 算法以及经典的 1-NN 分类器相比较[11, 56]，阐明辨别性 shapelets 在时间序列分类问题上的优势。所有的算法均是在 WEKA 框架下，使用 Java 代码实现的。所采用的数据集均包括训练集和测试集，其中 shapelets 的发现和分类器的构建都是在训练集上进行的，测试集仅用于测试分类器的分类准确率。

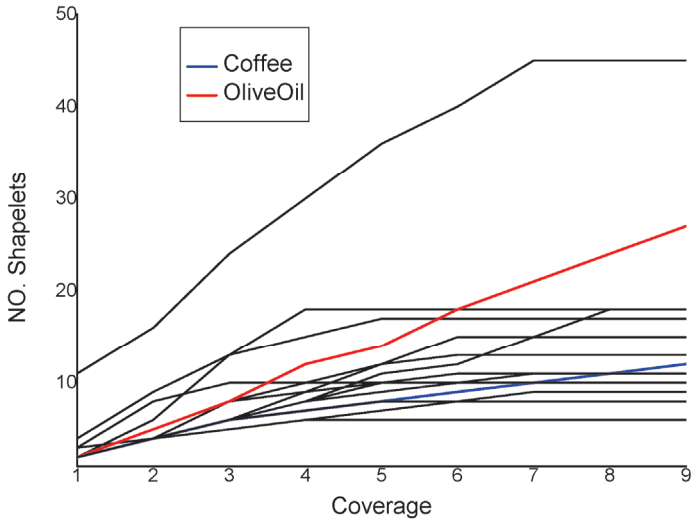

图 3.8 覆盖参数 δ 不断增长时 shapelet 数量的变化

Figure 3.8 Number of shapelet varies with δ

3.3.1 实验数据集

由于寻找 shapelets 需要花费大量的时间，本节从 UCR 数据集中选取 17 个具有相对较少属性或实例的数据集进行实验（如表 3.1 所示）。所有的数据集在获取时均已经被区分为训练集和测试集，其中 shapelets 的发现，分类器的构建以及执行时间的比对都是训练集上进行的，而分类准确率是在测试集上获取的。另外，由于 shapelet 的长度参数，*min* 和 *max*，决定了候选 shapelet 的长度范围，设置不正确时将会导致发现不了最具有辨别性的 shapelet，并且分类准确率也会收到影响。为保证覆盖足够多的候选 shapelet，*min* 设置为 $T/11$，*max* 设置为 $T/2$，T 为时间序列的长度。

表 3.1 数据集汇总

Table 3.1 Summary of datasets

Data	实例个数 (Train/Test)	长度	类值个数	Shapelet 长度 (*min*/*max*)
CBF	30/900	129	3	11/64
Coffee	28/28	287	2	26/143
ECG200	100/100	97	2	8/48
ECGFiveDays	23/861	137	2	12/68
Cricket	9/64	309	2	28/154
Cricket_new	9/98	309	2	28/154
DiatomSizeReduction	16/306	346	4	31/173
FaceFour	24/88	351	4	31/175
Gun_Point	50/150	151	2	13/75
ItalyPowerDemand	67/1029	25	2	2/12
Motes	20/1252	85	2	7/42
OliveOil	30/30	571	4	51/285
SonyAIBORobotII	27/953	66	2	6/33
SonyAIBORobot	20/601	71	2	6/35
Symbols	25/995	399	6	36/199
synthetic_control	300/300	61	6	5/30
TwoLeadECG	23/1139	83	2	7/41

3.3.2 覆盖参数

在时间序列分类任务中，为分类准确率考虑，此处希望用多个 shapelets 去覆盖某个实例。此部分的目标就是找到一个合适的 shapelet 覆盖参数 δ。为获取最好的 δ，本节首先讨论 δ 和 shapelets 数量间的关系；然后试图通过经验估计来设置能够获得最好分类准确率的覆盖参数 δ。

图 3.8 展示了 δ 变化时 17 个数据集（详见表 3.1）上发现 shapelets 数目的变化。所有实验都是在这些数据的训练集上进行的。从图中可以很明显的观察到，初始时，shapelets 的数目会随着 δ 的增长而增长，当 shapelets 的数目无法满足 δ 的需求时会到达 shapelets 数目的峰值并保持不变。而数据集 *Coffee* 和 *OliveOil* 是两个例外情况，它分别在图 3.8 中用蓝线和红线标出。原因是当覆盖参数 δ 从 1 增长到 9 时，这两个数据集有足够的非相似 shapelets 来满足覆盖参数的需求。

本节同样测试了 δ 变化时分类准确率的变化。首先选取四个数据集 *CBF*，*Coffee*，*Gun_Point* 和 *SonyAIBORobotSurface*。如图 3.9 所示，从这四幅图中并不能发现固定的规律，但是有一点是非常明显的，当 $\delta=4$ 时，所有 4 个数据集的分类准确率都达到了最大值。顾据此推测 $\delta=4$ 是用于分类的最好的覆盖值。

为验证该假设，我们在其他数据集上做了相应的实验，如图 3.10 所示。注意实验中的所有分类准确率都是通过在训练集上建立分类器，然后在测试集上进行测试所得到的。从图 3.10 中可以发现，当 $\delta=4$ 时，大部分数据集的分类准确率接

近或达到了它们的最大值。所以为保持简洁性和一致性，在接下来的实验中将覆盖参数 δ 设置为 4。

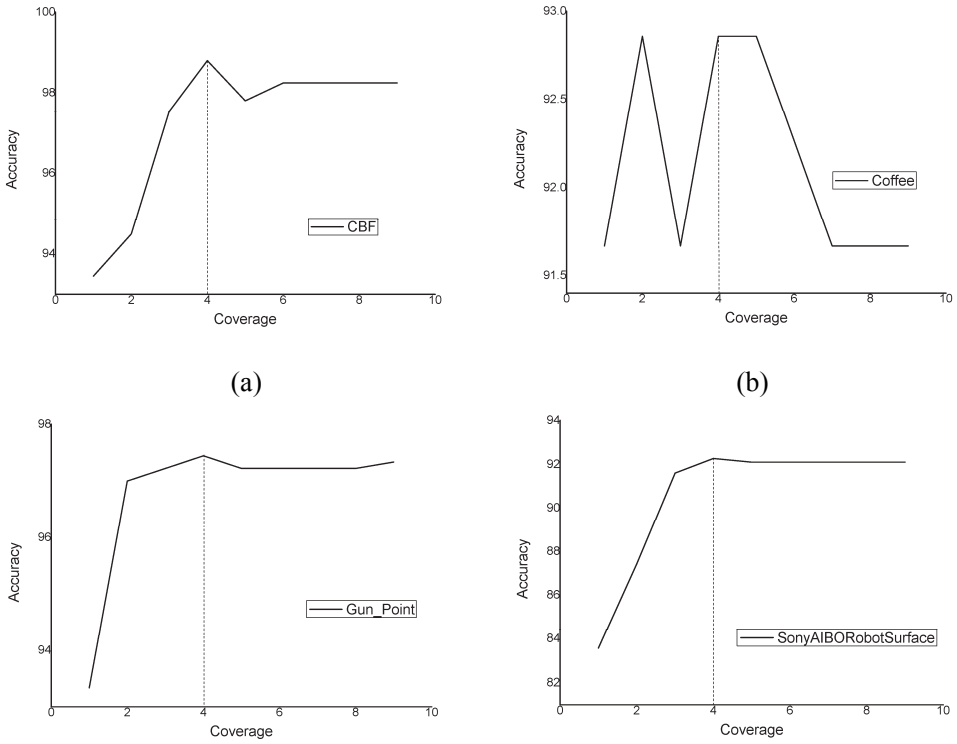

图 3.9 (a) δ 变化时数据集 CBF 的分类准确率; (b) δ 变化时数据集 Coffee 的分类准确率; (c) δ 变化时数据集 Gun_Point 的分类准确率; (d) δ 变化时数据集 SonyAIBORobotSurface 的分类准确率.

Figure 3.9 (a) Accuracy of CBF when δ varies; (b) Accuracy of Coffee when δ varies; (c) Accuracy of Gun_Point when δ varies; (d) Accuracy of SonyAIBORobotSurface when δ varies.

3.3.3 与 shapelets 树对比

本节的首要目的是说明本章所提出的 shapelet 剪枝和覆盖算法能够提高分类准确率，此部分将对比直接在时间序列上建立决策树的 ShapeletTree 算法和经过 shapelet 转换（包括传统 shapelet 和本章所提出的辨别性 shapelet）后的 C4.5 算法在多个数据集上的分类准确率（如表 3.2 所示）。本次测试所用于转换时间序列的 shapelet 数量如表 3.3 第二列所示。通过对比 ShapeletTree 和基于辨别性 shapelet 的 C4.5 算法，后者在 5 个数据集上取得了较好的分类准确率，两者在 9 个数据集上取得了相同的分类准确率。经过双尾 t 检验和 Wilcoxon 秩检验（显著性水平为 0.05），

两者不存在显著性差异，所以不存在证据表明本章所提出的 shapelet 剪枝和覆盖算法降低了分类准确率。

表 3.2 不同决策树间的准确率对比(%)

Table 3.2 Comparison of accuracy among different decision trees (%)

Data	ShapeletTree	C4.5 (classical shapelets)	C4.5 (discriminative shapelets)
CBF	79.56	79.56	**97.89**
Coffee	**89.29**	**89.29**	**89.29**
ECG200	20.00	17.00	**21.00**
ECGFiveDays	**98.95**	**98.95**	**98.95**
Cricket	**50.00**	**50.00**	**50.00**
Cricket_new	**66.33**	**66.33**	**66.33**
DiatomSizeReduction	**62.09**	**62.09**	**62.09**
FaceFour	**77.27**	75.00	75.00
Gun_Point	**91.33**	**91.33**	**91.33**
ItalyPowerDemand	88.44	88.44	**94.17**
MoteStrain	78.19	78.19	**81.79**
OliveOil	**70.00**	60.00	**70.00**
SonyAIBORobotII	**79.54**	**79.54**	**79.54**
SonyAIBORobot	73.21	**84.86**	**84.86**
Symbols	**68.94**	65.43	62.21
synthetic_control	**93.33**	**93.33**	92.33
TwoLeadECG	**86.39**	**86.39**	**86.39**

3.3.4 与 shapelets 转换方法对比

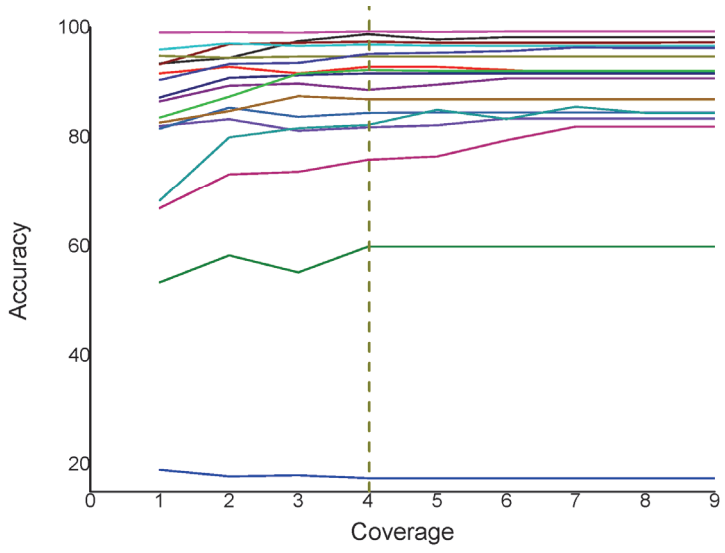

图 3.10 覆盖参数 δ 不断增长时平均分类准确率的变化

Figure 3.10 Varies of average accuracy when δ varies

此部分将对比辨别性 shapelets 和经典 shapelets 在多个时间序列数据集上的分

类准确率。所有的分类器都是在基于辨别性 shapelets 或经典 shapelets 转换后的数据上训练和测试的。由于本章采用了多个分类模型和数据集来评估所提出的算法，为有效地阐述所提出的 ShapeletSelection 算法的优点，除分类准确率之外，本章同时采用同一分类模型在相同数据集上的相对准确率和不同分类模型在同一数据集上的平均准确率来进行比对。表 3.3 展示了覆盖参数 δ=4 时在 17 个数据集上的测试结果。表中的第 1 列为所采用的数据的名称；第 2 列是与 δ 相对应的 shapelets 的个数。“Mean”记录了所采用的 6 个分类器的分类准确率平均值。为方便与 ShapeletFilter 算法做对比，此处用 ShapeletFilter 算法做了相同的实验并在表 3.3 的最后一列列出了 ShapeletFilter 算法的分类准确率平均值。用于 ShapeletFilter 算法中的 shapelets 的数量与 ShapeletSelection 算法相同。结果表明，本章所提出方法在 17 个数据集中的 14 个上优于 ShapeletFilter 算法，其中在 OliveOil 数据集上获得了 10%的准确率提升。

表 3.3 δ = 4 时 ShapeletSelection 算法在多个数据集上的分类准确率 (%)

Table 3.3 Classification accuracy of ShapeletSelectin on datasets when δ = 4 (%)

Data	Num	NB	C4.5	RF	1NN	RoF	BN	Mean	SF
CBF	9	99.78	97.89	98.00	99.67	99.33	98.11	**98.80**	92.91
Coffee	7	92.86	89.29	92.86	92.86	96.43	92.86	**92.86**	91.67
ECG200	18	21.00	21.00	18.00	12.00	14.00	19.00	17.50	**18.67**
ECGFiveDays	10	98.95	98.95	99.07	99.65	99.30	99.65	**99.26**	99.19
Cricket	6	70.31	50.00	56.25	59.38	51.56	71.88	**59.90**	59.64
Cricket_new	6	98.98	66.33	97.96	97.96	89.80	98.98	91.67	**92.52**
DiatomSizeReduction	8	85.62	62.09	91.50	85.29	83.99	82.35	**81.81**	78.87
FaceFour	6	81.82	75.00	93.18	97.73	87.5	96.59	**88.64**	84.47
Gun_Point	15	96.67	91.33	99.33	98.67	98.67	100.0	**97.44**	96.00
ItalyPowerDemand	10	95.92	94.17	95.53	94.85	96.02	91.74	**94.70**	93.42
MoteStrain	9	84.74	81.79	82.59	85.94	86.34	85.22	**84.44**	82.95
OliveOil	12	93.33	70.00	80.00	80.00	80.00	90.00	**82.22**	72.22
SonyAIBORobotII	10	91.08	79.54	87.30	86.78	88.35	88.56	86.94	**88.53**
SonyAIBORobot	7	96.01	84.86	88.69	94.68	94.34	95.01	**92.26**	86.99
Symbols	8	76.28	62.21	78.09	82.31	74.17	82.21	**75.88**	68.96
synthetic_control	30	98.00	92.33	96.67	99.00	97.67	97.67	**96.89**	95.17
TwoLeadECG	8	98.77	86.39	96.05	98.95	96.22	94.82	**95.20**	92.55

表 3.4 展示了辨别性 shapelets 对比于经典 shapelets 的相对准确率。结果表明，基于辨别性 shapelets 转换的数据能够提升所有六个分类器的分类准确率。针对所有的情况，ShapeletSelection 转换后的分类器在近四分之三的数据集上具有较好的结果，其中 Naïve Bayes 分类器的平均准确性提升了 3.64%，单项分类准确率最高提升了 23.33%。

3.3.5 辨别性 shapelets 与 shapelets 聚类对比

为缓解最好的 k 个 shapelets 中存在相似 shapelets 的问题，在发现最好的 k 个 shapelets 之后，Hill 等人[56]采用层次聚类来获取不相似的 shapelets。如 Rakthanmanon 等人以及 Mueen 等人所述[7, 50]，发现 shapelets 的时间复杂度为 $O(n^2m^4)$，其中 n 为数据集中时间序列的条数，m 为时间序列的长度。由于 shapelets 聚类和 shapelet 剪枝操作都是在产生 shapelets 后进行的，一般情况下，他们将会比 ShapeletFilter 花费较多的时间。然而，层次聚类在最坏情况下的时间复杂度为 $O(p^3)$，即使经过优化，层次聚类的复杂度仍有 $O(p^2logp)$，而本章所提出的 shapelets 剪枝和覆盖方法的时间复杂度为 $O(p^2)$，其中 p 为候选 shapelets 的个数。为方便对比，本章将 ClusterShapelet 算法的类簇个数设置为辨别性 shapelets 的个数（表 4.3 第二列），并做了相应的对比实验。表 3.5 展示了 ClusterShapelet 算法在不同数据集上的分类准确率。

表 3.4 辨别性 shapelets 与经典 shapelets 间的相对准确率

Table 3.4 Relative accuracy between discriminative shapelets and classical shapelets

Data	NB	C4.5	RF	1NN	RoF	BN	Mean
CBF	1.11	**18.33**	7.11	0.78	7.55	0.44	5.89
Coffee	0.00	0.00	3.57	0.00	3.57	0.00	1.19
ECG200	-1.00	4.00	-2.00	-3.00	-4.00	-1.00	-1.17
ECGFiveDays	0.00	0.00	-0.35	0.23	0.35	0.23	0.07
Cricket	0.00	0.00	0.00	3.13	-1.57	0.00	0.26
Cricket_new	0.00	0.00	0.00	2.04	-7.14	0.00	-0.85
DiatomSizeReduction	0.00	0.00	7.19	-6.54	**16.34**	0.65	2.94
FaceFour	5.68	0.00	-2.27	2.28	15.91	3.41	4.17
Gun_Point	2.67	0.00	5.33	-1.33	0.00	2.00	1.44
ItalyPowerDemand	2.04	5.73	0.29	-0.87	2.63	-2.14	1.28
MoteStrain	0.00	3.60	0.16	0.56	2.79	1.83	1.49
OliveOil	**23.33**	10.00	10.00	6.67	-3.33	**13.33**	**10.00**
SonyAIBORobotSurfaceII	-1.26	0.00	-0.63	-4.20	-0.95	-2.52	-1.59
SonyAIBORobotSurface	10.15	0.00	0.50	4.66	7.98	8.32	5.27
Symbols	8.54	-3.22	**11.66**	**8.94**	2.81	12.76	6.92
synthetic_control	2.67	-1.00	1.67	2.00	1.00	4.00	1.72
TwoLeadECG	7.99	0.00	3.95	2.99	1.40	-0.44	2.65
Average Improved	**3.64**	2.20	2.72	1.08	2.67	2.41	2.45
Data Sets Improved	9	5	11	11	11	10	**14**

表 3.6 展示了辨别性 shapelets 对比于 shapelets 聚类算法的相对准确率。结果同样表明，基于辨别性 shapelets 转换的数据能够提升所有六个分类器的分类准确率。针对所有的情况，我们的方法在近五分之四的数据集上具有较好的结果，

RandomForest 分类器的平均准确性提升了 7.87%，Naïve Bayes 分类器在 Cricket 数据集上的准确率升了 24.49%。

表 3.5 ClusterShapelet 算法在多个数据集上的分类准确率 (%)

Table 3.5 Testing accuracy of ClusterShapelet on diffierent classifiers (%)

Data	NO.	NB	C4.5	RF	1NN	RoF	BN	Mean
CBF	9	84.22	78.11	77.33	**86.22**	83.00	81.89	81.80
Coffee	7	92.86	89.29	89.29	**96.43**	92.86	92.86	92.26
ECG200	18	16.00	18.00	17.00	17.00	15.00	**21.00**	17.33
ECGFiveDays	10	86.53	85.60	88.50	86.06	**92.45**	89.66	88.13
Cricket_new	6	53.13	51.56	54.69	56.25	65.63	**81.25**	60.42
Cricket	6	74.49	67.35	72.45	83.67	**85.71**	83.67	77.89
DiatomSizeReduction	8	77.45	59.15	83.99	**91.83**	87.58	88.56	81.43
FaceFour	6	77.27	55.68	65.91	77.27	75.00	**81.82**	72.16
Gun_Point	15	90.00	90.00	98.67	97.33	97.33	**99.33**	95.44
ItalyPowerDemand	10	95.72	95.43	94.85	94.95	94.66	**96.31**	95.32
MoteStrain	9	**89.46**	69.25	78.99	83.31	89.30	85.46	82.63
OliveOil	12	80.00	63.33	**83.33**	**83.33**	80.00	76.67	77.78
SonyAIBORobotSurfaceII	10	**91.29**	87.62	78.28	90.24	88.88	90.56	87.81
SonyAIBORobotSurface	7	75.87	66.89	76.71	72.21	78.70	**83.69**	75.68
Symbols	8	84.42	64.82	69.95	86.63	**87.74**	76.08	78.27
synthetic_control	30	90.67	90.67	93.67	92.67	**95.67**	89.00	92.06
TwoLeadECG	8	95.70	76.03	93.77	94.82	95.52	**96.75**	92.10

3.3.6 其他分类器

采用 ED 或者 DTW 的 1-NN 分类器是目前解决时间序列分类问题的最好的方法之一。此部分将采用 ED 或者 DTW 的 1-NN 分类器与在辨别性 shapelets 转换后的数据上训练和测试的 1-NN 分类器做对比实验。实验结果如表 3.7 所示。在基于 ED 度量方面，基于辨别性 shapelets 转换的 1-NN 分类器（Prune_ED）明显优于初始的 1-NN 分类器。此处同时对比了基于 DTW 的 1-NN（1NN_DTW）和在转换后数据集上训练和测试的 Prune_DTW。没有任何证据表明辨别性 shapelets 转换使得分类准确率变低

3.4 实例解析

此部分将说明辨别性 shapelets 在解决几个现实问题时的优点，并展现所提出算法在真实数据集上的可解释性。

表 3.6 ShapeletSelection 与 ClusterShapelet 间的相对准确率 (%)

Table 3.6 Relative accuracies trained with discriminative shapelets vs. shapelets clustering (%)

Data	NB	C4.5	RF	1NN	RoF	BN	Mean
CBF	15.56	**19.78**	20.67	13.45	**16.33**	**16.22**	**17.00**
Coffee	0.00	0.00	3.57	-3.57	3.57	0.00	0.60
ECG200	5.00	3.00	1.00	-5.00	-1.00	-2.00	0.17
ECGFiveDays	12.42	13.35	10.57	13.59	6.85	9.99	11.13
Cricket_new	17.19	-1.56	1.56	3.13	-14.07	-9.37	-0.52
Cricket	**24.49**	-1.02	25.51	14.29	4.09	15.31	13.78
DiatomSizeReduction	8.17	2.94	7.51	-6.54	-3.59	-6.21	0.38
FaceFour	4.55	19.32	**27.27**	20.46	12.50	14.77	16.48
Gun_Point	6.67	1.33	0.66	1.34	1.34	0.67	2.00
ItalyPowerDemand	0.20	-1.26	0.68	-0.10	1.37	-4.57	-0.62
MoteStrain	-4.72	12.54	3.60	2.63	-2.96	-0.24	1.81
OliveOil	13.33	6.67	-3.33	-3.33	0.00	13.33	4.44
SonyAIBORobotSurfaceII	-0.21	-8.08	9.02	-3.46	-0.53	-2.00	-0.87
SonyAIBORobotSurface	20.14	17.97	11.98	**22.47**	15.64	11.32	16.58
Symbols	-8.14	-2.61	8.14	-4.32	-13.57	6.13	-2.39
synthetic_control	7.33	1.66	3.00	6.33	2.00	8.67	4.83
TwoLeadECG	3.07	10.36	2.28	4.13	0.70	-1.93	3.10
Average Improved	7.36	5.55	**7.87**	4.44	1.69	4.12	5.17
Data Sets Improved	13	11	**16**	10	10	9	13

表 3.7 基于欧氏距离和 DTW 的 1-NN 分类器在原始数据上的分类准确率和在辨别性 shapelets 转换后数据集上建立的 1-NN 分类器的分类器准确率比对(%)

Table 3.7 Accuracy of DTW and Euclidean distance with 1-NN on raw dataset and the 1-NN classifier built on the discriminative shapelets transformed dataset (%)

Data	1NN_ED	1NN_DTW	Prune_ED	Prune_DTW
CBF	85.23	**99.67**	**99.67**	98.89
Coffee	75.00	82.10	92.86	**96.43**
ECG200	11.00	**23.00**	12.00	21.00
ECGFiveDays	79.68	76.77	**99.65**	98.72
Cricket_new	43.75	**87.50**	59.38	64.06
Cricket	87.76	98.98	97.96	**100.00**
DiatomSizeReduction	93.46	**96.73**	85.29	96.08
FaceFour	79.41	82.95	97.73	**98.86**
Gun_Point	91.33	90.67	98.67	**99.33**
ItalyPowerDemand	95.53	95.04	94.85	**95.82**
MoteStrain	**87.86**	83.47	85.94	83.47
OliveOil	**86.67**	83.33	80.00	73.33
SonyAIBORobotSurfaceII	85.94	83.11	86.78	**87.51**
SonyAIBORobotSurface	69.56	72.55	**94.68**	88.19
Symbols	89.95	**94.97**	82.31	77.29
synthetic_control	88.00	**99.33**	99.00	92.67
TwoLeadECG	74.72	90.43	**98.95**	97.54

3.4.1 Sony AIBO Robot 数据集

如图 3.11 所示，SonyAIBORobotSurface 数据反映了 SONY AIBO 机器人在两个不同的地面上行走时，加速计上数值的变化，这两个地面分别为水泥地（cement）和地毯（carpet）。它的训练集包含 20 个长度为 70 的实例，每一个实例记录了该机器人在不同地面上的一个行走周期。测试集包含 601 个实例。每一类的行走周期示例如图 3.11 所示。从图中可以看出，当机器人行走在水泥地时，记录中存在着较多的大涨大落。当 shapelets 覆盖参数 $\delta = 4$ 时，通过在此数据集上进行实验，发现辨别性 shapelets 的数量为七个，其中三个的类标为“*walking on cement*”，如图 3.12 所示。

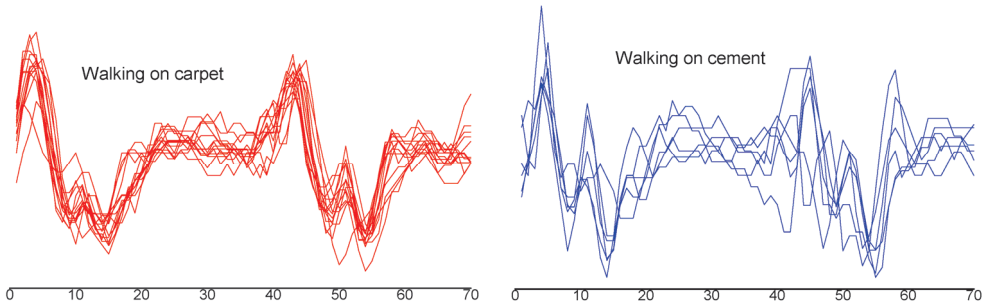

图 3.11 SONY AIBO 数据集中的两类时间序列

Figure 3.11 Two different classes of SONY AIBO dataset

基于 ED 和 DTW 的 1-NN 分类器在此数据集上的分类准确率分别为 67.72%和 72.55%。然而，在同样的任务中，通过使用辨别性 shapelets 转换，该算法在 1-NN 分类器上得到了 94.68%和 88.19%的分类准确率，明显优于在原始数据集上的基于欧式距离和 DTW 的 1-NN 分类器。

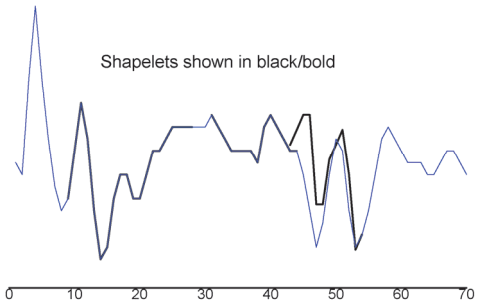

图 3.12 标记为 *walking on cement* 的辨别性 shapelets

Figure 3.12 Discriminative shapelets for class walking on cement

3.4.2 Coffee 数据集

Coffee 数据集是咖啡的一个光谱图集。在化学计量学中，它能够用于食物种类的分类，此任务能被广泛应用于食品安全和质量检测中。图 3.13 中所示即为咖啡的两个变种 *Arabica* 和 *Robusta* 的光谱图。此数据集包含 56 个实例，其中 28 个作为训练集，另外 28 个作为测试集，数据集中的每一个实例的长度都为 286.

实验中找到了三个类标为 *Arabica* 的辨别性 shapelets，如图 3.14 所示，这三个 shapelets 充分反映了 *Arabica* 序列的主要特征。由于 *Arabica* 和 *Robusta* 在总体上是相似的，因此很难区分 Coffee 数据集中的实例。基于 ED 和 DTW 的 1-NN 分类器在此数据集上的分类准确率分别为 75%和 82.1%。而基于辨别性 shapelets 转换的方法在六个不同分类器上的平均准确率为 92.68%，并且 Prune-DTW 的分类准确率高达 96.43%。基于辨别性 shapelets 转换的方法明显优于基于 ED/DTW 的 1-NN，并且该方法保持了 shapelets 所具有的可解释性。

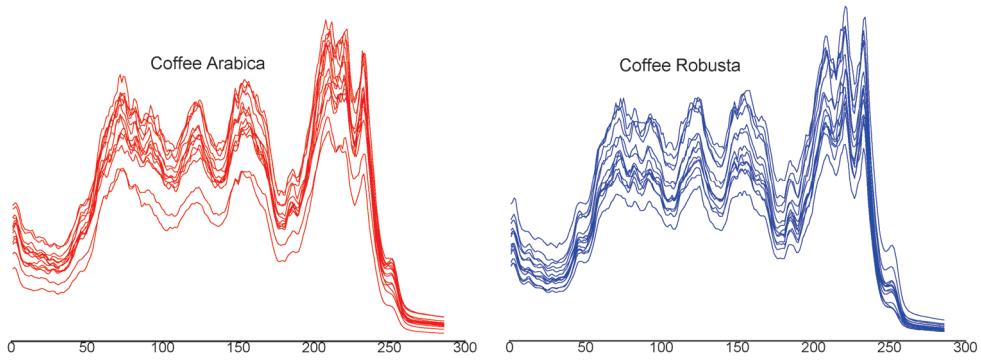

图 3.13 Coffee 数据集中的两类时间序列

Figure 3.13 Two different classes for Coffee dataset

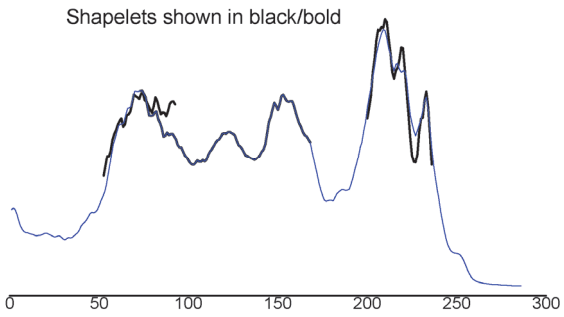

图 3.14 标记为 *Arabica* 辨别性 shapelets

Figure 3.14 Discriminative shapelets for class *Arabica*

3.4.3 ECG 数据集

ECG Five Days 数据集是一个 ECG 的时间序列，包含有 890 条实例，每一条实例的长度为 136，其中 23 条实例为训练集，另外 861 条实例为测试集。ECG 时间序列的示例如图 3.15 所示。这两类时间序列在整体上是相似的，所以很难用肉眼去区分它们。基于 ED 和 DTW 的 1-NN 分类器在此数据集上的分类准确率分别为 80.60%和 76.77%。然而，通过辨别性 shapelets 转换的 1-NN 分类器分别得到了 99.65%和 98.72%的分类准确率，明显优于在原始数据集上的基于欧式距离和 DTW 的 1-NN 分类器。所发现十个辨别性 shapelets 中类标为 *class1* 的三个如图 3.16 所示。从图 3.16 中可以观察到，虽然这三个 shapelets 具有很好的辨别性，但是它们之间仍存在着较多的重叠区域，怎样在保持分类准确率的同时移除掉相互重叠的辨别性 shapelets 是今后的主要研究方向之一。

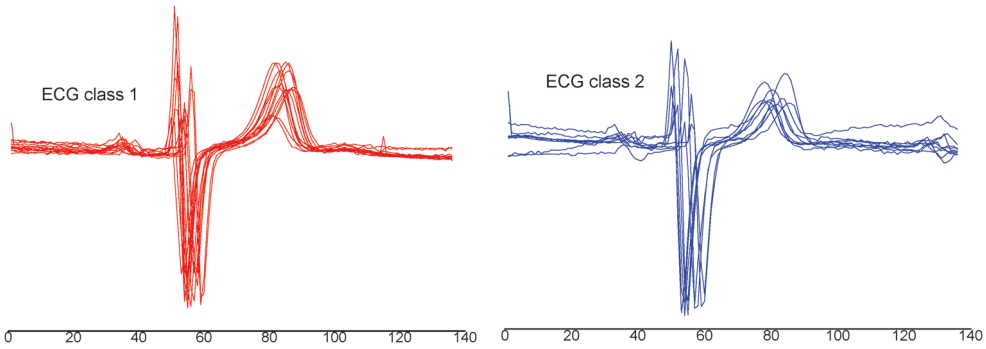

图 3.15 ECG Five Days 数据集中的两类时间序列

Figure 3.15 Two different classes for ECG Five Days dataset

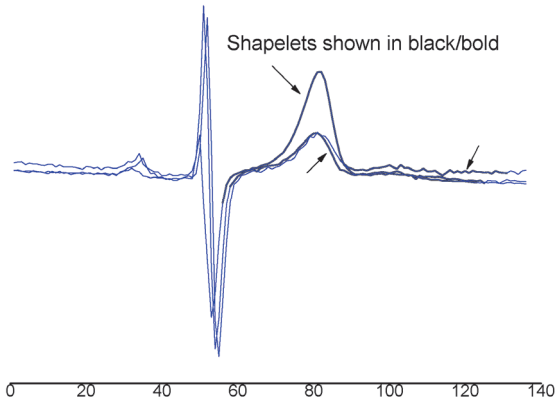

图 3.16 标记为 ECG *class1* 的辨别性 shapelets

Figure 3.16 Discriminative shapelts for ECG *class1*

3.4.4 Shapetlets 与数据表示

时间序列表示的主要目的是采用一种简洁的方式来强调数据的主要特征。许多高水准的时间序列表示方法已应用于数据挖掘中，包括离散傅里叶变换（DFT），离散小波变换（DWT），分段线性接近（PAA），自适应分段常数近似（APCA），奇异值分解（SVD），符号聚集近似（SAX）等等[16, 18, 22, 24, 84].

时间序列 shapelets 是时间序列中能够最大限度的表示一个类别的子序列。辨别性 shapelets 是由不相似的 shapelets 构成的，并能够保证分类的准确率。因为辨别性 shapelets 符合时间序列表示的基本性质，所以它可以当作一种新的时间序列表示形式。例如，从图 3.12，3.14 和 3.16 中可以发现，辨别性 shapelets 能够降低数据维度，强调数据集合的主要特征，并能够有效地处理噪音数据。

3.5 本章小结

本章介绍了 shapelet 剪枝和覆盖技术，提出了一种用于解决时间序列分类问题的辨别性 shapelets 转换方法，并通过实验和实例解析阐述了辨别性 shapelets 在时间序列分类方面的准确性与可解释性。尽管辨别性 shapelets 能够去除相似的 shapelets，辨别性 shapelets 间仍具有相互重叠的部分。今后的研究方向包括继续研究在保持分类准确率的同时移除掉相互重叠的辨别性 shapelets 的方法，并将辨别性 shapelets 应用到多变量时间序列分类和其他更多的实际应用问题中。

4 时间序列关联式分类算法

关联规则用于发现大量数据集中不同变量（或条目）之间的有趣模式，它通过使用有趣性评估（interestingness measure）方法，发现数据集中的强关联规则。本章首先介绍传统关联规则算法在时间序列应用方面的局限性，然后提出相应的解决方案和不同的规则评价方式。

4.1 引言

由于关联规则能够对知识进行简洁的、直观的描述，关联规则挖掘已引起了研究者广泛的关注。关联规则在分类方面的成功应用包括生物数据[85]，关系数据[83, 86-91]文本[92]以及图[93, 94]等等。高维的时间序列数据天然地需要强有力的分析工具来抽取出显著的和令人信服的规则，从而用其揭示序列模式和类标之间的重要关系，并将实值型的序列数据转换成相对容易理解的知识。但传统的关联规则挖掘算法主要集中于基于符号表示的事务数据上，为将关联规则挖掘算法应用于时间序列数据中，需要接受两方面的挑战。

首先，必须将时间序列数据离散化为片段，然后将每一个片段转换成一个符号。其次，尽管可以设置较高的支持度和置信度，但仍可能从高维时间序列数据中发现大量的无用的规则。

上述两个难题严重制约了关联规则在时间序列数据中的应用。针对第一个问题，本章采用了一种基于 SAX 的时间序列表示技术[24]，它能够在大幅度降低时间序列维度的同时保持时间序列数据的主要特征，并将原始数据离散化为符号序列。对于后一个问题，理想情况下，发现一条或者一小部分最显著的规则是优于产生大量的规则的，所以本章提出了一种懒惰式的关联式分类器，对于时间序列数据中的每一行，它都能发现最显著的一个或一系列类序列规则（CSR，Class Sequential Rule）。下面将用一个简单的例子进行阐述。

图 4.1 展示了经典的 *Gun/NoGun* 时间序列[49]和相应的 SAX 转换后的序列。初始的时间序列长度为 150，SAX 转换之后，高维的实值型序列被离散化为符号序列“*aabcddbbaa*”和“*bbbcddbbbb*”。在支持度和置信度框架下，对于 *Gun* 时间序列，懒惰式关联式分类器得到的最好规则为$\{A_0 = a, A_2 = b\} \rightarrow Gun$，如图 4.1（*a*）中红色标注部分所示。而符合 *NoGun* 时间序列的最好规则为$\{A_3 = c, A_7 = b\} \rightarrow NoGun$，如图 4.1（*b*）中红色标注部分所示。其中 $A_0, A_1, \ldots, A_9$ 分别表示转换后序列的属性，*a*，*b*，*c*，*d* 为对应的属性值。很明显，这两条规则充分展现了 *Gun*

和 *NoGun* 序列的主要不同点。

相比较于传统的时间序列分类方法如基于 ED 或 DTW 的 1-NN 分类器[37, 38]等，采用基于 SAX 表示的懒惰式时间序列关联式分类的优点主要在于以下几个方面。

（1）规则本身是直观的和可解释的。1-NN 分类器只能说明被分类对象与所分到的类的实例间具有较大的相似性，但不能指出各个类别之间具体的不同点。而基于 SAX 表示的关联式分类方法，可以较为充分地解释各类之间的区别，即具有较强的辨别性。图 4.1 中的例子很好的解释了这一点。

（2）类序列规则能够非常简洁紧凑地表示一个分类器。有时仅需要一条规则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。同时，由于规则的相对简洁性，它们更容易被研究者理解，并为探索未知类别的时间序列提供帮助。

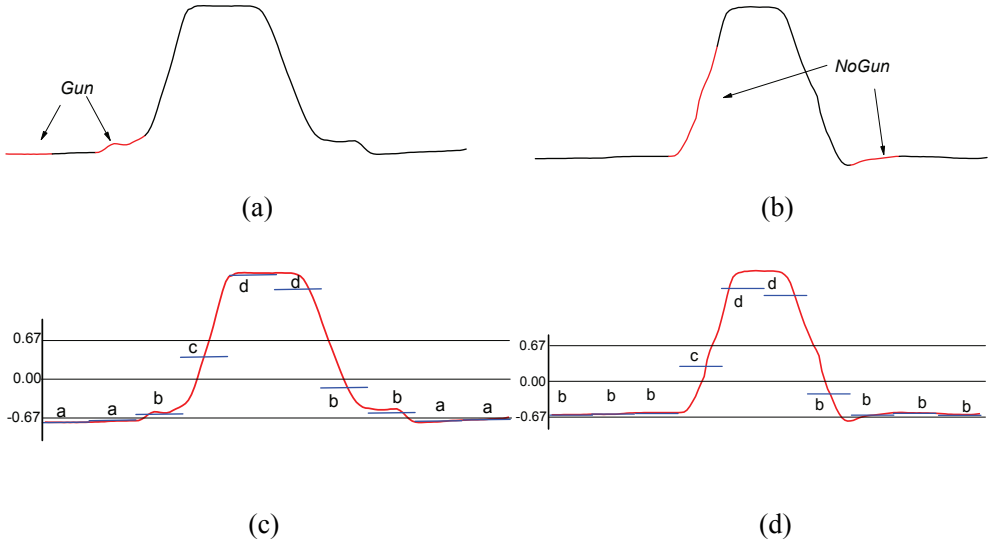

图 4.1 *Gun/NoGun* 时间序列的 SAX 表示和关联式分类器的可解释性；(a) *Gun* 时间序列和最好的 CSR；(b) *NoGun* 时间序列和最好的 CSR；(c) 与(a)中的 *Gun* 时间序列对应的 SAX 序列 *aabcddbbaa*；(d) 与(b)中的 *NoGun* 时间序列对应的 SAX 序列 *bbbcddbbbb*.

Figure 4.1 SAX representations for *Gun/NoGun* time series and the interpretability of associative classifier; (a) Time series *Gun* and the best CSR; (b) Time series *NoGun* and the best CSR; (c) The SAX words *aabcddbbaa* corresponding to time series *Gun* of (a); (d) The SAX words *bbbcddbbbb* corresponding to time series *NoGun* of (b).

（3）懒惰式的关联式分类器能够对时间序列中的每一行提供更完整的描述，并避免像其他基于 CBA（Classification Based on Associations）的算法[88]那样产生过量的计算。原因是懒惰式关联式分类器是基于需求驱动的，只有训练数据中有用的部分才被用于产生适用于测试实例的规则，所以它能够产生更多有用的规则。然

而，急切式分类器重点关注全局特征，许多根据全局搜索而产生的规则在分类过程中是无用的。4.4.4 节将给出详细的介绍。

使用关联式分类方法挖掘时间序列数据时，主要包括以下三个步骤。首先，将实值型高维数据离散化为低维的符号序列；其次，生成所有的类序列规则；最后，应用产生的类序列规则建造分类器并进行分类预测。

本章的贡献主要在以下几个方面。

（1）首次将关联式分类器应用于普遍的时间序列分类问题中，阐述了基于 SAX 表示的关联式分类器在时间序列数据上的可解释性。实验结果表明懒惰式的关联式分类器具良好的分类效果。

（2）本章首先提出了一种改进的 CBA 算法，用于发现类序列规则并分类预测，在此基础上，提出了一种懒惰式的关联式分类算法，避免产生过量的规则，并保证了规则对测试实例的覆盖。

（3）提出了四种不同的评价标准来选择用于分类的类序列规则。

本章的主要结构如下。第 4.2 节将总结相关的研究工作。第 4.3 节将阐述时间序列关联式分类问题相关定义与符号。第 4.4 节将详细阐述所提出的关联式分类方法，并给出四种不同的规则评价标准。第 4.5 节进行实验描述并评估所提出的算法。第 4.6 节将通过实例解析来说明关联式分类器的优点。第 4.7 节进行全文总结以及对进一步工作的展望。

4.2 相关工作

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于基于 ED、DTW 等不同距离度量方式的最近邻算法上[37, 38]。近年来，由于时间序列 shapelets[49]具有可解释性和分类准确性高等特点，基于时间序列 shapelets 的分类方法得到了广泛的应用[7, 49, 50]。然而，它们都未采用关联式分类技术。

Liu 等人[88]首次提出了类关联规则（CAR，Class Association Rule）的概念并通过产生所有的类关联规则建造分类器 CBA。对每一条测试实例，此算法选取与之相符的具有最大置信度的类关联规则进行分类，但它有可能产生过量的规则，并且不能覆盖所有测试实例。CMAR 算法[87]采用加权的卡方检验来选取多条规则进行分类，而 CPAR 算法[91]采用贪心策略直接从训练数据中产生规则。上述的三个方法均致力于挖掘高支持度，高置信度的规则来建立基于规则的分类器。

HARMONY 算法[90]采用了一种以实例为中心的规则产生方法，用以保证对于每一个训练实例，能够覆盖此实例的置信度最高的规则中的一个被包含在分类器中。Veloso 等人[89]也提出了一种懒惰式的关联式分类器，与之相比，本章的工作

有以下几个不同点。首先，本章所提出算法重点研究了时间序列数据的处理方法，而前者只能处理事务数据；其次，时间序列数据是实值型的有次序的数据，必须先离散化时间序列然后将它们转换成符号序列，同时还得考虑属性间的次序关系。换句话说，本章所提出算法用于发现类序列规则，而 Veloso 提出的方法发现的是类关联规则；最后，本章将展示了关联式分类器在时间序列数据集上的可解释性。

Cheng 等[83]最近提出了一种基于频繁模式的分类算法，作者采用有辨别性的频繁模式来表示数据集，并在预测过程中采用分类模型进行分类，而不是使用一个或多个排名靠前的规则进行分类。Cheng 等[86]又提出了此算法的改进算法，并采用分支限界法来直接搜索有辨别性的频繁模式，而不需要生成完全的模式集。关联规则发现也被应用于分析生物信息数据。比如针对高维基因谱中的每一行，研究者提出使用 top-k 覆盖规则建立一个 RCBT 分类器[85]。

在文献[95]中，作者采用了序列的和非序列的关联规则挖掘方法来处理一种基于模式的股票数据，它可以认为是时间序列数据挖掘的一部分，然而此方法只能处理特定的股票序列，不能应用于普遍的序列分类问题，并缺乏可解释性。Das 等[96]通过聚类和离散化时间序列的子序列得到符号序列，并应用简单的规则发现方法来获取序列中的规则，但是此算法缺乏可解释性并且需要设置过多的参数。

4.3 定义与符号

除时间序列数据的基本符号之外，文中涉及的其他符号如表 4.1 所示。

表 4.1 符号表

Table 4.1 Symbol table

符号	释义
$\boldsymbol{CSR}$	类序列规则的集合
csr_i	一条类序列规则
$\boldsymbol{x}^a$	类序列规则的前件
y	类序列规则的后件
$E(\mathbf{X})$	规则的熵
$I(csr_i)$	规则的信息增益
c	SAX 表示基数的个数
l	SAX 表示符号的长度

在关联式分类器中，SAX 表示方法首先将时间序列分割成片段，这里每一个片段都可以当作是时间序列的子序列。而转换后的符号，如图 4.1（c）和（d）中的 a，b，c，d，表示了子序列的主要特征。

定义 4.1：类序列规则

在进行关联规则挖掘时，发现的目标是不能事先定义的，而分类规则挖掘有且

仅有一个预先设定的目标，若将这两种挖掘技术结合起来，就可以用于挖掘 CAR，CAR 是关联规则的一个特殊的子集[2]。若此 CAR 是有序的，则称之为类序列规则（CSR，Class Sequential Rule）。

一个类序列规则 csr_i 是形如 $\boldsymbol{x}^a \rightarrow y$ 的规则，其中序列 $\boldsymbol{x}^a$ 为 $\boldsymbol{x}=<x_1, x_2, \ldots, x_T>$的非空子集，$y$ 为一个类值（$y \in \boldsymbol{Y}$）。需要注意的是，类序列规则的后件只有一个类属性，前件可以有多个属性，但前件中的属性不能为类属性且必须是有序的。

类序列规则可以通过改进关联规则挖掘算法而产生。即每一个项集必须包含一个类值并且产生规则时，后件必须是类属性，前件中的每一个属性必须是有序的。

当生成所有类序列规则之后，为分类转换后的时间序列，必须为所有生成的类序列规则排序。本章采用信息增益和置信度来度量生成的规则，相关的定义如下。

定义 4.2：类序列规则的支持度与置信度

一个类序列规则 csr_i 的支持度为时间序列数据库 $\boldsymbol{X}$ 中既包含 $\boldsymbol{x}^a$ 又包含 y 的时间序列的个数除以整个序列数据中时间序列的个数$|\boldsymbol{X}|$，csr_i 的置信度为 $\boldsymbol{x}^a$ 和 y 的支持度除以 $\boldsymbol{x}^a$ 的支持度。公式如下所示。

$$Support(x^a \rightarrow y) = \frac{count(x^a \cup y)}{|X|} = P(x^a \cup y) \tag{4.1}$$

$$Confidence(x^a \rightarrow y) = \frac{count(x^a \cup y)}{count(x^a)} = P(y \mid x^a) \tag{4.2}$$

定义 4.3：熵与信息增益

假设数据集 $\boldsymbol{X}$ 中有 N 个时间序列，类值的个数为$|\boldsymbol{Y}|$，每一个类值 y_i 都有 n_i 个标定的实例，其中 $n_1 + n_2 + \ldots + n_{|Y|} = N$。数据集 $\boldsymbol{X}$ 的熵定义为

$$E(X) = -\sum_{i=1}^{|Y|} \frac{n_i}{N} \log \frac{n_i}{N} \tag{4.3}$$

一个类序列规则 csr_i 将数据集 $\boldsymbol{X}$ 分割成互不相交的两个子集，$\boldsymbol{X}_m$ 表示数据集中满足 csr_i 的前件的所有实例，$\boldsymbol{X}_{um}$ 表示其他的实例，并且 $N_1=|\boldsymbol{X}_m|, N_2=|\boldsymbol{X}_{um}|$，那么 csr_i 的信息增益为

$$I(csr_i) = E(X) - \frac{N_1}{N} E(X_m) - \frac{N_2}{N} E(X_{um}) \tag{4.4}$$

4.4 关联式分类

此部分将详细介绍本章提出的算法。首先将在 4.4.1 节阐述时间序列关联式分类的主要思想；其次，四种不同的评价类序列规则的方法将会在 4.4.2 节进行阐述；在 4.4.3 节，我们将改进 CBA 算法，用于发现类序列规则；4.4.4 节将给出懒惰式关联分类的具体实现细节。

4.4.1 算法概述

本章应用了一种基于转换表示的方法来解决时间序列分类问题。通过这种方法，我们可以将初始的实值型的高维数据转换成离散的低维数据。显然，在低维的离散型的数据上进行搜索是更有效的；更重要的是，我们可以更加容易的从转换后的数据中发现类序列规则，进行关联式分类与预测。

（1）SAX 表示

众所周知，时间序列数据是难以掌控的，但当把它们当做符号序列而不是实值型的数据时，从中挖掘和发现有趣的模式或关联规则将变得更加容易。由于类序列规则从本质上说是一种符号表示，因此必须将实值型的高维数据转换为一种更加抽象的符号表示数据。本章采用了由 Lin 等人[24]提出的 SAX 表示方法（图 4.1 所示）。SAX 表示方法分为两个步骤，首先采用 PAA 表示时间序列（图 1.4），每个子序列的长度为 l。由于规范化后的时间序列的值呈高斯分布，可根据 SAX 基数的个数 c 和高斯函数的性质来定义分裂点，从而使得每一部分的概率相同，同时将时间序列转换成符号序列（图 1.5，4.1 所示）。

另一个问题存在于从符号序列归纳规则的过程中。一般情况下，从抽象的符号中抽取规则的质量与符号的抽象程度之间存在一个折衷，所以采用 SAX 表示时需要设置的基数（即符号的个数）c 和每个符号表示的子序列的长度 l，都影响着归纳出的规则的好坏。为保持简洁性和一致性，在本章的例子中，将基数设置为 4，每一个符号所能表示的长度设置为 15。如图 4.1 所示，通过使用 SAX 表示技术，将图 1.1 中所示的 *Gun/NoGun* 序列转换为符号序列“*aabcddbbaa*”与“*bbbcddbbbb*”。参数对实验结果的影响将会在 4.4.2 节进行阐述。

（2）发现类序列规则

发现类序列规则主要分为两个步骤。首先，使用 SAX 技术将实值型高维数据离散化为低维的符号序列；其次，根据离散化后的符号序列，生成所有的类序列规则。类序列规则可以通过改进已有的关联规则挖掘算法而产生。即每一个项集必须包含一个类属性并且产生的规则的后件必须为类属性，更为重要的是，每一个属性的次序必须反映在规则中。本章首先通过改进 Liu 等提出的用于发现类关联规则的 CBA 算法[88]来发现类序列规则，然后提出了一种懒惰式的关联式分类器来弥补改进的 CBA 算法的不足。

（3）分类预测

当发现所有类序列规则之后，需要根据所有的或者一部分类序列规则构建分类器。为建立更好的分类器，首先需要对规则进行排序，具体的规则排序方法如 4.4.2 节所示。假设 $\boldsymbol{CSR}$ 是算法生成的全部类序列规则的集合，$\boldsymbol{X}_{test}$ 为时间序列数

据的测试集。本章采用的分类算法的基本思想即为从 ***CSR*** 中抽取出一个或一组能够覆盖 $\boldsymbol{X}_{test}$ 中实例的规则进行分类预测。

4.4.2 规则评价标准

当类序列规则生成之后，它们被用于分类 SAX 转换后的时间序列测试集。但仍然需要解决几个问题。比如，哪个规则或者哪些规则将被用于分类？或者说选择规则的标准是什么？如果被选择的多个规则的后件是不同的，应该采纳哪一个等等？

为选择适当的类序列规则用于分类，首先需要排序所有的类序列规则。本章采用了四种不同的评价标准，即最大置信度（Best Confidence，BC），最大置信度和（Maximum Sum of Confidences，MSCs），最大信息增益（Best Information Gain，BIG）与最大信息增益和（Maximum Sum of Information Gains，MSIGs）。前两者根据置信度排序，而后两者是根据信息增益排序的。这两种排序规则的定义如下。

定义 4.4：置信度排序

给定两个类序列规则，csr_i 和 csr_j，如果 $csr_i > csr_j$ 或者说 csr_i 优于 csr_j，那么

（1）csr_i 的置信度大于 csr_j 的置信度，或者

（2）它们的置信度是相同的，但是 csr_i 的支持度大于 csr_j 的支持度，或者

（3）它们的置信度和支持度都相同，但是 csr_i 生成的更早或者比 csr_j 的属性更少。

需要注意的是，Lin 等人[88]定义的排序规则与上述的置信度排序有一定的相似之处，但它定义的是类关联规则的顺序，而此处定义的是类序列规则的顺序。

定义 4.5：信息增益排序

给定两个类序列规则，csr_i 和 csr_j，如果 $csr_i > csr_j$ 或者说 csr_i 优于 csr_j，那么

（1）csr_i 的信息增益大于 csr_j 的信息增益，或者

（2）它们的信息增益是相同的，但是 csr_i 的置信度大于 csr_j 的置信度，或者

（3）它们的信息增益和置信度都相同，但是 csr_i 的支持度大于 csr_j 的支持度，或者

（4）它们的信息增益、置信度和支持度都相同，但是 csr_i 生成的更早或者比 csr_j 的属性更少。

（1）最大置信度

对于最大置信度评价标准来说，所有满足测试实例的类序列规则中置信度最大的将被用于分类预测。一般来说，置信度越高，分类准确性越高。比如对于 *Gun/NoGun* 问题，如表 4.2 所示，表中的四条类序列规则均匹配测试实例 $\{A_0 = a,$

$A_1=a, A_2=a, A_3=d, A_4=d, A_5=d, A_6=c, A_7=b, A_8=a, A_9=a\}$，而我们将选择第一条规则 $R1$，因为它具有最大的置信度。虽然 $R1$ 和 $R2$ 的置信度相同，但根据排序规则，$R1$ 中的属性更少而且生成时间更早，所以最终的选择是 $R1$。

表 4.2 最大置信度

Table 4.2 Best Confidence

规则次序	类序列规则	置信度
$R1$	$\{A_7=b，A_8=a\}\rightarrow Gun$	94%
$R2$	$\{A_4=d，A_7=b，A_8=a\}\rightarrow Gun$	94%
$R3$	$\{A_5=d\}\rightarrow NoGun$	50%
$R4$	$\{A_4=d，A_9=a\}\rightarrow NoGun$	45%

表 4.3 最大置信度和

Table 4.3 Maximum Sum of Confidences

规则次序	类序列规则	置信度
$R1$	$\{A_7=b，A_8=a\}\rightarrow Gun$	94%
$R2$	$\{A_4=d，A_7=b，A_8=a\}\rightarrow Gun$	94%
$R3$	$\{A_1=b，A_4=d\}\rightarrow NoGun$	82%
$R4$	$\{A_1=b，A_5=d\}\rightarrow NoGun$	82%
$R5$	$\{A_5=d\}\rightarrow NoGun$	50%
$R6$	$\{A_4=d，A_9=a\}\rightarrow NoGun$	45%

（2）最大置信度和

直观上，关联式分类允许多个类序列规则覆盖同一训练或测试实例，所以选择适当的类序列规则是关联式分类中的一个疑难点。其中一个比较合理的方法是将所有满足测试实例的类序列规则聚集起来，并选择最优的那一类集合。本章考虑采用最大置信度和的方式。HARMONEY[90]算法也采用了类似的方法来选择类关联规则。如表 4.3 所示，表中的六条类序列规则全部匹配测试实例$\{A_0=a, A_1=b, A_2=a, A_3=d, A_4=d, A_5=d, A_6=c, A_7=b, A_8=a, A_9=a\}$。若根据最大置信度，此处将选择第一条规则 $R1$。但是根据最大置信度和，该测试实例被分类为 *Gun* 的置信度和为 $0.94+0.94=1.88$，被分类为 *NoGun* 的置信度和为 $0.82+0.82+0.50+0.45=2.59>1.88$，所以该测试实例将被分类为 *NoGun*。

（3）最大信息增益

众所周知，在决策树分类器如 ID3，C4.5[97, 98]中，决策树的构造是通过不断地循环选择信息增益最大的分裂节点完成的。而决策树可以看做是由一系列不重叠的决策规则组成的。每一条规则的信息增益等于决策树中对应路径的信息增益。类序列规则可以认为是一种特殊的决策规则[89]，它的信息增益可以按照计算决策规则信息增益的方式获得（如定义 4.3 所示）。一般情况下，一条规则的信息增益越大，此规则的辨别力越高。如表 4.4 所示，测试实例仍然为$\{A_0=a, A_1=a, A_2=a, A_3$

$= d, A_4 = d, A_5 = d, A_6 = c, A_7 = b, A_8 = a, A_9 = a\}$，根据最大信息增益，最终选择的类序列规则为 $R1$。从表中仍可以看到，尽管 $R3$ 的置信度小于 $R4$，但由于 $R3$ 的信息增益大于 $R4$，所以 $R3$ 优于 $R4$。

表 4.4 最大信息增益

Table 4.4 Best Information Gain

规则次序	类序列规则	置信度	信息增益
$R1$	$\{A_7=b，A_8=a\}\rightarrow Gun$	94%	0.36
$R2$	$\{A_4=d，A_7=b，A_8=a\}\rightarrow Gun$	94%	0.36
$R3$	$\{A_4=d，A_9=a\}\rightarrow NoGun$	45%	0.05
$R4$	$\{A_5=d\}\rightarrow NoGun$	50%	0.04

（4）最大信息增益和

同样，当选择信息增益来评价类序列规则的优劣时，仍有可能出现同一条测试实例被多条类序列规则覆盖的情况。为构造出更好的分类器，本章同时也考虑采用最大信息增益和作为规则的选择标准。如表 4.5 所示，测试实例$\{A_0 = a, A_1 = b, A_2 = a, A_3 = d, A_4 = d, A_5 = d, A_6 = c, A_7 = b, A_8 = a, A_9 = a\}$满足全部的六条类序列规则。若根据最大置信度和，该测试实例将被分类为 *NoGun*。但若根据最大信息增益和评价标准，该实例被分类为 *Gun* 的信息增益和为 $0.36 + 0.36 = 0.72$，被分类为 *NoGun* 的信息增益和为 $0.15 + 0.15 + 0.05 + 0.04 = 0.39 < 0.72$，所以该测试实例将被分类为 *Gun*。

表 4.5 最大信息增益和

Table 4.5 Maximum Sum of Information Gains

规则次序	类序列规则	置信度	信息增益
$R1$	$\{A_7=b，A_8=a\}\rightarrow Gun$	94%	0.36
$R2$	$\{A_4=d，A_7=b，A_8=a\}\rightarrow Gun$	94%	0.36
$R3$	$\{A_1=b，A_4=d\}\rightarrow NoGun$	82%	0.15
$R4$	$\{A_1=b，A_5=d\}\rightarrow NoGun$	82%	0.15
$R5$	$\{A_4=d，A_9=a\}\rightarrow NoGun$	45%	0.05
$R6$	$\{A_5=d\}\rightarrow NoGun$	50%	0.04

4.4.3 急切关联式分类算法 SCBA

此部分将提出一种改进的 CBA 算法，并命名为 SCBA（Sequence Classification Based on Associations）算法，用于发现类序列规则并分类。两者之间的主要区别在于，SCBA 在原有算法的基础上增加了序列的概念，用于保证得到的类关联规则的前件是有序的。另外，SAX 转换之后，时间序列数据集中每一个属性的属性值得取值范围是相同的。如在图 1.1 中，类序列规则 $abd \rightarrow Gun$ 可能表示多条规则，它可以表示$\{A_0 = a, A_2 = b, A_4 = d\} \rightarrow Gun$，也可以表示$\{A_0 = a, A_2 = b, A_5 = d\} \rightarrow Gun$等，所以 SCBA 算法必须严格记录每一个属性的次序和与之相对应的属性值。显

然，SCBA 算法是一个急切式的关联式分类算法。

```
算法: EagerAssoClassifier(X_train, X_test, sup, conf)
输入: 训练集 X_train, 测试集 X_test, 最小支持度 sup, 最小置信度 conf.
输出: 分类准确率 Accuracy.
 1: X' = TranstoSAX(X_train);
 2: X'' = TranstoSAX(X_test);
 3: Csr = FindallCSRs(X', sup, conf);
 4: Csr' = sort(Csr);
 5: rule = ∅, class = ∅, Accuracy = 0;
 6: for i = 0 to | X''| do {每一个测试实例}
 7:   rule = PickCSR(Csr', x''_i);
 8:   if (rule = ∅)
 9:     class[i] = DefaultClass;
10:   else
11:     class[i] = predict(rule);
12: end for
13: Accuracy = Compare(class, X'');
14: return Accuracy;
```

图 4.2 急切式关联式分类器 SCBA

Figure 4.2 Eager associative classifier SCBA

图 4.2 给出了此算法的基本步骤。如伪代码的 1-2 行所示，初始时，训练集和测试集都通过 SAX 转换成符号序列。然后，算法根据最小支持度和最小置信度挖掘所有的类序列规则并根据置信度或者信息增益的降序排序所有规则（如 3-4 行所示）。对于每一条转换后的测试实例 $\boldsymbol{x}''_i$，若规则的评价标准为最大置信度（BC）或最大信息增益（BIG），匹配实例 $\boldsymbol{x}''_i$ 的第一条类序列规则将被用于预测类值；否则，挑选出所有匹配 $\boldsymbol{x}''_i$ 的类序列规则并选择置信度和或信息增益和最大的那一组规则进行分类。若没有规则能够匹配 $\boldsymbol{x}''_i$，那么 $\boldsymbol{x}''_i$ 将被赋予训练集中出现次数最多的类值 *DefaultClass*（如 6-12 行所示）。最后，此算法返回分类的准确率 *Accuracy*（10-11 行）。需要注意的是，在测试阶段，关联式分类器仅仅是检查每一个类序列规则是否与相应的测试实例相匹配，最终将选择与第一个匹配的类序列规则相关联的类值或者与某一组类序列规则相关联的类值进行分类。

急切关联式分类器能够产生大量的类序列规则，但是，它们中的大部分并不能用于分类。比如，对于 *Gun/NoGun* 问题，当最小支持度为 40%，最小置信度为 60% 时，通过急切关联式分类器得到的已排序的类序列规则有 8 条，分别为：

（1）$\{A_1 = a, A_5 = d\} \rightarrow Gun$

（2）$\{A_0 = a, A_1 = a, A_5 = d\} \rightarrow Gun$

（3）$\{A_1 = a, A_4 = d, A_5 = d\} \rightarrow Gun$

（4）$\{A_0 = a, A_1 = a, A_4 = d, A_5 = d\} \rightarrow Gun$

（5）$\{A_1 = a, A_4 = d\} \rightarrow Gun$

（6）$\{A_0 = a, A_1 = a, A_4 = d\} \rightarrow Gun$

（7）$\{A_1 = a\} \rightarrow Gun$

（8）$\{A_0 = a, A_1 = a\} \rightarrow Gun$

对图 4.1（*d*）中所示的测试实例$\{A_0 = b, A_1 = b, A_2 = b, A_3 = c, A_4 = d, A_5 = d, A_6 = b, A_7 = b, A_8 = b, A_9 = b\}$而言，没有一条类序列规则能够与它相匹配，只能给定它一个默认的分类值，此默认值为所有训练实例中出现次数最多的类值。这种情况在 SCBA 算法中经常出现，而没有任何证据证明根据默认值进行分类是令人信服的。换句话说，急切式分类器所产生的规则在这种情况下是无效的。接下来将阐述懒惰式分类器的设计思想并改进急切式分类器的缺点。

4.4.4 懒惰关联式分类算法

本部分将介绍一种新的基于需求驱动的懒惰式关联式分类器。它与急切式关联式分类器间的主要不同在于，急切式关联式分类器在不知道测试实例的情况下，直接从训练集中提取出所有的类序列规则，而懒惰式关联式分类器分别针对测试集中的每一个测试实例归纳相应的类序列规则。图 4.3 阐述了该算法的基本步骤。同急切关联式分类器一样，算法的第一步需要将训练集和测试集都转换成符号序列（如 1-2 行所示）。然后，针对每一条转换后的测试实例 $\boldsymbol{x}''_i$，对训练集 $\boldsymbol{X}'$进行映射。需要注意的是，映射后的数据集 D_i 是由训练集 $\boldsymbol{X}'$中与测试实例 $\boldsymbol{x}''_i$ 至少有一个相同的特征的实例所组成的。表 4.6 展示了图 4.1（*d*）中的测试实例在 *Gun/NoGun* 数据的训练集上的映射情况。所有类序列规则都是根据映射后的训练集 $\boldsymbol{D}_i$ 产生并按照置信度或信息增益排序的，最终选取最好的一个或一组类序列规则进行分类（如 4-10 行所示）。由于所有的类序列规则是根据 $\boldsymbol{D}_i$ 获得，并且 $\boldsymbol{D}_i$ 只包含与测试实例 $\boldsymbol{x}''_i$ 相关的特征，所以所有生成的类序列规则 *Csr'*（第 7 行所示）都与 $\boldsymbol{x}''_i$ 相匹配。针对同一条测试实例$\{A_0 = b, A_1 = b, A_2 = b, A_3 = c, A_4 = d, A_5 = d, A_6 = b, A_7 = b, A_8 = b, A_9 = b\}$，当最小支持度为 40%时，懒惰关联式分类器所产生的类序列规则有 6 条（已根据置信度排序），分别为：

（1）$\{A_4=d\} \rightarrow NoGun$

（2）$\{A_5=d\} \rightarrow Gun$

（3）$\{A_5=d\} \rightarrow NoGun$

（4）$\{A_4=d，A_5=d\} \rightarrow Gun$

（5）$\{A_4=d，A_5=d\}\rightarrow NoGun$

（6）$\{A_4=d\}\rightarrow Gun$

每一条规则都与测试实例相匹配，根据最大置信度方法，将选取第一条规则将此实例分类为 *NoGun*。

```
算法: LazyAssoClassifier(X_train, X_test, sup)
输入:训练集 X_train, 测试集 X_test, 最小支持度 sup.
输出: 分类准确率 Accuracy.
 1: X' = TranstoSAX(X_train);
 2: X'' = TranstoSAX(X_test);
 3: rule = ∅, class = ∅, Accuracy = 0;
 4: for i = 0 to | X''| do {每一个测试实例}
 5:   D_i = Project(x''_i, X');
 6:   Csr = FindallCSRs(D_i, sup);
 7:   Csr' = sort(Csr);
 8:   rule = PickCSR(Csr', x''_i);
 9:   class[i] = predict(rule);
10: end for
11: Accuracy = Compare(class, X'');
```

图 4.3 懒惰式时间序列关联式分类算法

Figure 4.3 The lazy associative classifier for time series

懒惰式分类器优于急切式的 SCBA 算法的原因主要在两个方面：

（1）SCBA 算法从训练数据的所有特征中挖掘类序列规则，一些重要的类序列规则可能由于在训练集中不是频繁的而丢失。而由于映射后的训练集是初始训练集的子集，那些在训练集中被忽略的重要规则在映射后的数据集中可能是频繁的，所以懒惰式分类器有利于产生更多有用的规则。

（2）懒惰式的关联式分类器是根据测试实例的情况来产生训练实例的，而 SCBA 算法在不知道测试实例的情况下产生了所有的类序列规则。所以懒惰式分类器能够避免产生过量的不能覆盖测试实例的规则。

另外，懒惰式关联式分类器不需要设置最小置信度的阈值，因为懒惰式分类器产生的所有规则对测试实例来说都是有用的，所以仅仅需要设置最小支持度的阈值（如图 4.2 所示）。由于用户很难确定一个合适的置信度阈值，所以这种提升是有益的。对于 SCBA 算法来说，设置置信度太大时有可能找不到类序列规则，而设置过低时有可能降低重要规则的影响力并产生许多无用的规则，算法的运行时间也会随之增加。另外，由于懒惰式分类器需要为每一条序列数据建立分类器，此举会消耗大量的运算时间，但由于根据测试实例映射训练集之后，相同的类序列规

则拥有相同的置信度和信息增益，顾可以考虑采用缓存技术（比如存储最常用的前1000 条类序列规则，避免了对规则置信度和信息增益的重复计算）来加快类序列规则的发现过程。通过实验发现，采用缓存技术之后，懒惰式的关联分类器有可能比急切式分类器的分类速度更快。

表 4.6 根据测试实例映射训练集

Table 4.6 Projected training data

Gun/NoGun	A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9
NoGun	*b*	*b*	*b*	-	*d*	*d*	-	*b*	*b*	*b*
NoGun	*b*	*b*	*b*	-	*d*	*d*	-	-	*b*	*b*
Gun	-	-	-	*c*	*d*	*d*	-	*b*	*b*	-
Gun	-	-	*b*	*c*	*d*	*d*	-	*b*	-	-
NoGun	*b*	*b*	*b*	-	*d*	*d*	*b*	*b*	*b*	*b*
NoGun	*b*	*b*	*b*	-	*d*	*d*	*b*	-	-	-
NoGun	-	-	-	-	*d*	*d*	-	-	-	-
NoGun	-	-	-	*c*	-	-	-	-	-	-
...										
Gun	-	-	*b*	*c*	*d*	*d*	*b*	*b*	*b*	*b*
NoGun	-	-	-	-	*d*	*d*	-	-	-	-
?	*b*	*b*	*b*	*c*	*d*	*d*	*b*	*b*	*b*	*b*

4.5 实验与评价

本部分将对本章所提出的基于 SAX 的时间序列关联式分类算法进行测试。所采用数据集总结于表 4.7。4.5.1 节的可扩展性实验是为了测试本章所提出的算法在不同最小支持度或不同大小数据集上的表现。4.5.2 节阐述了 SAX 转换时，参数设置对分类准确率的影响。由于 1-NN 分类器是当前处理时间序列分类问题的最准确的和最健壮的分类器，此处选择它当作 4.5.1 和 4.5.2 节的对比基准。最后将所提出的算法与 1-NN 分类器和 FastShapelet 算法[50]做对比实验，验证其分类的准确性。所采用的数据集由 Ding 等[37]以及 Keogh 等所提供。

4.5.1 可扩展性实验

为测试算法的可扩展性，本章在多个数据集上进行了实验。对时间序列关联式分类算法而言，有两个主要的因素影响着算法的效率，时间序列数据集的大小和最小支持度的大小，下面将单独测试每一个因素对分类速度的影响。

（1）时间序列数据集的大小不变，最小支持度不断减小

如图 4.4 对比了不同最小支持度下，急切关联式分类器 SCBA、懒惰关联式分类器以及 1-NN 分类器在 *Gun/NoGun* 数据集上的运行时间。为避免 SAX 转换时参数设置的影响，我们统一将基数 c 设置为 4，每个符号所能代表的长度 l 设置为 15。

关联式分类器分别采用了四种不同的类序列规则评价方式，对于急切式关联式分类器 SCBA 来说，四种算法分别命名为 eBest，eAll，eInfo 和 eInfoA，它们分别与四个规则评价标准，最大置信度、最大置信度和、最大信息增益、最大信息增益和相对应；同样，懒惰式关联式分类器的四种算法分别命名为 laBest，laAll，laInfo 和 laInfoA。

表 4.7 数据集

Table 4.7 Summary of datasets

Data	Assessment	Instances (Train/Test)	Length	Number of classes
CBF	Train/Test	30/900	128	3
Coffee	Train/Test	28/28	286	2
ECGFiveDays	Train/Test	23/861	136	2
Cricket	Train/Test	9/98	308	2
Cricket_new	Train/Test	9/64	308	2
DiatomSizeReduction	Train/Test	16/306	345	4
FaceAll	Train/Test	560/1690	131	14
Gun_Point	Train/Test	50/150	150	2
ItalyPowerDemand	Train/Test	67/1029	24	2
Lighting7	Train/Test	70/73	319	7
Lighting2	Train/Test	60/61	637	2
MoteStrain	Train/Test	20/1252	84	2
SonyAIBORobotSurfaceII	Train/Test	27/953	65	2
SonyAIBORobotSurface	Train/Test	20/601	70	2
Synthetic_Control	Train/Test	300/300	60	6
Wafer	Train/Test	1000/6164	152	2
Haptics	Train/Test	155/308	1092	5

图 4.4 中所有数据均是通过 10 次测量求平均值的结果，所有的分类器均是根据训练集建立并在测试集上测试的。从图中可以观察到，当最小支持度不断减小时，由于 1-NN 算法的具体实现与最小支持度无关，它的运行效率保持不变。对 SCBA 算法而言，当最小支持度比较大时，算法的运行时间优于 1-NN 算法，但随着最小支持度的不断减小，发现的频繁序列模式和类序列规则不断增多，SCBA 算法的运行时间逐渐超过了 1-NN 算法的运行时间。对懒惰式关联式分类算法而言，当最小支持度比较大时，由于映射后的实例中满足最小支持度的特征比较少，该类算法的运行时间与 1-NN 相差无几，但随着最小支持度的不断减小，满足最小支持度的实例特征不断增多，懒惰式分类器所消耗的时间有了明显的增长。

另外还可以观察到，从运行效率上说，急切式的 SCBA 分类器在整体上优于懒惰式分类器。在急切式分类器的内部比较中，eBest>eAll，eInfo>eInfoA，eBest>eInfo，eAll>eInfoA；同样，在懒惰式分类器的内部比较中，laBest>laAll，laInfo>laInfoA，laBest>laInfo，laAll>eInfoA。究其原因，首先，根据最大置信度和或者最大信息增益和进行分类时，需要考虑所有产生的规则，而根据最大置信度或者最大信息增益进行分类时只需要找到第一条相匹配的规则，所以在运行效率方

面后者普遍优于前者；其次，根据信息增益排序时，需要事先计算规则的置信度，然后才计算其信息增益，所以根据信息增益来选取规则的算法的运行效率普遍低于根据置信度来选择规则的方法；最后，由于懒惰式分类器需要针对每一条测试实例建立相应的分类器，而急切式分类器只需要建立一次分类器，所以一般情况下后者的运行效率优于前者。

（2）最小支持度不变，时间序列数据集不断增大

为测试最小支持度不变，时间序列数据集不断增长时时间序列关联式分类算法的运行效率，此处将 wafer 数据集分割，用于获取不断增长的数据集。wafer 中有 1000 条实例，每条实例有 153 个属性（包含类属性），首先让第一个数据集包含的实例个数为 50 条，然后每增加 50 条实例就创建一个新的数据集，这样就产生了 20 个不断增长的数据集。为保持简洁性和统一性，将最小支持度设置为 0.1，算法运行时间为 10 折交叉验证后得到的平均时间。

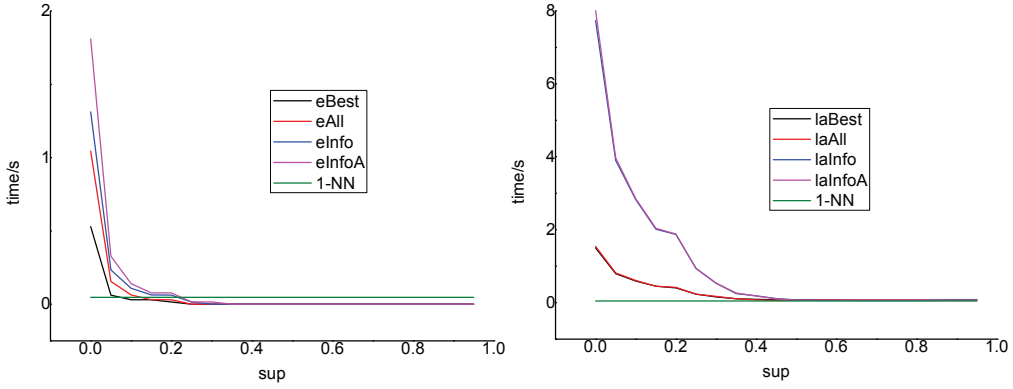

图 4.4 (a) SCBA 算法在不同支持度下的运行时间；(b) 懒惰式关联式分类器在不同支持度下的运行时间.

Figure 4.4 (a) Execution time for SCBA under different supports; (b) Execution time for lazy associative classifiers under different supports.

如图 4.5 所示，当数据集不断增长时，急切式和懒惰式分类器的运行时间都基本呈线性增长趋势。当数据集不断增大时，laInfo 和 laInfoA 算法的增长速率比较明显，原因是随着数据集不断增长，训练集中与测试实例具有相同特征的实例个数不断增多，懒惰式算法所产生的频繁序列模式和类序列规则也相应的增多，从而运行时间出现了明显的增长。

4.5.2 参数影响测试

由于关联规则能够对知识进行简洁在对时间序列数据进行 SAX 转换时需要设置两个参数，基数（即符号的个数）c 和每一个符号所表示的时间序列子序列的长

度 *l*。为避免最小支持度的设置对此次实验的影响，统一将最小支持度设置为 1/*N*, 其中 *N* 为数据集中的实例的个数。下面将逐个测试每一个参数对关联式分类器分类准确率的影响。所有的分类器均是根据训练集建立并在测试集上测试的。

（1）*l* 不变，改变 *c* 的大小

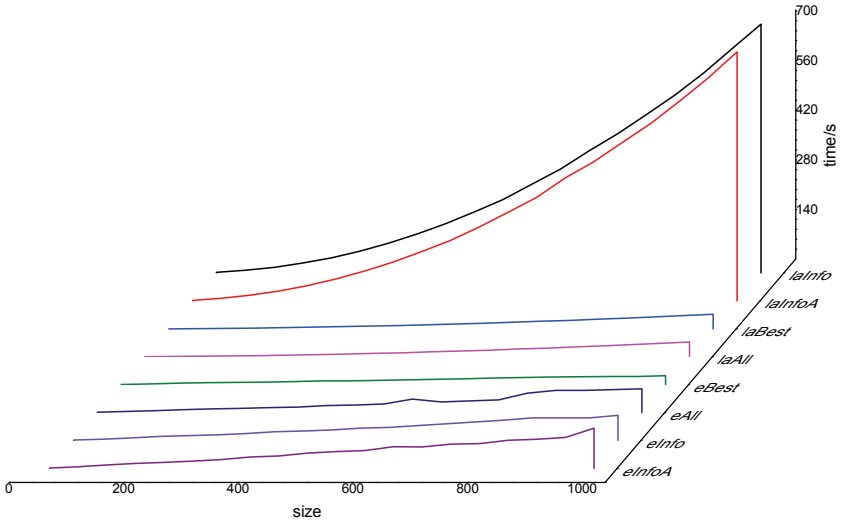

图 4.5 最小支持度不变数据集的大小不断增大时算法的运行时间

Figure 4.5 Comparisons of execution time between eager and lazy classifiers when the size of dataset grows and the minimum support remains unchanged.

为测试 *l* 不变，*c* 变化时对分类准确率的影响，针对 *Gun/NoGun* 数据集，设置 *l* 为 15，*c* 的变化范围为 3 到 10。如图 4.6 所示，图中给出了 *c* 从 3 到 10 变化时急切式关联式 SCBA 分类器、懒惰式关联式分类器以及 1-NN 分类器在 *Gun/NoGun* 数据集上的准确率变化曲线。

从图 4.6 中可以看到，当 *c* 在 3 到 10 范围内不断变化时，各个关联式分类器的准确率变化没有固定的规律。但是，当 $c=4$ 时，可以很明显的观察到，算法 eAll，laAll，eInfoA 和 laInfoA 的分类准确率达到了最大值，并优于 1-NN 算法。另外，懒惰式关联式分类器在与 SCBA 算法的所有对比中，均优于或等于 SCBA 算法的分类准确率，其中 laInfo 算法对比于 eInfo 算法最高能提升 12.6%的准确率。从图 4.6 中还可以观察到，采用最大置信度和或者最大信息增益和评价标准的算法，如 eAll，eInfoA，laAll，laInfoA 等的分类准确率高于采用最大置信度或者最大信息增益评价标准的算法。原因在于前者所采用的评价标准综合考虑了所有能够匹配相应测试实例的类序列规则，而前者只是根据排序最靠前的一条规则进行判断，而排序在前面的规则有可能出现信息增益较大，但置信度不高等情况。

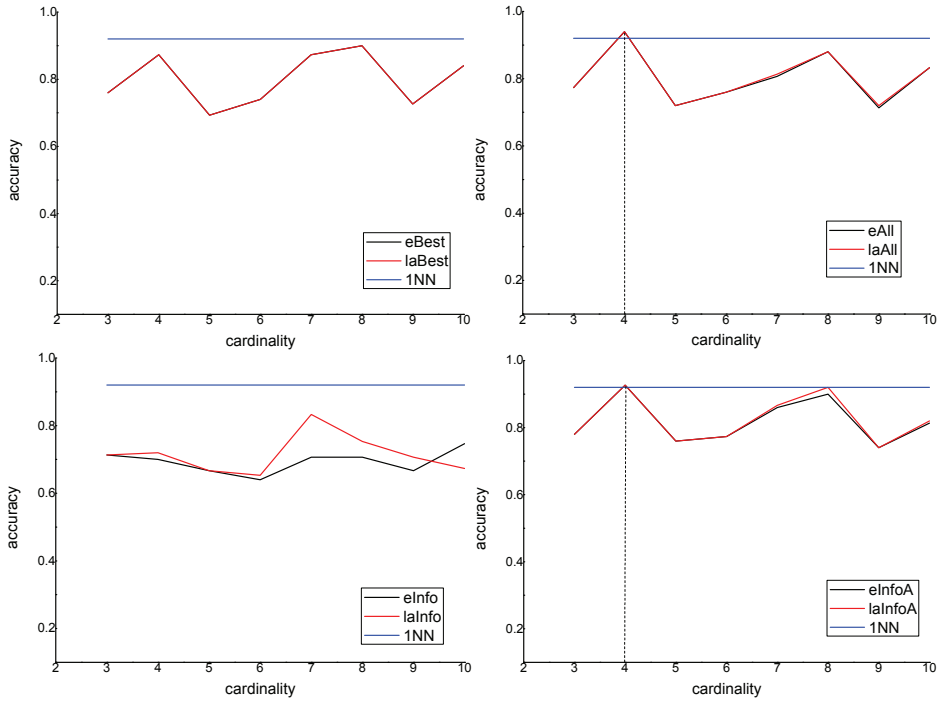

图 4.6 *c* 从 3 变化到 10 时分类准确率的变化

Figure 4.6 Curves of accuracy when *c* varies from 3 to 10

（2） *c* 不变，改变 *l* 的大小

此部分将测试 *c* 不变，*l* 变化时各个分类器分类准确率的变化。为保持简洁性和一致性，此处将 *c* 值统一设置为 4。由于每个数据集的属性的个数不同，而 *l* 又刚好表示了一个序列符号所能代表的实值型的时间序列子序列的长度，为保持一致性，此处假设数据集中的实例的个数为 *N*，实例的属性个数为 *T*，最小支持度为 1/*N*，*l* 的变化范围设置为 *T*/11 到 *T*/2。

图 4.7 仍选取 *Gun/NoGun* 数据集进行测试。该数据集的属性个数为 151（包含类属性），*l* 的变化范围为 13 到 75。从图中可以观察到，当 *l* 不断增大时，各个关联式分类器的分类准确率在出现较大幅度的波动的情况下，整体上呈下降趋势；而 1-NN 分类器由于不受 *l* 变化的影响，在整个测试过程中其分类准确率保持不变。通过仔细分析可以发现，基于置信度或信息增益评价标准的懒惰式关联式分类器的分类准确率的峰值出现在 $l = 15$，即属性个数的十分之一处，并且峰值的分类准确率大于 1-NN 分类器的分类准确率。当 *l* 不断变化时，算法 eAll，eInfoA，laAll 和 laInfoA 的分类准确率在多数情况下分别优于 eBest，eInfo，eBest 和 eInfoA 的分类准确率。

综上所述，当在选取评价标准用于关联式分类时，从分类准确率方面考虑，应优先考虑基于最大置信度和或最大信息增益和评价标准的懒惰式分类器，若进一

步从分类时间上考虑，则应优先选择最大置信度和评价标准。

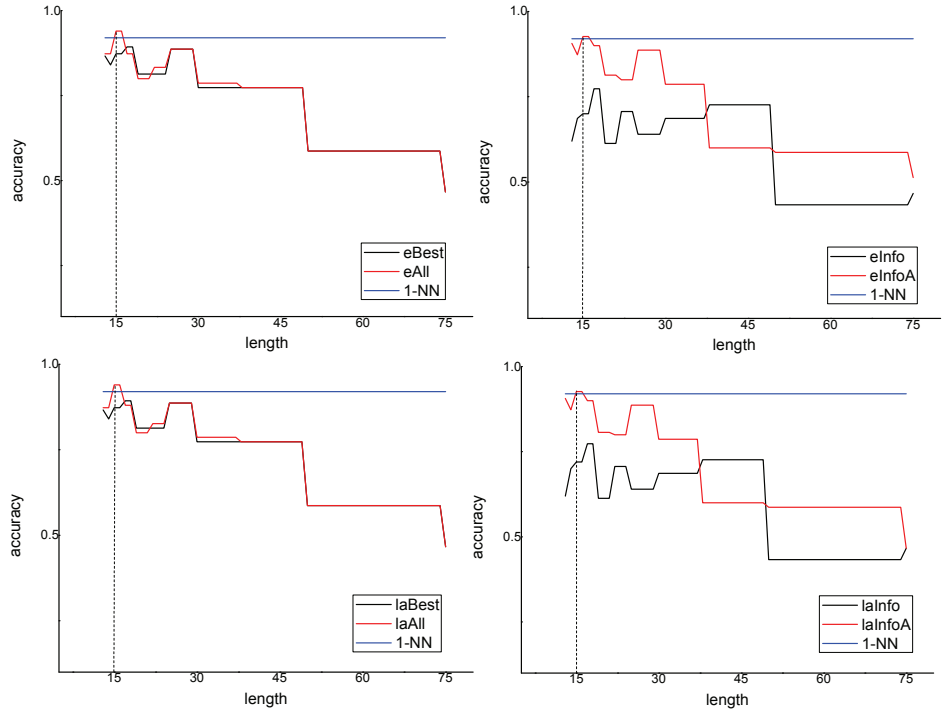

图 4.7 *l* 变化时分类器在 *Gun/NoGun* 数据集上的准确率变化曲线

Figure 4.7 Curves of accuracy on *Gun/NoGun* dataset when *l* varies

表 4.8 懒惰关联式分类器在不同评价准则下的分类准确率

Table 4.8 Accuracy of lazy associative classifier under different evaluations

Data	laBest	laAll	laInfo	laInfoA
CBF	81.89%	**95.33%**	59.44%	90.22%
Coffee	70%	**92.86%**	70%	77.14%
ECGFiveDays	60.28%	**86.64%**	56.33%	69.34%
Cricket	**95.92%**	93.88%	90.82%	89.80%
Cricket_new	7.81%	**50%**	7.81%	**50%**
DiatomSizeReduction	85.95%	**90.52%**	60.13%	68.30%
FaceAll	46.27%	**60.71%**	7.46%	46.21%
Gun_Point	87.33%	**94%**	70%	92.67%
ItalyPowerDemand	89.50%	**94.36%**	91.93%	93.59%
Lighting7	28.77%	**50.68%**	28.77%	45.21%
Lighting2	60.66%	**78.69%**	73.77%	59.02%
MoteStrain	72.68%	**82.27%**	74.04%	81.63%
SonyAIBORobotSurfaceII	70.09%	**82.16%**	67.37%	75.76%
SonyAIBORobotSurface	49.25%	67.72%	51.08%	**75.37%**
Synthetic_Control	83%	**94.33%**	33.33%	91.33%
wafer	97.47%	**99.06%**	94.01%	98.99%
Haptics	37.01%	**43.18%**	30.84%	29.55%

4.5.3 对比实验

此部分将首先评估不同度量方式下懒惰关联式分类器的分类准确率，然后将所提出的懒惰式关联式分类算法与 1-NN 以及 FastShapelet 算法作对比实验。其中 1-NN 算法是目前处理时间序列分类问题的最有效的算法，而 FastShapelet 算法同样采用了 SAX 转换技术，同时具有可解释性好，分类准确率高等特点。

由 4.4.2 节可知，懒惰式关联式分类器的分类准确率能够在 $c = 4$ 并且 $l = T/10$ 时到达峰值。表 4.8 展示了本章多提出的懒惰关联式分类器在不同评价准则下的分类准确率。从表中很容易得出，laAll 是四种评价方式中最好的分类器，它在 17 个数据集中的 15 个上表现最好。对比与 laInfo 和 laInfoA 算法，后者在 17 个数据集的 14 个上具有较高的分类准确率，尤其在 synthetic_control 数据集上，分类准确率的增幅达 58%。

综合考虑分类准确性和运算时间之后，选取 laAll 算法与 1-NN，FashShapelet 算法作对比实验，并设置 $c = 4$，$l = T/10$。为保证公平起见，在与 1-NN 算法进行对比时，使用了 WEKA 的默认设置；在与 FastShapelet 算法对比时，使用了 FastShapelet 算法的原作者所提供的代码并根据他们所推荐的参数设置进行实验。所有的分类器均是根据训练集建立并在测试集上测试的。具体测试数据如表 4.9 所示。

表 4.9 laAll, 1-NN 和 FastShapelet 在不同数据集上的准确率比对

Table 4.9 Accuracy of laAll, 1-NN and FastShapelet on different datasets

Data	laAll	1-NN	FastShapelet	Number of classes
CBF	**95.33%**	85%	94.22%	3
Coffee	**92.86%**	75%	**92.86%**	2
ECGFiveDays	86.64%	79.68%	**99.88%**	2
Cricket	93.88%	87.76%	**97.86%**	2
Cricket_new	**50%**	43.75%	47.32%	2
DiatomSizeReduction	90.52%	**93.46%**	87.25%	4
FaceAll	60.71%	**71.37%**	58.93%	14
Gun_Point	**94%**	91.33%	**94%**	2
ItalyPowerDemand	94.36%	**95.53%**	92.42%	2
Lighting7	50.68%	57.54%	**60.27%**	7
Lighting2	**78.69%**	75.41%	75.41%	2
MoteStrain	82.27%	**87.86%**	79.79%	2
SonyAIBORobotSurfaceII	82.16%	**85.94%**	75.03%	2
SonyAIBORobotSurface	67.72%	**69.56%**	68.55%	2
Synthetic_Control	**94.33%**	88%	91.33%	6
wafer	99.06%	99.55%	**99.84%**	2
Haptics	**43.18%**	37.12%	38.44%	5

所有算法在 17 个数据集上完成了测试（舍弃了那些无法在 24 小时内完成的数据集）。首先对比 laAll 和 1-NN 分类器，从表中可以观察到，在这 17 个数据集中，laAll 算法在 9 个数据集上的分类准确率较高（最高提升达 17.86%），1-NN 算

法在 8 个数据集上的分类准确性较高。在 laAll 与 FastShapelet 的对比中，laAll 算法在 10 个数据集上的分类准确率较高，FastShapelet 算法在 5 个数据集上的分类准确性较高，在 *Gun_Point* 和 *Coffee* 数据集上，两者的分类准确率相同。由此可以得出，基于 SAX 的懒惰式时间序列关联式分类器和 1-NN 算法，FastShapelet 算法相比，具有较好的分类准确率。从表 4.9 中还可以看出，FastShapelet 算法比较擅长处理二值分类问题，而 laAll 在多值和二值分类问题上均具有较好的分类准确率。下一节将阐述本章所提出算法的可解释性。

4.6 实例解析

此部分将通过实例解析，阐述懒惰式时间序列关联式分类器在解决时间序列分类问题时的可解释性。

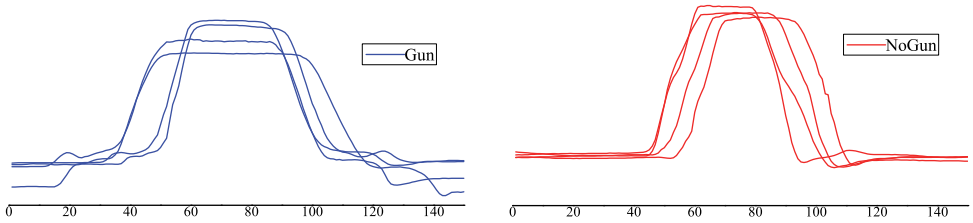

图 4.8 *Gun/NoGun* 数据集中的两类时间序列

Figure 4.8 Two different classes of *Gun/NoGun* dataset

4.6.1 *Gun/NoGun* 数据集

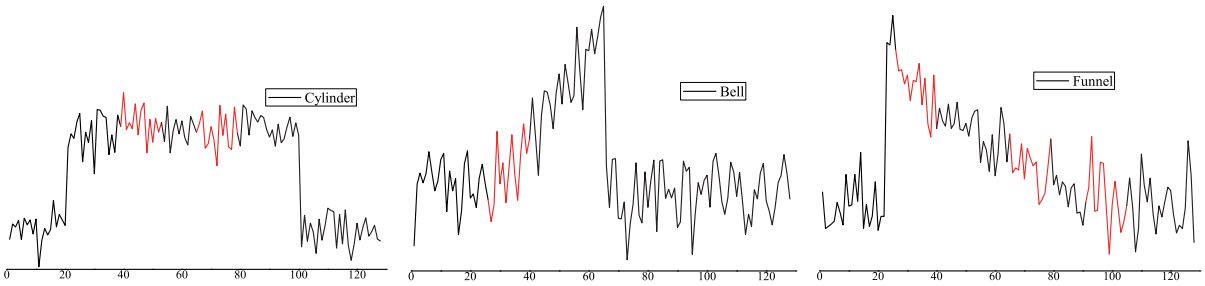

图 4.9 *CBF* 数据集中的三类时间序列

Figure 4.9 Three different classes of *CBF* dataset

经典的 *Gun/NoGun* 数据集是一组根据动作捕捉而产生的时间序列，它是时间序列中最被广泛应用的数据集之一。此数据集包含 200 条实例，其中 50 条作为训练集，另外 150 条作为测试集，数据集中的每一条实例的长度均为 151（包含类属

性）.图 4.8 展示了该数据集中 *Gun* 与 *NoGun* 类别的代表性序列，其中 *Gun* 序列表示行动者手中有枪，而 *NoGun* 序列表示行动者手中没有枪。根据懒惰式关联式分类器得到的最好的规则如图 4.1（*a*），4.1（*b*）所示。基于欧式距离的 1-NN 分类器在此数据集上的分类准确率为 91.33%。而懒惰式分类器的分类准确率为 94%，与 FastShapelet 的分类准确率相同，是当前已知的在 *Gun/NoGun* 数据集上的最高分类准确率。

4.6.2 CBF 数据集

由于关联规则能够对知识进行简洁的、直观的描述，关联规则挖掘已引起了研究者广泛的关注。关联规则在分类 *CBF* 数据集是一组人造数据集，里面的三个类别 **C**ylinder，**B**ell，**F**unnel 分别代表了三种简单的人造信号。此数据集包含 930 条实例，其中 30 条作为训练集，另外 900 条作为测试集，数据集中的每一条实例的长度均为 129（包含类属性）。图 4.9 给出了 CBF 数据集中三种不同类别信号的代表性时间序列。

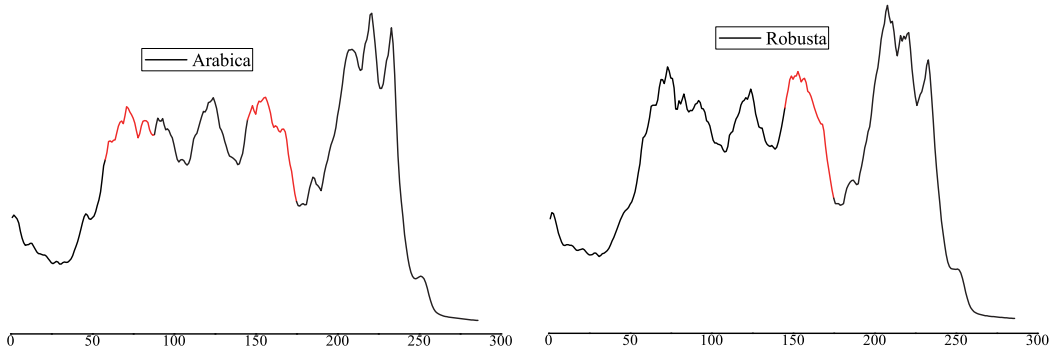

图 4.10 Coffee 数据集中的两类时间序列

Figure 4.10 Two classes of time series from the Coffee dataset

针对每一类时间序列，懒惰式关联式分类器得到的最好的规则分别为$\{A_3 = d, A_5 = d\} \rightarrow Cylinder$，$\{A_2 = b\} \rightarrow Bell$，$\{A_2 = d, A_5 = b, A_7 = b\} \rightarrow Funnel$。映射到时间序列上即为图 4.9 中不同序列的红色标注部分所示。从图中容易观察到，类序列规则$\{A_3 = d, A_5 = d\} \rightarrow Cylinder$ 反映了 *Cylinder* 信号在序列中部比较平稳的特征，规则$\{A_2 = b\} \rightarrow Bell$ 反映了 *Bell* 信号在序列前部逐步上升的特征，而规则$\{A_2 = d, A_5 = b, A_7 = b\} \rightarrow Funnel$ 充分反映了 *Funnel* 信号整体呈下降趋势的特征。基于欧式距离 1-NN 分类器和 FastShapelet 在此数据集上的分类准确率分别为 85%和 94.22%。然而，在同样的任务中，通过使用懒惰式关联式分类器，得到了 95.33%的分类准确率，明显优于基于欧式距离的 1-NN 分类器和 FastShapelet 分类器。

4.6.3 Coffee 数据集

Coffee 数据集是咖啡豆的一个光谱图集。在化学计量学中，它能够用于食物种类的分类，此任务也被广泛应用于食品安全和质量检测中。图 4.10 中所示即为咖啡豆的两个变种 *Arabica* 和 *Robusta* 的光谱图。此数据集包含 56 条实例，其中 28 条作为训练集，另外 28 条作为测试集，数据集中的每一条实例的长度均为 287（包含类属性）。

Arabica 和 *Robusta* 在总体上是相似的，这导致很难区分 Coffee 数据集中的实例。懒惰关联式分类器找到的最好的规则分别为$\{A_2 = c, A_5 = c\} \rightarrow$ *Arabica* 和$\{A_5 = d\} \rightarrow$ *Robusta*，如图 4.10 红色标注部分所示。从图中的红色标注部分可以发现，*Robusta* 咖啡豆的光谱序列波动幅度较大，而 *Arabica* 咖啡豆的光谱序列相对平滑。基于 ED 的 1-NN 分类器在此数据集上的分类准确率为 75%。而懒惰式分类器在此数据集上的分类准确率为 92.68%，与 Fastshapelet 的准确率相同。

4.7 本章小结

本章阐述了关联式分类算法在时间序列分类问题上的可解释性；在使用 SAX 技术将实值型高维数据转换成离散型符号序列的基础上，首先通过改进 CBA 算法来发现类序列规则，然后针对急切式 SCBA 算法的缺点，提出了一种懒惰式的关联式分类器，并提出了四种不同的类序列规则评价标准；通过实验测试了不同规则评价方法和参数设置对分类准确性的影响，验证了关联式分类算法在时间序列分类问题上的可扩展性和高效性；并通过将所提出的算法与经典的 1-NN 分类器和 FastShapelet 作比较，说明了懒惰式时间序列关联式分类算法具有较好的分类准确率。

但是，应用于时间序列的关联式分类器仍存在一些问题。比如当数据集较大时，随着频繁序列和发现的类序列规则的不断增多，可能会出现内存溢出等情况；当最小支持度的阈值较小时，算法的运行速度较慢等，这些都是以后需要进一步研究解决的重点问题。

5 时间序列对齐算法及其在 *k*-NN 中的应用

从之前的描述可知，基于 ED 或 DTW 的 *k*-NN 算法是时间序列分类算法中的基准分类器。本章将针对 *k*-NN 分类方法可解释性较差的弱点，依据时间序列对齐方法，提出相应的解决方案，并寻找出时间序列中的辨别性子序列。

5.1 引言

时序度量学习算法主要用于时间序列间的对比，它在时间序列分类、聚类以及分析等领域有着至关重要的作用，它也被广泛应用于姿势识别[99-102]，手写字符识别[103]，音乐信息校准[12, 104]，语音识别[105, 106]，基因序列比对[107]等领域。

时序对齐，包括基于 ED 的和基于 DTW 的时序对齐策略，是一种根据时间序列相似性或非相似性度量来连接或匹配时间序列，从而提供时序度量的方法。其中基于欧式距离的对齐策略是一种线性的映射，而 DTW 是一种非线性映射。

DTW 用于寻找两个序列之间的最优对齐方式，基于 DTW 的 *k*-NN 也是当前用于时间序列分类的最好的算法之一。它已经被成功应用于分类那些类内具有全局相似性的，并有较大范围时间延迟的时间序列。但是在某些实际应用中，DTW 不能很好地处理那些类内具有不同行为特性，而类间具有相似行为特性的复杂的时间序列。究其缘由，在于标准的 DTW 是一种按照一对对时间序列来逐步 (pairwise alignment) 比较的对齐策略，它忽视了类内或类间时间序列的关联性；另外，DTW 赋予对齐序列的每一维度同样的权值，并且对齐过程未能有效利用其他数据分析方法（如聚类，分类等），因此削弱了其处理复杂时间序列数据的能力。例如，图 5.1 展示了真实电力消费数据的样本。此数据记录了某个家庭中几乎一年 (共 349 天) 的电力消耗情况。每一条时间序列根据每十分钟的采样情况记录了一天的电力消耗情况。在分类为 *Low* 的时间序列中，其在耗电高峰期晚六点到八点间(图中范围[108, 120])的耗电量低于平均的消耗水平；相应的在分类为 *High* 的时间序列中，其在高峰期的耗电量高于平均水准。从图中可以很明显得观察到，同类数据有很大的不同，而正是这种数据暴露了 DTW 的缺点。

研究者已经提出了一些 DTW 的改进方法并将其应用到时间序列分类或聚类问题中。他们中的大多数方法集中于定义更好的限制策略，全局的或半全局的加权方法。例如，第一种改进方式依赖于 Sakoe-Chiba[105] 或者 Itakura[108]方法来限制 DTW 方法的对齐空间，或者通过预处理时间序列，比如获取高阶形状特征[109]，平均化时间序列[110]，通过多层次方法近似求解 DTW 最优路径[111]等，加快 DTW 的

计算效率。第二种改进方法主要集中于定义全局的加权策略。比如 Jeong 等人[46]提出了一种基于 Sakoe-Chiba 限制的加权策略，根据 DTW 代价矩阵，其定义了一种递增函数用于赋予离对角线近的点较低的权值，离对角线远的点较高的权值。另一个例子来源于姿势识别领域，Reyes 等人[100]提出了一种针对多维时间序列的辨别性加权策略。其根据 DTW 的均值来计算类内和类间的方差，并将同样的权值赋予所有的姿势类别。第三种方法是一种半全局的加权策略[99, 112]，它也是 Reyes 等人[100]所提出算法的扩展，其主要区别在于为每一个姿势类别的特征赋予不同的权值。

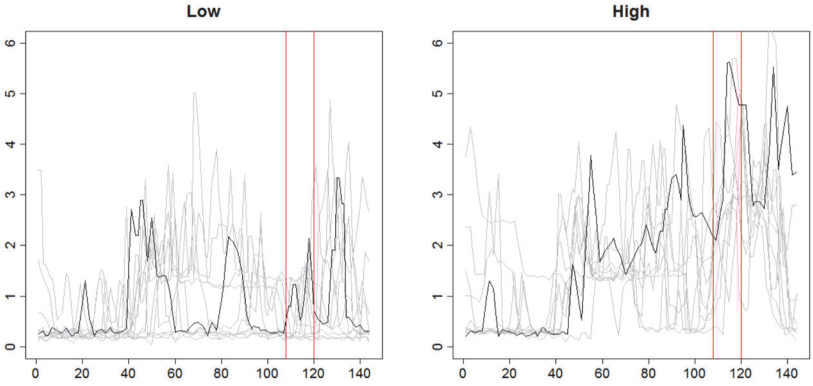

图 5.1 *Low* 和 *High* 类别的电力消耗数据

Figure 5.1 The electrical power consumption of *Low* and *High* classes

尽管上述工作能够产生较准确的时间对齐方式，但仍存在许多缺点，比如给所有类别和时间点(time stamp)赋予相同的权值（全局加权策略）；缺乏对类内以及类间局部不同行为的考虑；辨别性子序列是通过标准 DTW 学习到的，而标准 DTW 平等地对待每一个特征；权值并没有随着时间的推进而改变等等。在实际应用中，很容易想到时间序列的某一段是明显区别于其他地方的，所以不同的时间点应赋予不同的权值。换言之，为发现时间序列的局部辨别性行为，需要定义一种局部加权策略，赋予每一条时间序列的每一个时间点不同的权值。Frambourg 等人[113]对此具有挑战性的问题中进行了初步讨论，它依赖于方差 / 协方差矩阵来增强或削弱对齐关系，但由于其需要大量的矩阵运算，导致其在面对大量数据时效率会大幅下降。另外，虽然最近有研究者[106]提出了一种专门针对语音数据的局部加权方式，它忽视了类内方差对权值的影响。

为解决上述问题，本章提出了一种基于辨别性特征的时间序列对齐方式，它在增大类间方差的同时降低类内方差，根据离它最近的同类序列和不同类序列迭代地获取权值。

众所周知，基于 DTW 或者欧式距离的 *k*-NN 分类器是时间序列分类领域最成功的以及最被广泛使用的分类器（R 语言提供了 DTW 对齐可视化程序包[114]）。但是，由于 *k*-NN 对噪音敏感，*k* 的变化很容易引起已预测类标的变化，特别是当不

同类的数据间具有相同的全局特征时此情况更易发生。如图 5.2 所示，针对 Coffee 数据集，通过多维尺度分析[115]（Multidimensional Scaling，MDS），基于 DTW 或 ED 的 *k*-NN 分类器并不能很好地区分两种类别，但是当应用本章将要介绍的辨别性局部加权 DTW (Weighted DTW，WDTW)时，两者被十分明显的区分开来。另外，由于 *k*-NN 是一个懒惰式的分类器，它不像贝叶斯网络或者决策树那样具有明显的可解释性，但是本章将要介绍的 WDTW 方法展现了 *k*-NN 分类器的可解释性。

本章致力于描述一种具有辨别性的基于局部加权 DTW 的时间对齐方式，并将

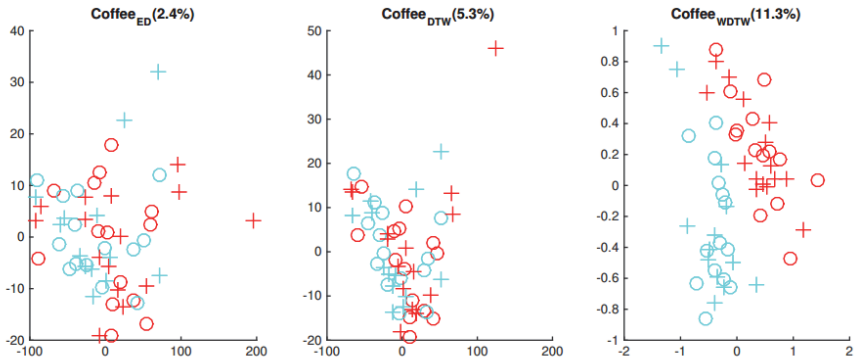

图 5.2 通过多维尺度分析(MDS)得到 Coffee 数据集在不同度量方式下的潜在结构，不同颜色代表不同的类别，圆形表示训练集，十字表示测试集。

Figure 5.2 Underlying structure detected by multidimensional scaling (MDS) for different measures on dataset *Coffee*, different colors represent different classes, and circle symbols mean the training set, cross symbols the test set.

其应用于 *k*-NN 中。所提出时间序列加权方法通过最近的同类序列和不同序列来迭代的多步对齐 (multiple alignment) 时间序列并获取辨别性特征。它能够在增大类间方差的同时降低类内方差，并提高分类准确率。主要工作可总结为以下几点：

（1）提出了一种新的有效的时间序列加权模型，并为每一条时间序列对齐点提供权值；

（2）提出了两种不同的 DTW 加权方式来发现辨别式的子序列；

（3）和其他基于相似性或不相似性的 *k*-NN 分类器相比较，展示了其可解释性；

（4）将所提出模型扩展至多变量时间序列分类问题，并讨论了其特殊情况，加权的欧式距离；

（5）通过在多个公共数据集上与多个方法的对比，展现所提出模型在时间序列分类问题上的有效性。

本章组织结构如下。第 5.2 节介绍序列对齐的背景知识。基于 DTW 的时间序

列局部加权模型以及其学习算法将在 5.3 节和 5.4 节进行详细介绍。第 5.5 节将通过实验评价所提出模型在多个数据集（包括单变量和多变量时间序列）上的分类效果。第 5.6 节通过在多个人工数据集和真实数据集上的实例解析，阐释所提出模型的可解释性。最后进行本章的总结工作。

5.2 背景知识

定义 5.1：时间序列对齐

对于一个包含有 N 个单变量离散时间序列的数据集 $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$，每一条时间序列 $\boldsymbol{x} = \{x_1, x_2, \ldots, x_T\}$的长度为 T[2]。两个时间序列 $\boldsymbol{x}_i$ 与 $\boldsymbol{x}_j$ 间的一个对齐排列 π 的长度为$|\pi| = m$ $(T \le m \le T\text{-}1)$，它是由 $\boldsymbol{x}_i$ 与 $\boldsymbol{x}_j$ 的对齐点所组成的，即

$\pi = ((\pi_1(1), \pi_2(1)), (\pi_1(2), \pi_2(2)), \ldots, (\pi_1(m), \pi_2(m)))$，

其中定义在 1 到 m 的成对π_1 和π_2 遵循以下限制边界和单调性条件：

(1) $1 = \pi_1(1) \le \pi_1(2) \le \ldots \le \pi_1(m) = T$;

(2) $1 = \pi_2(1) \le \pi_2(2) \le \ldots \le \pi_2(m) = T$;

(3) $\forall l \in \{1, \ldots, m\}$, $\pi_1(l+1) \le \pi_1(l) + 1$, $\pi_2(l+1) \le \pi_2(l) + 1$, $(\pi_1(l+1) - \pi_1(l)) + (\pi_2(l+1) - \pi_2(l)) \ge 1$。

表 5.1 符号表

Table 5.1 Symbol table

符号	符号释义
$\boldsymbol{a}$ / $\boldsymbol{b}$	一条离散时间序列
$\boldsymbol{w}$ / $\boldsymbol{w}_i$	时间序列权重向量
π	两条时间序列间的一个对齐排列
M	对齐点的个数
Y	时间序列 $\boldsymbol{x}$ 的类标
A	两条时间序列间的所有对齐排列
t	时间点
c	时间序列 $\boldsymbol{x}$ 的最近邻的个数
cNN^+/cNN_i^+	$\boldsymbol{x}$ / $\boldsymbol{x}_i$ 的正例集合
cNN^-/cNN_i^-	$\boldsymbol{x}$ / $\boldsymbol{x}_i$ 的负例集合
$\boldsymbol{x}_{it}^+$	$\boldsymbol{x}_i$ 在时间点 t 的正对齐集合
$\boldsymbol{x}_{it}^-$	$\boldsymbol{x}_i$ 在时间点 t 的正对齐集合
q	多维时间序列每个点包含的数值的个数
α	影响加权机制的控制参数
min	训练集中不同类别实例的最小

另外，定义 $\mathcal{A}$ 为两个序列间所有对齐排列的组合，$y \in \{1, \ldots, K\}$为时间序列 $\boldsymbol{x}$ 的类标。本章中所涉及到的符号如表 5.1 所示。

[2]因为本章所使用的时间序列对齐以及非相似性度量方法均能定义在多变量时间序列中，所以此定义也可扩展至多变量，可能具有不等长度的时间序列，5.3.2 节将进行相关阐述。

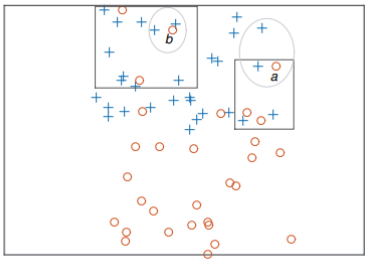

图 5.3 对于时间序列 ***a*** 或 ***b***, cNN^{+}由矩形中 2 个($c = 2$)最近的圆形组成，而 cNN^{-}由椭圆中两个不同类别的十字型组成。

Figure 5.3 For time series ***a*** or ***b***, cNN^{+} is composed of 2 ($c = 2$) closest series of the same class (red circle) in the area of rectangle, while cNN^{-} is composed of 2 closest series of the different class (blue cross) in the area of eclipse.

5.3 时间序列局部加权模型

本章介绍的局部加权模型综合了 Frambourg 等人介绍的方差/协方差对齐模型和 Weinberger 等人提出的最大化边际度量学习算法的优点[116]。为计算每一条时间序列每一个时间点的权值，对每一个 ***x*** 我们计算其两个集合 cNN^{+}和 cNN^{-} (分别称为正例集和负例集)，它们分别由 ***x*** 的 c 个距离最近的同类别时间序列和不同类别时间序列组成。如图 5.3 所示，cNN^{+}和 cNN^{-} 都是从训练集中得到的。注意，本章中定义的正例集和负例集不同于 Weinberger 等人提出的目标集(targets)和入侵集(impostors)，其中入侵集指的是侵入到目标集范围内的不同类别的实例，而正例集和负例集之间不存在此限制。相应的，对时间序列 $\boldsymbol{x}_i$ 的每一个时间点 t，其对应于两个集合 $\boldsymbol{x}_{it}^{+}$和 $\boldsymbol{x}_{it}^{-}$，定义为:

$$\boldsymbol{x}_{it}^{+} = \{t'/(t, t') \in \pi_{it'}, x_{i'} \in cNN_i^{+}\}$$

$$\boldsymbol{x}_{it}^{-} = \{t'/(t, t') \in \pi_{it'}, x_{i'} \in cNN_i^{-}\}$$

其中$\pi_{it'}$是 x_i 和 $x_{i'}$之间的标准 DTW 对齐排列，cNN_i^{+}和 cNN_i^{-}是的正例集和负例集。

对于每一个 $\boldsymbol{x}_i$，赋予其一个权值向量 $\boldsymbol{w}_i = (w_{i1}, w_{i2},\ldots,w_{iT})$，此向量根据 $\boldsymbol{x}_i$ 中每个点对拉近正例集和远离负例集的贡献而确定。所以此处定义 w_{it} 为：

$$w_{it} = \frac{A_{it}}{\sum_{t'=1}^{T} A_{it'}}, w_{it} > 0, \sum_{t=1}^{T} w_{it} = 1 \qquad (5.1)$$

其中$|\pi_{it'}|$是 x_{it} 对齐点的个数，除以$|\pi_{it'}|$意味着获取正例集和负例集对齐点的均值。

$$A_{it}=\frac{\sum_{x_{i'}\in cNN_i^-}\frac{1}{|\pi_{ii'}|}\left(\sum_{(t,t'\in\pi_{ii'})}(x_{it}-x_{i't'})^2\right)}{\sum_{x_{i'}\in cNN_i^+}\frac{1}{|\pi_{ii'}|}\left(\sum_{(t,t'\in\pi_{ii'})}(x_{it}-x_{i't'})^2\right)} \tag{5.2}$$

对于公式（5.2）而言，分母可认为是 x_{it} 的类内方差，而分子是类间方差。由于此模型的目标是在最大化类间方差的同时缩小类内方差，A_{it} 或者 w_{it} 越大，x_{it} 的辨别性就越高。

尽管每一个 x_{it} 都至少拥有一对正例集或负例集的对齐点（当正例集存在时），当所有的 $x_{i't'}$ 都与 x_{it} 相等时，公式（5.2）的分母将变为 0，从而公式（5.2）变得无意义。为避免此情况，公式（5.2）被修改为：

$$A_{it}=\frac{\sum_{x_{i'}\in cNN_i^-}\frac{1}{|\pi_{ii'}|}\left(\sum_{(t,t'\in\pi_{ii'})}(x_{it}-x_{i't'})^2\right)}{\sum_{x_{i'}\in cNN_i^+}\frac{1}{|\pi_{ii'}|}\left(\sum_{(t,t'\in\pi_{ii'})}(x_{it}-x_{i't'})^2\right)+\varepsilon} \tag{5.3}$$

其中ε是一个非常小的数，比如 10^{-6}。

5.3.1 局部加权 DTW

不可否认，DTW 是最普遍的以及最被广泛使用的时间序列度量方法之一，它最简单的定义形式如下：

$$\mathrm{DTW}(\boldsymbol{x}_i,\boldsymbol{x}_{i'}):=\min_{\pi\in\mathcal{A}}\left(\sum_{(t,t'\in\pi_{ii'})}\|x_{it}-x_{i't'}\|^2\right) \tag{5.4}$$

其中 $\boldsymbol{x}_i$ 的每一个点都有相同的权重 $1/T$。

本章介绍的局部加权 DTW（locally Weighted Dynamic Time Warping, WDTW）非相似性度量方法计算加权序列$(\boldsymbol{x}_i, \boldsymbol{w}_i)$的和 $\boldsymbol{x}_{i'}$ 间的距离：

$$\mathrm{WDTW}((\boldsymbol{x}_i,\boldsymbol{w}_i),\boldsymbol{x}_{i'}):=\min_{\pi\in\mathcal{A}}\left(\sum_{(t,t'\in\pi_{ii'})}f(w_{it})\|x_{it}-x_{i't'}\|^2\right) \tag{5.5}$$

其中$\|x_{it} - x_{i't'}\|^2$ 为两者间的欧式距离。尽管存在一些高级的函数来估计 w_{it} 的贡献，为不失一般性和简洁性，此处将 $f(w_{it})$定义为如下所示的指数函数或者幂函数。

$$f(w)=e^{-\partial w} \tag{5.6}$$

$$f(w)=w^{-\partial} \tag{5.7}$$

对于公式（6）和（7），$\alpha\in\mathbf{R}^+$控制加权机制的影响，并且$\alpha=0$ 对应于标准的，

未加权的，均匀的 DTW。之所以用负号是为了保证优先选择最重要的时间点，因为 WDTW 是为了最小化所有时间序列时间点的值之间的非相似性。

5.3.2 多变量时间序列

对于多变量时间序列，每一条时间序列 $\boldsymbol{x}_i=\{\boldsymbol{x}_{i1},\boldsymbol{x}_{i2},\ldots,\boldsymbol{x}_{iT}\}$包含 T 个 q 维的数值变量，其中 $\boldsymbol{x}_{it}\in\mathbf{R}^q$ 并且 $\boldsymbol{x}_{it}=\{x_{it}^1,\ldots,x_{it}^q\}^{\mathrm{T}}$。尽管每条时间序列的长度 T 可能不同，但基于 DTW 度量方式能够应对。为将本章提出的 WDTW 方法应用于多变量时间序列，需要为每一条时间序列的每一个时间点的每一个数值变量计算权值，针对多变量时间序列的 DTW 被定义为

$$\mathrm{DTW}(\boldsymbol{x}_i,\boldsymbol{x}_{i'})\coloneqq\min_{\pi\in\mathcal{A}}\left(\sum_{k=1}^{q}\sum_{(t^k,t'^k)\in\pi_{ii'}^k}\|x_{it}^k-x_{i't'}^k\|^2\right) \tag{5.8}$$

因此，公式（5.5）可被修正为

$$\mathrm{WDTW}((\boldsymbol{x}_i,\boldsymbol{w}_i),\boldsymbol{x}_{i'})\coloneqq\min_{\pi\in\mathcal{A}}\left(\sum_{k=1}^{q}\sum_{(t^k,t'^k)\in\pi_{ii'}^k}f(w_{it}^k)\|x_{it}^k-x_{i't'}^k\|^2\right) \tag{5.9}$$

其中 $\boldsymbol{w}_i$ 从 $1\times T$ 维矩阵（单变量时间序列）转变为 $q\times T$ 维矩阵（多变量时间序列），相似的变化也发生在$\pi_{ii'}$上。

5.3.3 局部加权 ED

对于基于欧式距离的 *k*-NN 分类器，两个时间序列 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 之间存在一对一的关系。所以长度为 T 对齐排列π被定义为 T 对 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 之间的对齐元素：

$\pi=((\pi_1(1),\pi_2(1)),(\pi_1(2),\pi_2(2)),\ldots,(\pi_1(T),\pi_2(T)))$。

和 DTW 的定义相比较，很容易发现基于 ED 的度量是 DTW 的一个特例。实际上，DTW 代价矩阵的对角线就代表了 ED 度量，因此很容易将 WDTW 的思想应用到加权的 ED 中（Weighted Euclidean Distance，WED）：

$$\mathrm{WED}((\boldsymbol{x}_i,\boldsymbol{w}_i),\boldsymbol{x}_{i'})\coloneqq\sum_{t=1}^{T}f(w_{it})\|x_{it}-x_{i't'}\|^2 \tag{5.10}$$

w_{it} 的定义与公式（5.1）相同，而 A_{it} 被修正为：

$$A_{it}=\frac{\sum_{x_{i'}\in cNN_i^-}(x_{it}-x_{i't'})^2}{\sum_{x_{i'}\in cNN_i^+}(x_{it}-x_{i't'})^2+\varepsilon},\varepsilon>0 \tag{5.11}$$

5.4 基于 WDTW 的最近邻算法

为方便起见，本章将只介绍在单变量时间序列中学习局部加权 WDTW 和相应的 *k*-NN 分类器。

算法 1：WDTW($\boldsymbol{X}_{train}$,$\boldsymbol{X}_{test}$)
输入：训练集 $\boldsymbol{X}_{train}$,测试集 $\boldsymbol{X}_{test}$
输出：错误率 *errorRate*

```
1: 初始化：读取训练集 X_train,测试集 X_test; i = 1, 1 ≤ i ≤ N_train,j = 1,1 ≤ j ≤ N_test;最近同类或不同类序列的个数 c 以及最近邻的个数 k
2: repeat
3:   Update weight w_i of x_i (根据算法 2 更新权重)
4:   i = i + 1
5: until all x_i in X_train is learned
6: repeat
7:   Find k nearest neighbors for x_j using Eq.5
8:   Predict class label for x_j
9:   j = j + 1
10: until all x_j in X_test is labelled
11: Compare with true labels
12: return errorRate
```

图 5.4 基于 WDTW 的 *k*-NN 算法

Figure 5.4 WDTW based *k*-NN Algorithm

对于基于 WDTW 的 *k*-NN 分类器，需要找到每一条测试实例 $\boldsymbol{x}_j$ ($1 \le j \le N_{test}$)的 *k* 个根据公式（5.5）所得的距离最近的时间序列。在图 5.4 所示的 WDTW 算法中，同类/不同类时间序列的个数 *c* 将会在下一节进行讨论（第 1 行）。WDTW 算法和标准 DTW 算法之间的主要不同在于，前者需要根据图 5.5 所示的算法 2 来计算每一个 $\boldsymbol{x}_i$ 的权值 $\boldsymbol{w}_i$（第 2-5 行），而后者直接寻找最近邻，不同考虑权重的不同。两者建立 *k*-NN 分类器以及计算错误率的步骤相同（第 9-12 行）。

算法 2（图 5.5 所示）用于学习每一条训练序列的辨别性权重 $\boldsymbol{w}_i$，初始时 $\boldsymbol{w}_i^0 = (1/T,\ldots,1/T)$ 意味着每一个特征或者时间点的权重相同，所以$\Pi^{*(0)}$是根据标准 DTW 获得的最优对齐排列（第 1-6 行）。随着 $\boldsymbol{w}_i^p$ 的不断更新（第 14 行），权重可以变得不均匀，并且$\Pi^{*(p)}$变为根据 WDTW 获得的最优对齐排列（第 11 行）。需要注意的是，为得到最优排列$\Pi^{*(p)}$，每一次迭代都需要根据 $\boldsymbol{w}_i^p$ 和 *c* 找到 $\boldsymbol{x}_i$ 的 cNN_i^+和 cNN_i^-。该算法不断迭代直到 $\boldsymbol{w}_i^p$ 保持不变或者 $\boldsymbol{w}_i^p$ 和 $\boldsymbol{w}_i^{p-1}$ 非常接近。算法停止标准将在 3.5.2 节进行讨论。

依靠动态规划思想，计算 DTW 的最优对齐方式的时间复杂度为 $O(T^2)$。由于

本章所提出的算法 WDTW 是经典算法的加权扩展，其时间复杂度依然为 $O(T^2)$。但由于其需要根据 DTW/WDTW 寻找每一个训练实例的正例集和负例集，此举需要花费大量的时间。同样的，WDTW 也可以通过使用对齐排列在对角线附近的子集来加速[105, 108]。

算法 2: *WeightLearn*(*c*, $\boldsymbol{X}_{train}$)
输入: *c*，正、负例集中序列个数；$\boldsymbol{X}_{train}$，训练集
输出: 局部权重 $\boldsymbol{W}^p$

```
Compute standard DTW cost matrix for X_train (根据公式（5.4）)
Get Π*(0) (每对时间序列间的 DTW 对齐方式)
for x_i, 1 ≤ i ≤ N_train
    Find cNN_i+, cNN_i- according to Π*(0) and DTW cost matrix
    Compute w_i^0 (根据公式（5.1）)
end for
p = 0
repeat
     p = p + 1
    Compute WDTW cost matrix based on W^(p-1) (根据公式（5.5）)
    Get Π*(p) (每对时间序列间的 WDTW 对齐方式)
    for x_i, 1 ≤ i ≤ N_train
        Find cNN_i+, cNN_i- using Π*(p) and WDTW cost matrix
        Update w_i^p (根据公式（5.1）)
     end for
until W^p ≈ W^(p-1)
```

图 5.5 权重更新算法

Figure 5.5 Learning local weighting system

5.5 实验与评价

本节将首先描述所使用的实验数据集，然后介绍参数的设置方法，最后对比不同算法（包括 ED，DTW，基于指数函数的 WDTW/WED，基于幂函数的 WDTW/WED 等）的实验结果。

5.5.1 实验数据集

如表 5.2 所示，对比实验在 40 个公共数据集上完成[3]。数据集 Trajectories 是

[3] UMD 和 BME 获取于 http://ama.liglab.fr/~douzal/tools.html, Conseason 获于 http://ama.liglab.fr/~soheily/

表 5.2 数据集

Table 5.2 Summary of datasets

数据集	实例个数 N_{train}/N_{test}	长度 T	类个数 K	正例集限制 min
ItalyPowerDemand	67/1029	24	2	33
ECG200	100/100	96	2	31
Gun Point	50/150	150	2	24
MoteStrain	20/1252	84	2	10
SonyAIBORobotSurfaceII	27/953	65	2	11
CinC ECG torso	40/1380	1639	4	5
wafer	1000/6164	152	2	97
Car	60/60	577	4	11
ECGFiveDays	23/861	136	2	9
SwedishLeaf	500/625	128	15	26
Adiac	390/391	176	37	4
ChlorineConcentration	467/3840	166	3	91
Haptics	155/308	1092	5	18
Conseason	37/328	144	2	18
OliveOil	30/30	570	4	4
CBF	30/900	128	3	8
FacesUCR	200/2050	131	14	4
Lighting2	60/61	637	2	20
Plane	105/105	144	7	9
Symbols	25/995	398	6	3
Trace	100/100	275	4	21
TwoLeadECG	23/1139	82	2	11
CC	300/300	60	6	50
BME	30/150	128	3	10
UMD	36/144	150	3	12
MALLAT	55/2345	1024	8	2
Two Patterns	1000/4000	128	4	237
Beef	30/30	470	5	6
Coffee	28/28	286	2	14
FaceFour	24/88	350	4	3
Lighting7	70/73	319	7	5
MedicalImages	381/760	99	10	6
SonyAIBORobotSurface	20/601	70	2	6
FISH	175/175	463	7	21
FaceAll	560/1690	131	14	40
OSULeaf	200/242	427	6	15
WordsSynonyms	267/638	270	25	2
yoga	300/3000	426	2	137
InlineSkate	100/550	1882	7	9
Trajectories	286 / 2572	–	20	8

长度不同的多变量时间序列。数据集 Conseason 的错误率通过 10 次有放回抽样获取，而其他数据在获取时已被区分为训练集与测试集，所有错误率是在测试集上计算的，其计算公式如下：

datasets/Power%20Consumption.rar, Trajectories 获取于 https://archive.ics.uci.edu/ml/datasets/Character+Trajectories, 其他数据获取于 http://www.cs.ucr.edu/~eamonn/time_series_data/

$$ErrorRate = \frac{N_{test} - N_{correct}}{N_{test}} \tag{5.12}$$

其中 $N_{correct}$ 代表被正确分类的测试集的个数。另外，由于正例集于负例集均是从训练集中获取的，表 3.2 的最后一行 *min* 用于表示训练实例中不同类别实例的最小个数。比如，人工数据集 CBF 由三个不同的类别组成，分别为 *Cylinder*，*Bell* 和 *Funnel*，每一个类别在训练集中的个数分别为 10，12 和 8，所以 CBF 的 *min* 被设置为 8.

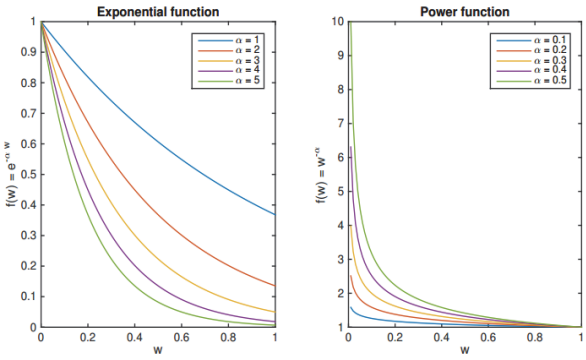

图 5.6 参数α的调谐效应

Figure 5.6 The tuning effect of the parameter α

5.5.2 参数设置

在本章的所有实验中，最近邻的个数 *k* 属于{1,3,5,7,9}中的一个。另外，最近正例集或负例集的个数 *c* 是很难确定的，因为 *c* 值的设定关系到辨别性特征的发现，设置不正确时将会很难发现辨别性的特征，进而影响分类准确率。为覆盖尽量多的实例并不失一般性，*c* 被设置为 $c \in \{1,2,3,5,10\}$并且 $c < min$。若 $c \geq min$，有可能发现不了足够的正例集。例如，CBF 的 *min* 被设置为 8，那么 *c* 只能从{1,2,3,5}中取，$c=10$ 时类别 *Funnel* 将不能得到足够的正例集，此时设置其权重为初始权重 1/*T*。具体的参数设置总结于表 5.3。

表 5.3 参数表

Table 5.3 Summary of parameters

参数	值
α_{exp}	1
α_{pow}	0.1
c	$c \in \{1,2,3,5,10\}$
k	$k \in \{1,3,5,7,9\}$
$\boldsymbol{w}_i^p \approx \boldsymbol{w}_i^{p-1}$	$\|\|\boldsymbol{w}_i^p - \boldsymbol{w}_i^{p-1}\|\|^2 \leq 0.1 * \|\| \boldsymbol{w}_i^0 \|\|^2$ 或者 p=4

参数α的调谐效应如图 5.6 所示。从图中可以很清楚的看到，当 w 从 0 到 1 不断变化时（$0<w<1$），指数函数的取值范围为从 1 到 0（$0<f(w)<1$），而幂函数的变化范围是从到 1（$1<f(w)<+\infty$）。随着α的增长，较大 w 之间的差别在逐渐减小，而较小 w 之间的差别在不断增大。图 5.7 展示了α 变化时分类错误率的变化（对于指数函数，α 变化范围为 0 到 10，而幂函数中的α 变化范围是 0 到 1，此实验中 $k=1$，$c=3$），分别在两个不同的数据集 ItalyPowerDemand 和 Beef 上进行了测试，其中实线表示指数函数的实验结果，虚线表示幂函数的实验结果。$\alpha=0$ 对应于标准的 DTW 算法。当α 从 0 增长到 10（或者相应的从 0 增长到 1）时，尽管分类错误率曲线有些许起伏，但整体是上升的。所以为简洁起见，本章在训练 k-NN 分类器时，将指数函数的α设置为 1，幂函数的α设置为 0.1。

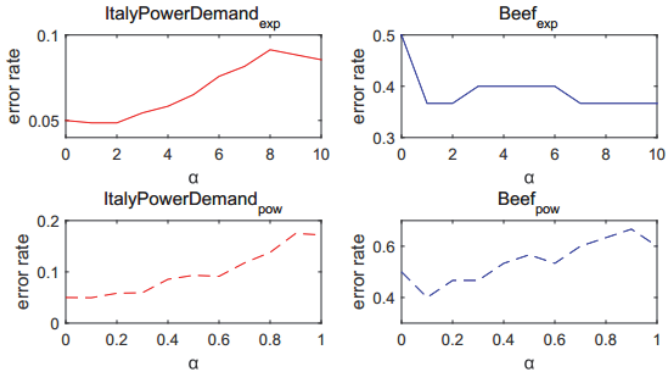

图 5.7 不同参数α下的错误率

Figure 5.7 Error rate under different parameter α

在图 5.5 所示的算法 2 中，本章所提出的模型必须保证每一条时间序列 $\boldsymbol{x}_i$ 的权重 $\boldsymbol{w}_i$ 收敛。如图 5.8 所示（此实验中 $c=3$，$\alpha_{exp}=1$，$\alpha_{pow}=1$ 保持不变），仅通过一次迭代，大多数的规范化欧式距离$||\boldsymbol{w}_i^p - \boldsymbol{w}_i^{p-1}||^2$ 就接近收敛于 0。为进一步保证收敛性，迭代次数被设置为 4。注意，若$||\boldsymbol{w}_i^p - \boldsymbol{w}_i^{p-1}||^2 \le 0.1*||\boldsymbol{w}_i^0||^2$ 为真，迭代停止。因为初始时 $\boldsymbol{w}_i^0=(1/T,\ldots,1/T)$，0.1 倍的$||\boldsymbol{w}_i^0||^2$ 足以满足条件 $\boldsymbol{w}_i^p \approx \boldsymbol{w}_i^{p-1}$。

5.5.3 结果展示

此部分将本章所提出的基于 WED/WDTW 的 k-NN 分类器与经典的基于 ED/DTW 的 k-NN 相比较。表 3.4 展示了所有对比算法在 40 个公共数据集上的分类错误率，其中加粗的数值表示所对比方法中最优的结果。如果一个分类错误率与最优结果相比具有显著性差异[117]（显著性水平设置为 0.01）,则用“*”标注它。需要注意的是，尽管实验时对所有参数进行了网格式搜索（grid search），最优分类错误率可能在不同的 k 和 c 中出现多次，为简洁明了起见，当多个分类错误率相同

时，表中只列出最小的 k 值情况下最小的 c 值（当 c 值存在时）。

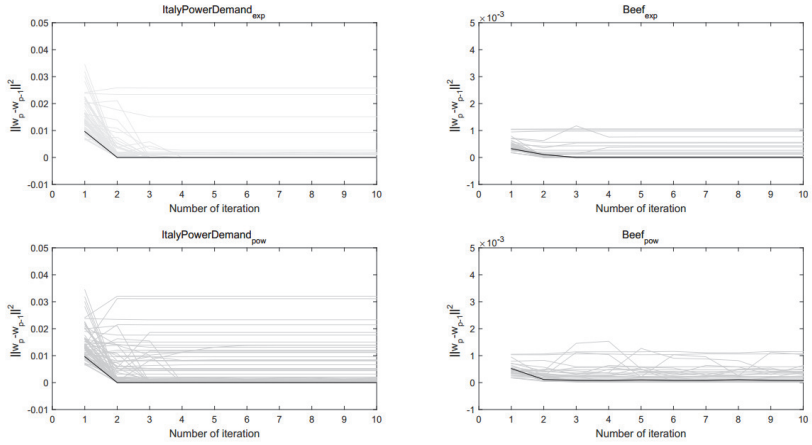

图 5.8 权重的收敛性示例

Figure 5.8 The convergence of weights

对比与标准的度量算法，表 5.4 中的误分类错误率展现了本章所提出的局部加权度量方法 WED 和 WDTW 在区分复杂时间序列时的高效性。尽管标准度量方式在两个数据集 MoteStrain 和 WordsSynonyms 上具有较好的分类错误率，但它与本章所提出方法之间并没有显著性差异，也就是说，它们在因此下一个测试实例的类标时会得到相同的结果。本章所提出的算法在 40 个数据集中的 38 个上表现较好（包括平局情况，另外需注意因为其长度不相等，ED/WED 不能处理多变量数据集 Trajectories）。对于没有时间延迟或具有较少延迟的序列，WED 算法能够提升其分类准确率，原因在于这些序列不需要或只有少量的时间弯曲，而 WDTW 能够很好的处理具有大量时间延迟或时间弯曲的序列。从表中也可以看到，基于 1-NN 的度量算法在多于 80%的数据集上优于其他 k 近邻算法，这也和 Wang 等人[42]以及 Petitjean 等人[110]的描述相符。对于最近邻个数 k 与正例集或负例集个数 c 之间的关系，k 大于 c 的情况仅仅在 158 次测试中出现了 24 次，所以为分类准确率考虑，c 应该大于 k。

另外，通过在 ED/DTW/WED/WDTW 的逐对不相似矩阵上使用 MDS，能够可视化数据集的潜在结构。图 5.9 在二维空间中展示了数据的潜在结构（不同颜色的圆圈表示训练集的不同类别，不同颜色的十字表示测试集的不同类别，相同的颜色表示相同的类别），当括号中的参数 stress 小于 20%时，所获取的图像被认为数据集潜在结构的精确表示（stress 的大小用于度量所选取 MDS 算法的适合度，stress 值越小，该表示就越契合原数据的潜在结构）。从图中可以观察到，WED/WDTW 展现了较强的区分能力。换句话说，WED/WDTW 在增大类间方差的同时降低类内方差。尤其是对于数据集 Coffee，不同的训练集和测试集之间存在一条明显的分割

线。

表 5.4 分类错误率

Table 5.4 Summary of classification error rates

数据集	标准度量(k)		新的度量 (k, c)			
	ED	DTW	WED_{exp}	WED_{pow}	$WDTW_{exp}$	$WDTW_{pow}$
ItalyPower.	0.037(9)	0.049(7)	0.036(9,10)	**0.035(7,1)**	0.049(1,3)	0.047(7,1)
ECG200	0.100(3)	0.200*(3)	0.100(3,1)	**0.070(3,2)**	0.200*(3,1)	0.140(3,1)
Gun Point	0.087(1)	0.093(1)	**0.080(1,3)**	0.087(1,5)	0.093(1,1)	**0.080(1,10)**
MoteStrain	**0.121(1)**	0.145*(9)	0.122(1,5)	0.134(1,5)	0.137(9,2)	0.138(9,1)
SonyAIBOII	0.141(1)	0.169*(1)	0.136(1,2)	**0.132(1,1)**	0.165*(1,1)	0.170*(1,2)
CinC ECG.	**0.103(1)**	0.349*(1)	0.104(1,1)	**0.103(1,10)**	0.347*(1,2)	0.351*(1,2)
wafer	0.005(1)	0.020*(1)	**0.004(1,2)**	0.005(1,1)	0.020*(1,1)	0.025*(1,1)
Car	0.267(1)	0.267(1)	0.267(1,1)	**0.250(1,3)**	0.267(1,1)	0.283(1,5)
ECGFive.	0.203(1)	0.232*(1)	**0.190(1,2)**	**0.190(1,3)**	0.230*(1,3)	0.196 (1,5)
SwedishLeaf	0.211(1)	0.208(1)	0.213(1,1)	**0.199(1,1)**	0.205(1,2)	0.200(1,3)
Adiac	**0.389(1)**	0.396(1)	**0.389(1,10)**	0.394(1,5)	0.396(1,2)	0.396(1,3)
ChlorineCon.	0.350(1)	0.352(1)	**0.349(1,1)**	0.354(1,1)	0.353(1,2)	0.355(1,5)
Haptics	0.581(7)	0.571(3)	0.581(7,1)	**0.562(7,3)**	0.565(3,5)	0.575(5,3)
Conseason	0.224(3)	0.278*(9)	0.203(9,1)	**0.178(9,10)**	0.254*(5,1)	0.257*(7,5)
OliveOil	0.133(1)	0.133(1)	0.133(1,1)	0.133(1,1)	0.133(3,1)	0.133(1,3)
CBF	0.148*(1)	0.003(1)	0.144*(1,1)	0.109*(1,2)	0.003(1,2)	**0.000(3,2)**
FacesUCR	0.231*(1)	**0.095(1)**	0.231*(1,2)	0.221*(1,1)	**0.095(1,3)**	0.101(1,3)
Lighting2	0.230*(3)	0.131(1)	0.197*(3,1)	0.213*(1,1)	0.115(1,2)	**0.082(1,10)**
Plane	0.038*(1)	**0.000(1)**	0.038*(1,1)	0.038*(1,1)	**0.000(1,1)**	**0.000(1,1)**
Symbols	0.101*(1)	**0.050(1)**	0.101*(1,1)	0.098*(1,5)	**0.050(1,1)**	0.069*(1,3)
Trace	0.240*(1)	**0.000(1)**	0.230*(1,2)	0.250*(1,10)	**0.000(1,1)**	**0.000(1,1)**
TwoLeadECG	0.253*(1)	0.096(1)	0.252*(1,1)	0.253*(1,5)	**0.095(1,3)**	0.101(1,3)
CC	0.090*(3)	0.007(1)	0.080*(3,3)	0.103*(3,5)	**0.007(1,1)**	**0.007(1,2)**
BME	0.173*(1)	0.107*(1)	0.180*(1,1)	0.220*(1,1)	**0.053(3,1)**	**0.053(3,1)**
UMD	0.194*(1)	0.118*(1)	0.181*(1,2)	0.174*(1,1)	**0.007(1,2)**	0.014(1,1)
MALLAT	0.080*(3)	0.066(1)	0.080*(3,1)	0.072*(3,10)	0.067(1,1)	**0.060(1,1)**
TwoPatterns	0.093*(1)	**0.000(1)**	0.089*(1,5)	0.091*(1,5)	**0.000(1,1)**	**0.000(1,1)**
Beef	0.467(1)	0.500(1)	0.467(1,1)	0.467(1,1)	0.367(1,1)	**0.333(1,2)**
Coffee	0.179*(3)	0.179*(1)	0.179*(3,1)	0.179*(1,3)	**0.000(1,1)**	**0.000(1,1)**
FaceFour	0.216(1)	**0.170(1)**	0.216(1,1)	0.193(1,1)	**0.170(1,2)**	0.205(1,2)
Lighting7	0.397*(3)	0.247(5)	0.370*(3,1)	0.329(3,1)	0.233(5,1)	**0.219(5,2)**
MedicalImag.	0.316*(1)	0.263(1)	0.311*(1,1)	0.300*(1,1)	**0.262(1,2)**	0.263(1,2)
SonyAIBO	0.304(1)	0.275(1)	0.301(1,2)	0.300(1,5)	0.275(1,3)	**0.266(1,3)**
FISH	0.217(1)	**0.166(1)**	0.217(1,1)	0.200(1,5)	**0.166(1,1)**	0.171(1,5)
FaceAll	0.286*(1)	0.190*(5)	0.280*(1,10)	0.286*(1,1)	0.130(1,5)	**0.124(1,5)**
OSULeaf	0.483*(1)	0.409(1)	0.480*(1,3)	0.470(1,10)	0.405(1,10)	**0.397(1,10)**
WordsSyn.	0.382(1)	**0.351(1)**	0.381(1,1)	0.386(1,1)	0.354(1,1)	0.389(1,1)
yoga	0.170(1)	0.164(1)	0.169(1,5)	0.170(1,3)	0.162(1,1)	**0.161(1,3)**
InlineSkate	0.658*(1)	0.616(1)	0.658*(1,1)	0.649(1,10)	0.615(1,1)	**0.605(1,1)**
Trajectories	–	0.159*(1)	–	–	**0.068(1,1)**	**0.068(1,1)**

5.6 实例解析

此部分将通过实例解析，展示本章所提出的 WDTW/WED 模型在人造数据集以及实际应用中的有效性与可解释性。

5.6.1 人工数据集

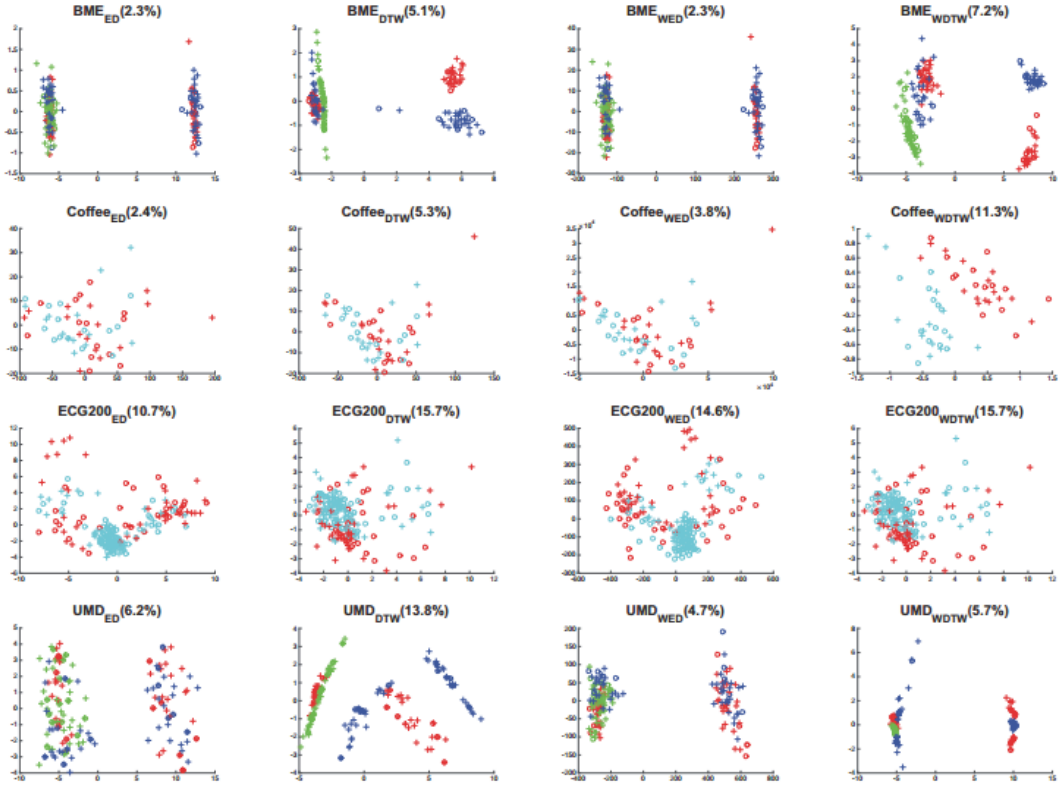

图 5.9 不同度量方法在不同数据集上的潜在结构

Figure 5.9 Underlying structures for different metrics on different datasets

为展示本章所提出算法的有效性，首先考虑两个人工数据集 BME 和 UMD，这两个数据集均由类间相似而类内不同的时间序列组成[113]。BME 数据集的长度为 128，分别由三个类别 *Begin*，*Middle* 和 *End* 组成。图 5.10 展示了 BME 中不同类别的时间序列，其中黑色曲线为代表性序列，灰色曲线为该类别的其他不同形状的

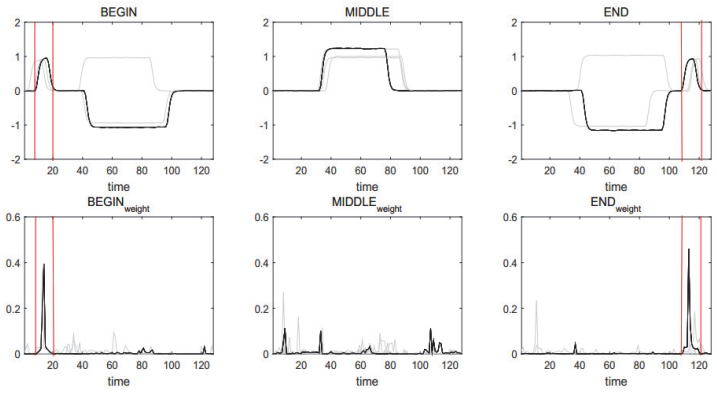

图 5.10 BME 数据集的辨别性 WDTW 权重

Figure 5.10 Discriminative WDTW weights for BME

序列。在 *Begin* 类别中，所有时间序列都有一个共同点，即开始阶段会出现一个小钟型子序列，而对于 *End* 类别，钟型子序列会出现在序列快结束时。但是在 *Beign* 或 *End* 序列中，大钟型子序列有可能朝上或者朝下出现，其与类别的相关性不大。另外，具有朝上方向大钟型子序列的 *Beign* 或 *End* 序列在整体上是与 *Middle* 序列相似的。

图 5.11 展示了数据集 UMD 不同类别的轮廓，它长度为 150，由三个相似的类别 *Up*，*Middle* 和 *Down* 组成。其相对于 BME 数据集更加复杂，主要区分性在于小钟型子序列的方向是朝上还是朝下的，与其出现位置或时间点并没有直接联系。从图 5.11 中可以观察到，小钟型子序列方向朝上时，无论其出现在序列开头还是末尾，均定义为 *Up* 类别；小钟型子序列方向朝下时，无论其出现在序列开头还是末尾，均定义为 *Down* 类别；当 *Up/Down* 中的大钟型子序列方向朝上时，它们与 *Middle* 序列具有全局相似性。

通过卡方优度拟合检验（无参假设为学习到的权重来自于均匀分布），图 5.10 中的黑色曲线标记的 $BEGIN_{weight}$ 和 END_{weight} 在 0.05 的显著性水平上拒绝了该假设（相应的，图 5.11 中的黑色曲线标记的 UP_{weight} 和 $DOWN_{weight}$ 在 0.05 的显著性水平上拒绝了该假设），所以本章所提出的 WDTW 模型能够发现辨别性的特征，如图 5.10 或 5.11 的红色区域所示，最具辨别性的特征刚好对应于小钟型子序列出现的地方。同时，图 5.10 和 5.11 中的 $MIDDLE_{weight}$ 接受了该假设，所以与其他两类相比，无论是 UMD 或 BME 数据集中的 *Middle* 类别，均没有明显的辨别性特征，与之前对数据集的描述相符。

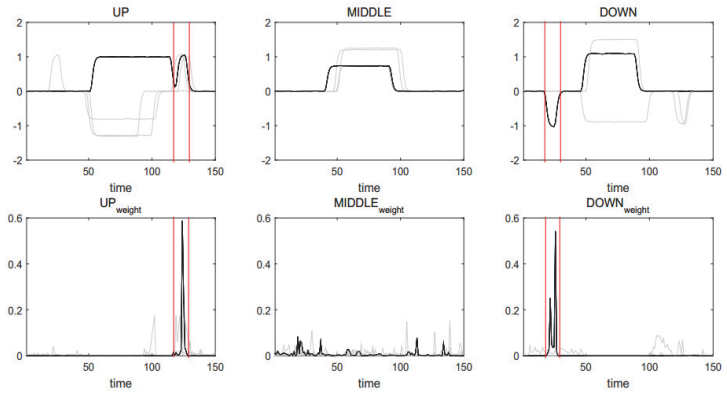

图 5.11 UMD 数据集的辨别性 WDTW 权重

Figure 5.11 Discriminative WDTW weights for UMD

基于 ED 或者 DTW 的 *k*-NN 分类器在该两个数据集上的分类错误率分别为 0.173 和 0.107（BME），0.194 和 0.118（UMD），而在同样的任务中，通过使用本章提出的辨别性 WDTW，BME 的错误率为 0.053，而 UMD 的错误率为 0.007，明

显优于基于 ED 或者 DTW 的 *k*-NN 分类器。

5.6.2 真实数据集

经典的 *Gun/NoGun* 数据集是一组根据动作捕捉而产生的时间序列，它是时间序列中最被广泛应用的数据集之一。此数据集包含 200 条实例，其中 50 条作为训练集，另外 150 条作为测试集，数据集中的每一条实例的长度均为 150（不包含类属性）。图 5.12 展示了该数据集中 *Gun* 与 *NoGun* 类别的代表性序列，其中 *Gun* 序列记录了表演者举起手枪并放回原位的动作序列，而 *NoGun* 序列让行动者做同样的动作，但手中并没有手枪。两者的主要区别在于，若表演者手中没枪，记录序列中就会出现如图 5.12 中红色方框中的类似于波谷的子序列。很明显，本章所提出的 WDTW 算法能够找到该经典数据集的辨别性子序列，所发现子序列也与时间序列 shapelets[49, 118]以及可解释性的序列规则[119]相符。

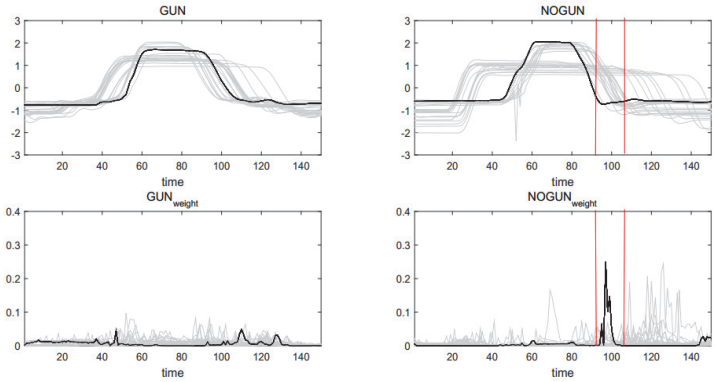

图 5.12 *Gun/NoGun* 数据集的辨别性 WDTW 权重

Figure 5.12 Discriminative WDTW weights for *Gun/NoGun*

Coffee 数据集是咖啡的一个光谱图集。在化学计量学中，它能够用于食物种类的分类，此任务能被广泛应用于食品安全和质量检测中。图 5.13 中所示即为咖啡的两个变种 *Arabica* 和 *Robusta* 的光谱图。此数据集包含 56 个实例，其中 28 个作为训练集，另外 28 个作为测试集，数据集中的每一个实例的长度都为 286。由于两种类别具有相似的全局特性，所以很难区分它们。但是，当放大图 5.13 中红色方框中的子序列时，可以很明显的发现，*Arabica* 的子序列的尾部具有明显的上升趋势，而 *Robusta* 的子序列的尾部保持下降趋势。基于 ED 或者 DTW 的 *k*-NN 分类器在该数据集上的分类错误率均为 0.179，而本章所提出方法的错误率为 0，意味着它能完全区分该序列，并明显优于基于标准度量的 *k*-NN。

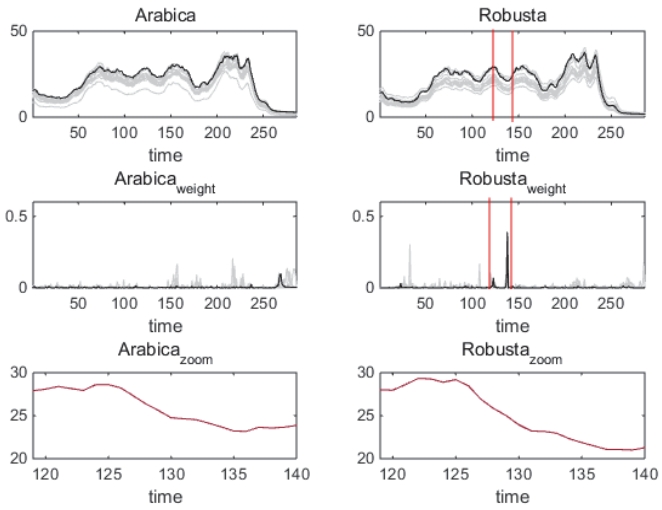

图 5.13 Coffee 数据集的辨别性 WDTW 权重

Figure 5.13 Discriminative WDTW weights for Coffee

对于时间序列数据集，它们并不需要完全按照时间顺序排列，逻辑上有序就足够了。所以，图像可以被转变为时间序列，然后应用时间序列分类方法处理图像分类问题。FaceAll 数据集就是将毕业学生的头像照片转换为时间序列的数据集，其主要任务是通过它们头部的轮廓来区分不同的面孔。该数据集共有 2250 个长度为 131 的实例，其中 560 个为训练集，其他为测试集，类值的个数为 14。基于 ED 或者 DTW 的 *k*-NN 分类器在该数据集上的分类错误率分别为 0.286 和 0.190，而本章多提出的 WDTW 算法得到了 0.124 的错误率，和其他错误率之间具有显著性差异

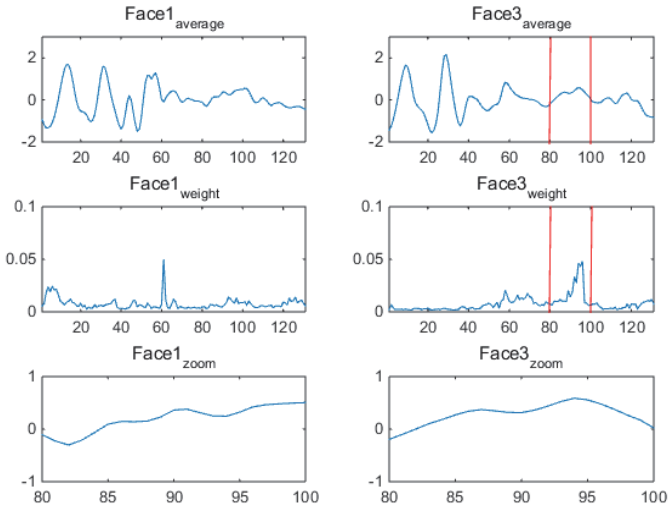

图 5.14 FaceAll 数据集的辨别性 WDTW 权重

Figure 5.14 Discriminative WDTW weights for FaceAll

（如表 5.4 所示）。为简洁起见，图 5.14 展示了两个全局相似的类别 *Face1* 和 *Face3*（两者均用其更具有代表性的均值序列表示），通过放大红色方框部分的子序列，可以很明显地发现两者之间的区别。

ECG200 数据集包含 200 个 ECG 数据，其中 100 个为训练集，另 100 个为测试集。每一条时间序列有 96 个度量值，并代表一次心跳反应。200 个数据集中的 133 个被标记为正常心跳（*normal*），其他 67 个被标记为不正常心跳（*abnormal*）。该数据集是从是从一条非常长的 ECG 数据上分割得到的，它模拟了医生从一条 ECG 数据记录上寻找出不正常心跳的过程[41]。本章所提出的 WED 算法在该数据集上得到了最优错误率 0.70。如图 5.15 所示，在时间序列的均值序列和权重的均值序列中，很容易发现 *normal* 与 *abnormal* 的最大不同在于，正常心跳数据在开始阶段会出现明显的转折点，而不正常心跳数据并不会出现（如红色方框区域所示）此例子也说明了本章所提出加权策略的可解释性。

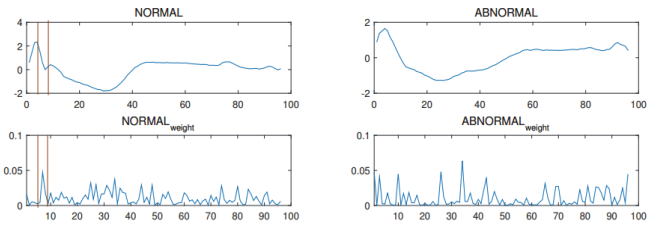

图 5.15 ECG200 数据集的辨别性 WDTW 权重

Figure 5.15 Discriminative WDTW weights for ECG200

5.6.3 多变量时间序列数据

Trajectories 数据集是一个多变量的，长度不等的手写字符集，它记录了某人在书写 20 个不同字母（*a*, *b*, *c*, *d*, *e*, *g*, *h*, *l*, *m*, *n*, *o*, *p*, *q*, *r*, *s*, *u*, *v*, *w*, *y*, *z*）时笔尖的行走轨迹。所有的实例均来自于同一个书写者，并且不同字母间具有明显的全局相似性。该数据集包含 2858 个不同长度的手写字母，每一个字母由 3 个不同维度的笔尖速率轨迹组成，其可视化表示根据速率随着时间变化的加和组成。基于 DTW 的 *k*-NN 分类器在该数据集上的最优分类错误率为 0.159，而本章多提出的 WDTW 模型所获取的最优分类错误率为 0.068。图 5.16 中展示了部分手写字符，从图中可以观察到，对于待分类字母“*e*”，基于 DTW 的度量容易将字母“*e*”与字母“*d*”，“*u*”和“*z*”混淆在一起，而 WDTW 能够很好的找到它们的最近邻。

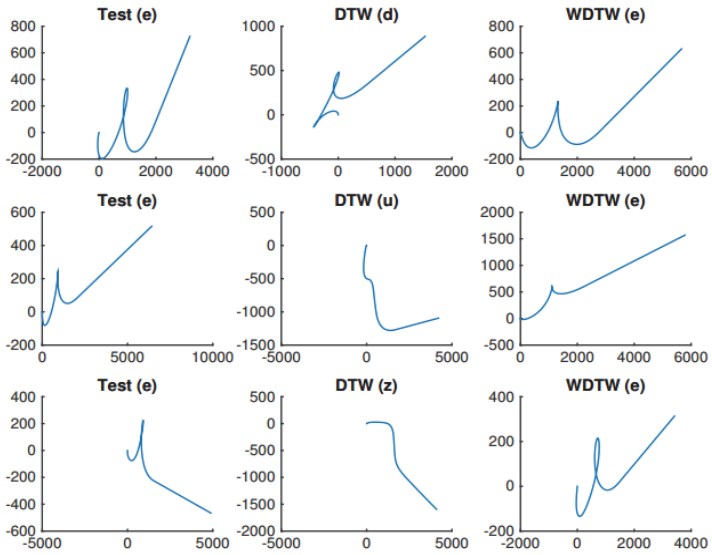

图 5.16 DTW 和 WDTW 算法得到的与字母“*e*”最近的邻居

Figure 5.16 The learned nearest neighbor by DTW and WDTW for character “*e*”

5.7 本章小结

本章介绍了一种基于局部加权方式的辨别式 DTW 对齐方式。它根据训练实例与正例集或负例集间的对齐点来发现最有辨别性的子序列，通过在多个具有挑战性的数据集上测试该算法（基于两种不同的加权函数）的性能，表明其无论是在单变量时间序列还是多变量时间序列上都具有较强的分类能力。另外，通过实例解析，阐明了基于 WDTW / WED 的 *k*-NN 分类器具有可解释性。但由于该模型需要迭代地学习权值，因此需要消耗大量的时间，未来的研究方向包括使用优化学习方法加速权重的学习过程，以及将该模型扩展到基于核函数的 *k*-NN 或支持向量机分类模型等等。

6 总结与展望

此部分将首先总结并集中比对本文提出的四种算法，接着对未来的研究工作进行进一步的展望。

6.1 研究工作总结

时间序列数据为高维的，实值型序列数据，并且具有不断更新的特点。由于其没有明确的特征，在时间序列数据上进行特征选择或建立可解释性的分类器是十分困难的。现阶段研究者提出的方法多集中于时间序列的表示与索引上，针对可解释性时间序列分类器的研究较为缺乏。本文主要针对如何建立可解释的时间序列分类器进行研究，分别提出了以下四种算法。本文所提出算法与基准分类器的综合比对如表 6.1 所示。

（1)提出了一种基于逻辑 shapelets 转换的时间序列分类器，用于提升 shapelets 的可解释力，并提升分类器的分类准确率。

实际应用中，虽然 shapelets 的发现过程是离线计算的，但根据 shapelets 递归建立决策树的过程仍会消耗大量的时间。为优化 shapelets 的选择过程，可以考虑首先选取最优的 k 个 shapelets，然后通过空间转换的思想，将原始的高维有次序的时间序列数据转换为只有 k 个属性并且不用考虑次序关系的实例，从而将时间序列分类问题映射到经典的分类场景，并应用传统的分类方法进行分类。由于时间序列规范化和子序列比对过程需要消耗大量时间，首先针对时间序列的规范化，通过采用充分统计量来提升其运算效率；其次，通过合取或析取多个 shapelets 组成逻辑 shapelets，并选取信息增益最大的 k 个进行数据转换，从而提升 shapelets 的可解释性。通过实验验证了所提出方法在多个分类器和数据集上的分类准确率由于基于 shapelets 转换的算法，对比于 1-NN 基准分类器，所提出算法在 SonyAIBORobotSurface 数据集上准确率的最大提升达 21.46%。

（2）提出一种简单有效的 shapelet 剪枝和覆盖方法，并将其扩展至逻辑 shapelets 的应用场景中。

虽然基于 shapelets 转换的思想能够提升分类准确率与可解释性，但通过研究观察发现，用于数据转换的 k 个 shapelets 之间存在较大相似性，并且 k 的具体数值是难以确定的。为解决以上两个问题，首先根据信息增益的定义，提出了一种 shapelets 剪枝策略，用于剪除与当前最好 shapelet 相似的 shapelets，同时降低了候选 shapelets 样本的数量；然后，为在选择最终 shapelets 的同时保证对实例的覆盖，

提出了 shapelet 覆盖的概念用于决定 k 的值。最后将 shapelets 剪枝和覆盖的概念推广至逻辑 shapelets 的剪枝与覆盖。由于所提出算法在保证对实例覆盖的同时允许更多辨别性 shapelets 参与到分类过程中，其在多个数据集以及多个分类器上的展示了较好的分类准确率。

（3）首次将关联式分类器应用于时间序列分类问题中，并提出了一种懒惰关联式分类器。

众所周知，关联规则能够简洁的表示知识，并且具有易于解释的优点。但是，关联规则的发现多应用于事务数据库中。为简洁有效的表示时间序列中的知识，本文首先应用 SAX 表示方法将数值型时间序列离散化为符号序列，然后利用考虑序列顺序的改进 CBA 算法来发现类序列规则；其次，为解决生成规则过多且无法保证规则对实例覆盖的问题，提出了一种懒惰式的类序列规则发现算法，同时采用了四种不同的规则评价方式来选取最终的类序列规则。通过实验和实例解析，阐述了关联式分类器在时间序列分类问题中的可解释性。

（4）提出了一种具有可解释性的 k-NN 分类器，并将其扩展至多变量时间序列分类问题中。

基于不同距离度量方式的 k-NN 分类器是否具有可解释性一直是一个有争议的话题。而本文通过实例解析，展示了所提出基于局部加权 DTW 的 k-NN 分类器的可解释性。首先，为寻找时间序列中的局部辨别性子序列，提出了一种新的时间序列加权模型，用于为每条时间序列的每一个特征值赋予不同的权值，进而根据权值发现辨别性子序列。其次，将所提出模型扩展至序列长度不相等的多变量时间序列分类问题，通过实例解析（比如字符识别）说明其有效性。另外，还讨论了加权 DTW 非相似性度量的特殊情况，加权 ED 度量在时间序列分类问题上的应用。通过实验说明，WED 能够较好地处理没有延迟或延迟较少的序列信号，而 WDTW 擅长处理具有较多延迟的时间序列。

由于算法运行效率等各方面的原因，本文所提出的四个算法在 7 个数据集上完成了实验，如表 6.1 所示。其中加粗的数值表示所对比方法中最优的结果。如果一个分类错误率与最优结果相比具有显著性差异，则用“*”标注它。很容易观察到，本文所提出算法的分类错误率优于基准分类器，除了 ItalyPowerDemand 数据集之外，最优的分类结果与其他分类器间具有显著性差异。根据没有免费的午餐理论[120]（no free lunch theorem），若训练数据包含了所有可能的状态，那么任何算法在此数据集上取得的效果是一致的。但是，由于通常得到的数据集是一个有偏差的样本，所以往往一个算法擅长处理某个子集，而不擅长处理另一个子集。从另一个角度来说，算法在某个评价指标上的提升，往往意味着其在其他方面做出了牺牲。比如，对于本文所提出的四种算法，他们均能提升分类准确率并具有可解释性，但其

在运行效率方面会有所下降。比如，寻找 shapelets 时需要不断遍历序列，发现关联规则时需要首先符号化时间序列，计算权重时需要找到每一个训练实例的正负子集等等。

表 6.1 本文所提出算法与基准分类器的集中比对（错误率）[4]

Table 6.1 Comparision of all the proposed algorithms with benchmarks (error rate)

Algorithm\Data	Sony.	SonyII.	TwoLe.	MoteS.	ItalyPower.	ECGFiv.	Cricket
1-NN（ED）	0.304*	0.141*	0.253*	0.121	0.045	0.203*	0.122*
1-NN（DTW）	0.275*	0.169*	0.096*	0.165*	0.050	0.232*	0.010
LG-1NN (ED)	0.090*	**0.107**	0.061*	**0.113**	0.068*	**0.000**	0.083*
Prune_ED	**0.053**	0.132*	**0.011**	0.140*	0.052	0.005*	0.020
Prune_DTW	0.118*	0.125	0.025*	0.165*	**0.042**	0.013*	**0.000**
laAll	0.178*	0.323*	0.224*	0.177*	0.056	0.134*	0.061*
WED_{exp}	0.301*	0.136*	0.252*	0.122	**0.042**	0.190*	0.051*
WED_{pow}	0.300*	0.132*	0.253*	0.134	**0.042**	0.190*	**0.000**
$WDTW_{exp}$	0.275*	0.165*	0.095*	0.137*	0.049	0.230*	**0.000**
$WDTW_{pow}$	0.266*	0.170*	0.101*	0.138*	0.050	0.196*	**0.000**

6.2 未来工作展望

本文针对可解释性时间序列分类器的模型构建与算法设计进行了系统的研究，提出了一些解决方案，但距离实际应用仍有较大的差距，所提出算法也有可供进一步改进的空间，今后可从以下几个方面做进一步的研究。

（1）时间序列符号化研究。除了 SAX 方法及其变种之外，现阶段研究者并没有提出其他有效地将实值型时间序列转换为符号序列的方法，所以如何将时间序列数据表示为符号序列，同时保持其原有特征，仍拥有着巨大的研究空间；另外，时间序列 motif 是指反复出现在较长时间序列中的子序列[121-125]，它的概念与关联规则发现中频繁模式[126]的概念相近，顾可考虑将时间序列 motif 的发现与关联规则的生成相结合，省略第 4 章中的离散化步骤，从而保留完整地时间序列信息，并直接在数值型时间序列上建立具有可解释性的关联式分类器。

（2）支持向量机[67]（Support Vector Machine，SVM）或核方法（kernel method）在时间序列分类上的研究。据作者所知，大部分应用于时间序列上的 SVM 方法忽略了原有数据中的时间顺序，运行效率较慢[67, 127]或可解释性较差[128-130]，并且基于 DTW 的核方法[131-134]在时间序列数据上的分类准确率较低，并不具有可解释性，针对上述缺点，可考虑使用基于 L1-regularization 的 SVM 进行特征选择[135, 136]并结合不同规范化方法[137]的线性 SVM[138]解决上述问题，希望其能在寻找出辨别性

[4] 数据集名词缩写，Sony.: SonyAIBORobotSurface, SonyII.: SonyAIBORobotSurfaceII, TwoLe.: TwoLeadECG, ItalyPower.: ItalyPowerDemand, ECGFiv.: ECGFiveDays

子序列的同时具有较高的分类准确率，并能够处理大规模数据集。另外，如何将核方法应用于时序关联规则的发现中，也是一个亟待解决的问题。

（3）针对时间序列的分类器融合策略。分类器融合策略的主要思想是将多样性融入所集成的分类器中。此处所谓的多样性包括：使用不同的分类算法训练每一个基分类器来构成一个异构的集成分类器；通过采样机制或直接复制实例为每一个基分类器提供不同的训练数据（Bagging 思想[139, 140]）；随机选择不同的属性来训练每一个分类器；通过重新评估训练数据的权重来修正每一个分类器（AdaBoost 思想[141]）等等。融合学习策略也已被应用于时间序列数据挖掘中，如时间序列森林[54]，集成灵活性与非灵活性距离度量的分类器[41]，以及集成不同表示方式进行转换[3]的分类器等。但是，现阶段并没有设计出综合考虑时域相似性，形状相似性和变化相似性的集成分类器，避免只在时域或形状相似性上建立分类器所引起的偏差，并根据不同的数据集和应用场景，为不同相似性的基分类器赋予不同的权重。

（4）时间序列数据的可视化研究。人类作为三维动物无法感知高维空间，但时间序列数据恰恰具有维度高的特点，为更好的认识和分析时间序列数据，需要使用线性或非线性降维方法，如流行学习方法 LLE[142, 143]，Isomap[144]以及 MDS 等等，在低维空间中展示时间序列数据，发现不同类别时间序列间的分布情况，从而能够更好的进行分类处理。

（5）回复式神经网络[145]（Recurrent Neural Network，RNN）在时间序列分类问题上的研究。RNN 在时间序列预测（或回归）问题上有着显著的效果[146]，但其在时间序列分类问题上的研究仍处于起步阶段[147]。虽然预测与分类任务之间有很大的不同，但由于预测任务能够很好地发现数据的内部结构和表示输入序列数据的主要特征（这些正是分类问题所关心的），因而可考虑将之应用于解决序列分类问题。

参考文献

[1] Cryer J D, Chan K-S, 潘红宇. 时间序列分析及应用: R 语言[M]. 北京: 机械工业出版社, 2011.

[2] Xing Z, Pei J, Keogh E. A brief survey on sequence classification[J]. ACM SIGKDD Explorations Newsletter, 2010, 12(1): 40-48.

[3] Bagnall A, Davis L M, Hills J, et al. Transformation based ensembles for time series classification[C]. //Proceeding of 2012 SIAM International Conference on Data Mining (SDM 2012), CA. Springer. 307-318.

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

[5] Mcgovern A, Rosendahl D H, Brown R A, et al. Identifying predictive multi-dimensional time series motifs: an application to severe weather prediction[J]. Data Mining and Knowledge Discovery, 2010, 22(1-2): 232-258.

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

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

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

[9] Lines J, Bagnall A, Caiger-Smith P, et al. Classification of household devices by electricity usage profiles[C]. //Proceeding of 12nd International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2011), Norwich, UK. Springer. 403-412.

[10] Chen Y, Keogh E, Hu B, et al. The UCR Time Series Classification Archive[EB/OL] www.cs.ucr.edu/~eamonn/time_series_data/.

[11] Lines J, Davis L M, Hills J, et al. A shapelet transform for time series classification[C]. //Proceeding of 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2012), Beijing. ACM Press. 289-297.

[12] Garreau D, Lajugie R, Arlot S, et al. Metric learning for temporal sequence alignment[M]. //Advances in neural information processing systems. 2014: 1817-1825.

[13] Fu T-C. A review on time series data mining[J]. Engineering Applications of Artificial Intelligence, 2011, 24(1): 164-181.

[14] Esling P, Agon C. Time-series data mining[J]. ACM Computing Surveys, 2012, 45(1): 1-34.

[15] Ratanamahatana C, Keogh E, Bagnall A J, et al. A novel bit level time series representation with implication of similarity search and clustering[M]. //Advances in Knowledge Discovery and Data Mining. Springer, 2005: 771-777.

[16] Agrawal R, Faloutsos C, Swami A. Efficient similarity search in sequence databases[C]. //Proceeding of 4th International Conference on Foundations of Data Organization and Algorithms (FODO 1993), Chicago, Illinois. Springer.

[17] Bagnall A, Janacek G J, Powell M. A likelihood ratio distance measure for the similarity between the fourier transform of time series[C]. //Proceeding of 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2005), Hanoi, Vietnam. Springer. 737-743.

[18] Chan K-P, Fu A-C. Efficient time series matching by wavelets[C]. //Proceeding of 15th International Conference on Data Engineering (ICDE 1999), Sydney, Australia. IEEE Computer Society Press. 126-133.

[19] Popivanov I, Miller R J. Similarity search over time-series data using wavelets[C]. //Proceeding of 18th International Conference on Data Engineering (ICDE 2002), San Jose, CA. IEEE Computer Society. 212-221.

[20] Liabotis I, Theodoulidis B, Saraaee M. Improving similarity search in time series using wavelets[J]. International Journal of Data Warehousing and Mining (IJDWM), 2006, 2(2): 55-81.

[21] Keogh E J, Pazzani M J. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback[C]. //Proceeding of 4th International Conference of Knowledge Discovery and Data Mining, NY. AAAI Press. 239-243.

[22] Keogh E, Chakrabarti K, Pazzani M, et al. Dimensionality reduction for fast similarity search in large time series databases[J]. Knowledge and information Systems, 2001, 3(3): 263-286.

[23] Keogh E, Chu S, Hart D, et al. An online algorithm for segmenting time series[C]. //Proceeding of 2001 IEEE International Conference on Data Mining (ICDM 2001), CA. IEEE Computer Society. 289-296.

[24] Lin J, Keogh E, Wei L, et al. Experiencing SAX: a novel symbolic representation of time series[J]. Data Mining and Knowledge Discovery, 2007, 15(2): 107-144.

[25] 钟清流, 蔡自兴. 基于统计特征的时序数据符号化算法[J]. 计算机学报, 2008, 31(10): 1857-1864.

[26] Bondu A, Boullé M, Cornuéjols A. Symbolic representation of time series: a hierarchical coclustering formalization[J]. Proceedings of AALTD 2015, 2015: 27.

[27] Malinowski S, Guyet T, Quiniou R, et al. 1d-sax: A novel symbolic representation for time series[M]. //Advances in Intelligent Data Analysis XII. Springer, 2013: 273-284.

[28] Camerra A, Palpanas T, Shieh J, et al. iSAX 2.0: Indexing and mining one billion time series[C]. //Proceeding of 2010 IEEE 10th International Conference on Data Mining (ICDM 2010). IEEE. 58-67.

[29] Shieh J, Keogh E. iSAX: disk-aware mining and indexing of massive time series datasets[J]. Data Mining and Knowledge Discovery, 2009, 19(1): 24-57.

[30] Azzouzi M, Nabney I T. Analysing time series structure with Hidden Markov Models[C]. //Proceeding of IEEE Conference on Neural Networks and Signal Processing. 402-408.

[31] Lin T-H, Kaminski N, Bar-Joseph Z. Alignment and classification of time series gene expression in clinical studies[J]. Bioinformatics, 2008, 24(13): i147-i155.

[32] Zhong S, Ghosh J. HMMs and coupled HMMs for multi-channel EEG classification[C]. //Proceeding of IEEE International Joint Conference on Neural Networks, Honolulu, Hawaii.

IEEE Service Center. 1254-1159.

[33] Wang P, Wang H, Wang W. Finding semantics in time series[C]. //Proceeding of 2011 ACM SIGMOD International Conference on Management of Data. ACM. 385-396.

[34] Yang B, Guo C, Jensen C S. Travel cost inference from sparse, spatio temporally correlated time series using markov models[J]. Proceedings of the VLDB Endowment, 2013, 6(9): 769-780.

[35] Kalpakis K, Gada D, Puttagunta V. Distance measures for effective clustering of ARIMA time-series[C]. //Proceeding of IEEE International Conference on Data Mining (ICDM 2001), CA. IEEE Computer Society. 273-280.

[36] Nanopoulos A, Alcock R, Manolopoulos Y. Feature-based classification of time-series data[J]. International Journal of Computer Research, 2001, 10(3): 49-61.

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

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

[39] Faloutsos C, Ranganathan M, Manolopoulos Y. Fast subsequence matching in time-series databases[C]. //Proceeding of 1994 ACM SIGMOD International Conference on Management of Data Minnesota. ACM Press. 419-429.

[40] Batista G E, Wang X, Keogh E J. A complexity-invariant distance measure for time series[C]. //Proceeding of the 11th SIAM Conference on Data Mining (SDM 2011), Mesa, Arizona. SIAM / Omnipress. 699-710.

[41] Buza K A. Fusion methods for time-series classification[D]. University of Hildesheim, Germany, 2011.

[42] Wang X, Mueen A, Ding H, et al. Experimental comparison of representation methods and distance measures for time series data[J]. Data Mining and Knowledge Discovery, 2013, 26(2): 275-309.

[43] Berndt D J, Clifford J. Using dynamic time warping to find patterns in time series[C]. //Proceeding of KDD workshop. 359-370.

[44] Keogh E, Ratanamahatana C A. Exact indexing of dynamic time warping[J]. Knowledge and information Systems, 2005, 7(3): 358-386.

[45] Fu A W-C, Keogh E, Lau L Y, et al. Scaling and time warping in time series querying[J]. The International Journal on Very Large Data Bases, 2008, 17(4): 899-921.

[46] Jeong Y-S, Jeong M K, Omitaomu O A. Weighted dynamic time warping for time series classification[J]. Pattern Recognition, 2011, 44(9): 2231-2240.

[47] Xi X, Keogh E, Shelton C, et al. Fast time series classification using numerosity reduction[C]. //Proceeding of 23rd International Conference on Machine Learning (ICML 2006). ACM. 1033-1040.

[48] 李正欣, 张凤鸣, 李克武, 张晓丰. 一种支持 DTW 距离的多元时间序列索引结构[J]. 软件学报, 2014, 25(03): 560-575.

[49] Ye L, Keogh E. Time series shapelets: a new primitive for data mining[C]. //Proceeding of 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD

2009), Paris, France. ACM Press. 947-955.

[50] Rakthanmanon T, Keogh E. Fast shapelets: a scalable algorithm for discovering time series shapelets[C]. //Proceeding of 13th SIAM International Conference on Data Mining (SDM 2013), Austin, Texas. SIAM Press. 668-676.

[51] Yamada Y, Suzuki E, Yokoi H, et al. Decision-tree induction from time-series data based on a standard-example split test[C]. //Proceeding of 20th International Conference (ICML 2003), Washington, USA. AAAI Press. 840-847.

[52] Balakrishnan S, Madigan D. Decision trees for functional variables[C]. //Proceeding of 2006 International Conference on Data Mining (ICDM 2006), Washington, DC. IEEE Computer Society. 798-802.

[53] Douzal-Chouakria A, Amblard C. Classification trees for time series[J]. Pattern Recognition, 2012, 45(3): 1076-1091.

[54] Deng H, Runger G, Tuv E, et al. A time series forest for classification and feature extraction[J]. Information Sciences, 2013, 239: 142-153.

[55] Breiman L. Random forests[J]. Machine learning, 2001, 45(1): 5-32.

[56] Hills J, Lines J, Baranauskas E, et al. Classification of time series by shapelet transformation[J]. Data Mining and Knowledge Discovery, 2014, 28(4): 851-881.

[57] 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[J]. 软件学报, 2015, 26(9): 2311-2325.

[58] 原继东, 王志海, 韩萌, 孙艳歌. 基于逻辑 shapelets 转换的时间序列分类算法[J]. 计算机学报, 2015, 38(7): 1448-1459.

[59] Povinelli R J, Johnson M T, Lindgren A C, et al. Time series classification using Gaussian mixture models of reconstructed phase spaces[J]. IEEE Transactions on Knowledge and Data Engineering, 2004, 16(6): 779-783.

[60] Deng K, Moore A W, Nechyba M C. Learning to recognize time series: combining arma models with memory-based learning[C]. //Proceeding of IEEE International Symposium on Computational Intelligence in Robotics and Automation. IEEE Press. 246-251.

[61] 冯玉才, 蒋涛, 李国徽, 朱虹. 高效时序相似搜索技术[J]. 计算机学报, 2009, 32(11): 2107-2122.

[62] Batal I, Sacchi L, Bellazzi R, et al. Multivariate time series classification with temporal abstractions[C]. //Proceeding of 22nd International FLAIRS Conference.

[63] Baydogan M G, Runger G. Learning a symbolic representation for multivariate time series classification[J]. Data Mining and Knowledge Discovery, 2015, 29(2): 400-422.

[64] Wistuba M, Grabocka J, Schmidt-Thieme L. Ultra-fast shapelets for time series classification[J]. arXiv preprint arXiv:150305018, 2015.

[65] Chouakria-Douzal A. Compression technique preserving correlations of a multivariate temporal sequence[M]. //Advances in Intelligent Data Analysis V. Springer, 2003: 566-577.

[66] 李航. 统计学习方法[M]. 北京: 清华大学出版社, 2012.

[67] Chang C-C, Lin C-J. LIBSVM: a library for support vector machines[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2011, 2(3): 27.

[68] Santosh K C, Lamiroy B, Wendling L. DTW–radon-based shape descriptor for pattern recognition[J]. International Journal of Pattern Recognition and Artificial Intelligence, 2013,

27(03): 1350008.

[69] Do C-T, Douzal-Chouakria A, Marié S, et al. Multiple metric learning for large margin kNN classification of time series[C]. //Proceeding of 23rd European Signal Processing Conference (EUSIPCO). IEEE. 2346-2350.

[70] Kulis B. Metric learning: a survey[J]. Foundations and Trends in Machine Learning, 2012, 5(4): 287-364.

[71] Lhermitte S, Verbesselt J, Verstraeten W W, et al. A comparison of time series similarity measures for classification and change detection of ecosystem dynamics[J]. Remote Sensing of Environment, 2011, 115(12): 3129-3152.

[72] Grabocka J, Schilling N, Wistuba M, et al. Learning time-series shapelets[C]. //Proceeding of 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2014). ACM. 392-401.

[73] Gardner W A. Learning characteristics of stochastic-gradient-descent algorithms: A general study, analysis, and critique[J]. Signal Processing, 1984, 6(2): 113-133.

[74] Zakaria J, Mueen A, Keogh E. Clustering time series using unsupervised-shapelets[C]. //Proceeding of 12nd IEEE International Conference on Data Mining (ICDM 2012), Brussels, Belgium, 2012. IEEE Computer Society. 785-794.

[75] Xing Z, Pei J, Philip S Y. Early classification on time series[J]. Knowledge and information Systems, 2012, 31(1): 105-127.

[76] Xing Z, Pei J, Philip S Y, et al. Extracting interpretable features for early classification on time series[C]. //Proceeding of 11th SIAM International Conference on Data Mining (SDM 2011), Mesa, Arizona. SIAM Press. 247-258.

[77] Sakurai Y, Papadimitriou S, Faloutsos C. Braid: stream mining through group lag correlations[C]. //Proceeding of ACM SIGMOD International Conference on Management of Data, Maryland, USA. ACM Press. 599-610.

[78] Liaw A, Wiener M. Classification and regression by randomforest[J]. R news, 2002, 2(3): 18-22.

[79] Kuncheva L I, Rodríguez J J. An experimental study on rotation forest ensembles[M]. //Multiple Classifier Systems. Springer, 2007: 459-468.

[80] Rodriguez J J, Kuncheva L I, Alonso C J. Rotation forest: a new classifier ensemble method[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006, 28(10): 1619-1630.

[81] Friedman N, Geiger D, Goldszmidt M. Bayesian network classifiers[J]. Machine learning, 1997, 29(2-3): 131-163.

[82] Hall M, Frank E, Holmes G, et al. The WEKA data mining software: an update[J]. ACM SIGKDD Explorations Newsletter, 2009, 11(1): 10-18.

[83] Cheng H, Yan X, Han J, et al. Discriminative frequent pattern analysis for effective classification[C]. //Proceeding of 23rd International Conference on Data Engineering (ICDE 2007), Istanbul, Turkey. IEEE Press. 169-178.

[84] Keogh E, Chakrabarti K, Pazzani M, et al. Locally adaptive dimensionality reduction for indexing large time series databases[C]. //Proceeding of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2001), Santa Barbara, CA. ACM Press. 151-162.

[85] Cong G, Tan K-L, Tung A K, et al. Mining top-k covering rule groups for gene expression data[C]. //Proceeding of ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland. ACM Press. 670-681.

[86] Cheng H, Yan X, Han J, et al. Direct discriminative pattern mining for effective classification[C]. //Proceeding of 24th International Conference on Data Engineering (ICDE 2008), Cancun, Mexico. IEEE Press. 169-178.

[87] Li W, Han J, Pei J. CMAR: accurate and efficient classification based on multiple class-association rules[C]. //Proceeding of 2001 IEEE International Conference on Data Mining, San Jose, CA. IEEE Computer Society Press. 369-376.

[88] Liu B, Hsu W, Ma Y. Integrating classification and association rule mining[C]. //Proceeding of 4th International Conference on Knowledge Discovery and Data Mining (KDD 1998), New York AAAI Press.

[89] Veloso A, Meira W, Zaki M J. Lazy associative classification[C]. //Proceeding of 6th International Conference on Data Mining (ICDM 2006), Washington, DC. IEEE Computer Society. 645-654.

[90] Wang J, Karypis G. HARMONY: efficiently mining the best rules for classification[C]. //Proceeding of 5th SIAM International Conference on Data Mining (SDM 2005) Newport Beach, CA. Springer. 205-216.

[91] Yin X, Han J. CPAR: classification based on predictive association rules[C]. //Proceeding of SIAM International Conference on Data Mining (SDM 2003), San Francisco, CA. SIAM Press. 369-376.

[92] Hu M, Liu B. Opinion feature extraction using class sequential rules[C]. //Proceeding of AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs. 61-66.

[93] Deshpande M, Kuramochi M, Wale N, et al. Frequent substructure-based approaches for classifying chemical compounds[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(8): 1036-1050.

[94] Yan X, Han J. gspan: graph-based substructure pattern mining[C]. //Proceeding of 2002 IEEE International Conference on Data Mining (ICDM 2002), Maebashi, Japan. IEEE Computer Society Press. 721-724.

[95] Ting J, Fu T-C, Chung F-L. Mining of stock data: intra-and inter-stock pattern associative classification[C]. //Proceeding of 2006 International Conference on Data Mining (ICDM 2006), Washington, DC. IEEE Computer Society. 30-36.

[96] Das G, Lin K-I, Mannila H, et al. Rule discovery from time series[C]. //Proceeding of 4th International Conference on Knowledge Discovery and Data Mining (KDD 1998), NY. AAAI Press. 16-22.

[97] Quinlan J R. Induction of decision trees[J]. Machine learning, 1986, 1(1): 81-106.

[98] Quinlan J R. C4.5: programs for machine learning[M]. City: Elsevier, 1992.

[99] Arici T, Celebi S, Aydin A S, et al. Robust gesture recognition using feature pre-processing and weighted dynamic time warping[J]. Multimedia Tools and Applications, 2014, 72(3): 3045-3062.

[100] Reyes M, Domínguez G, Escalera S. Feature weighting in dynamic timewarping for gesture recognition in depth data[C]. //Proceeding of 2011 IEEE International Conference on Computer

Vision Workshops (ICCV Workshops). IEEE. 1182-1188.

[101] Zhou F, De La Torre F, Hodgins J K. Hierarchical aligned cluster analysis for temporal clustering of human motion[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, 35(3): 582-596.

[102] Zhou F, Torre F. Canonical time warping for alignment of human behavior[C]. //Proceeding of Advances in neural information processing systems. 2286-2294.

[103] Rath T M, Manmatha R. Word image matching using dynamic time warping[C]. //Proceeding of Computer Vision and Pattern Recognition (CVPR 2003). IEEE Computer Society. I521-527.

[104] Adams N H, Bartsch M A, Shifrin J B, et al. Time series alignment for music information retrieval[J]. Ann Arbor, 2004, 1001: 48109-42110.

[105] Sakoe H, Chiba S. Dynamic programming algorithm optimization for spoken word recognition[J]. IEEE Transactions on Acoustics, Speech and Signal Processing, 1978, 26(1): 43-49.

[106] Zhang X-L, Luo Z-G, Li M. Merge-weighted dynamic time warping for speech recognition[J]. Journal of Computer Science and Technology, 2014, 29(6): 1072-1082.

[107] Aach J, Church G M. Aligning gene expression time series with time warping algorithms[J]. Bioinformatics, 2001, 17(6): 495-508.

[108] Itakura F. Minimum prediction residual principle applied to speech recognition[J]. IEEE Transactions on Acoustics, Speech and Signal Processing, 1975, 23(1): 67-72.

[109] Keogh E J, Pazzani M J. Derivative dynamic time warping[C]. //Proceeding of SIAM International Conference on Data Mining (SDM 2001). SIAM. 5-7.

[110] Petitjean F, Forestier G, Webb G, et al. Dynamic time warping averaging of time series allows faster and more accurate classification[C]. //Proceeding of 2014 IEEE International Conference on Data Mining (ICDM 2014). 470-479.

[111] Salvador S, Chan P. Toward accurate dynamic time warping in linear time and space[J]. Intelligent Data Analysis, 2007, 11(5): 561-580.

[112] Celebi S, Aydin A S, Temiz T T, et al. Gesture recognition using skeleton data with weighted dynamic time warping[C]. //Proceeding of VISAPP. 620-625.

[113] Frambourg C, Douzal-Chouakria A, Gaussier E. Learning multiple temporal matching for time series classification[M]. //Advances in Intelligent Data Analysis XII. Springer, 2013: 198-209.

[114] Giorgino T. Computing and visualizing dynamic time warping alignments in R: the dtw package[J]. Journal of statistical Software, 2009, 31(7): 1-24.

[115] Kruskal J B. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis[J]. Psychometrika, 1964, 29(1): 1-27.

[116] Weinberger K Q, Saul L K. Distance metric learning for large margin nearest neighbor classification[J]. The Journal of Machine Learning Research, 2009, 10: 207-244.

[117] Dietterich T G. Statistical tests for comparing supervised classification learning algorithms[J]. Oregon State University Technical Report, 1996, 1: 1-24.

[118] Yuan J-D, Wang Z-H, Han M. A discriminative shapelets transformation for time series classification[J]. International Journal of Pattern Recognition and Artificial Intelligence, 2014, 28(6): 1450014.

[119] Yuan J, Wang Z, Han M, et al. A lazy associative classifier for time series[J]. Intelligent Data

Analysis, 2015, 19(5): 983-1002.

[120] Wolpert D H. What the no free lunch theorems really mean; how to improve search algorithms[M]. //Santa fe institute working paper. 2012: 12.

[121] Begum N, Keogh E. Rare time series motif discovery from unbounded streams[J]. Proceedings of the VLDB Endowment, 2014, 8(2): 149-160.

[122] Chiu B, Keogh E, Lonardi S. Probabilistic discovery of time series motifs[C]. //Proceeding of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2003), Washington. DC. ACM Press. 493-498.

[123] Lin J, Keogh E, Lonardi S, et al. Finding motifs in time series[C]. //Proceeding of 2nd Workshop on Temporal Data Mining at KDD, Edmonton, Alberta, Canada. ACM Press. 53-68.

[124] Mueen A, Keogh E, Bigdely-Shamlo N. Finding time series motifs in disk-resident data[C]. //Proceeding of 9th IEEE International Conference on Data Mining (ICDM 2009) Miami, Florida, USA. IEEE Computer Society. 367-376.

[125] Mueen A, Keogh E J, Zhu Q, et al. Exact discovery of time series motifs[C]. //Proceeding of 9th SIAM International Conference on Data Mining (SDM 2009). SIAM. 473-484.

[126] Han J, Kamber M, Pei J. Data mining: concepts and techniques[M]. City: Elsevier, 2011.

[127] Rodríguez J J, Alonso C J, Maestro J A. Support vector machines of interval-based features for time series classification[J]. Knowledge-Based Systems, 2005, 18(4): 171-178.

[128] Chen H, Tang F, Tino P, et al. Model-based kernel for efficient time series analysis[C]. //Proceeding of Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2013). ACM. 392-400.

[129] Gudmundsson S, Runarsson T P, Sigurdsson S. Support vector machines and dynamic time warping for time series[C]. //Proceeding of IEEE International Joint Conference on Neural Networks (IJCNN 2008). IEEE. 2772-2776.

[130] Lei H, Sun B. A study on the dynamic time warping in kernel machines[C]. //Proceeding of 3rd International IEEE Conference on Signal-Image Technologies and Internet-Based System (SITIS 07) IEEE. 839-845.

[131] Bahlmann C, Haasdonk B, Burkhardt H. Online handwriting recognition with support vector machines-a kernel approach[C]. //Proceeding of 8th international workshop on frontiers in handwriting recognition. IEEE. 49-54.

[132] Cuturi M. Fast global alignment kernels[C]. //Proceeding of Proceedings of the 28th International Conference on Machine Learning (ICML 2011). 929-936.

[133] Cuturi M, Vert J P, Birkenes O, et al. A kernel for time series based on global alignments[C]. //Proceeding of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007) IEEE. II-413-II-416.

[134] Noma H S K-I, Shimodaira K. Dynamic time-alignment kernel in support vector machine[M]. //Advances in neural information processing systems. 2002: 921.

[135] Bradley P S, Mangasarian O L. Feature selection via concave minimization and support vector machines[C]. //Proceeding of 15th International Conference on Machine Learning (ICML 1998). 82-90.

[136] Ng A Y. Feature selection, L1 vs. L2 regularization, and rotational invariance[C]. //Proceeding of 21st International Conference on Machine Learning (ICML 2004). ACM. 78.

[137] Pedroso J P, Murata N. Support vector machines with different norms: motivation, formulations and results[J]. Pattern recognition letters, 2001, 22(12): 1263-1272.

[138] Fan R-E, Chang K-W, Hsieh C-J, et al. LIBLINEAR: A library for large linear classification[J]. The Journal of Machine Learning Research, 2008, 9: 1871-1874.

[139] Breiman L. Bagging predictors[J]. Machine learning, 1996, 24(2): 123-140.

[140] Quinlan J R. Bagging, boosting, and C4.5[C]. //Proceeding of 14th National Conference on Artificial Intelligence. 725-730.

[141] Schapire R E, Singer Y. Improved boosting algorithms using confidence-rated predictions[J]. Machine learning, 1999, 37(3): 297-336.

[142] Roweis S T, Saul L K. Nonlinear dimensionality reduction by locally linear embedding[J]. Science, 2000, 290(5500): 2323-2326.

[143] Seung H S, Lee D D. The manifold ways of perception[J]. Science, 2000, 290(5500): 2268-2269.

[144] Tenenbaum J B, De Silva V, Langford J C. A global geometric framework for nonlinear dimensionality reduction[J]. Science, 2000, 290(5500): 2319-2323.

[145] Connor J T, Martin R D, Atlas L E. Recurrent neural networks and robust time series prediction[J]. IEEE transactions on neural networks, 1994, 5(2): 240-254.

[146] Längkvist M, Karlsson L, Loutfi A. A review of unsupervised feature learning and deep learning for time-series modeling[J]. Pattern recognition letters, 2014, 42: 11-24.

[147] Hüsken M, Stagge P. Recurrent neural networks for time series classification[J]. Neurocomputing, 2003, 50: 223-235.

作者简历及攻读博士学位期间取得的研究成果

一、作者简历

原继东，男，出生于 1989 年，河南省焦作市武陟县人，2010 年本科毕业于东北大学计算机科学与技术专业，2012 年硕士毕业于北京交通大学计算机科学与技术专业，同年开始攻读博士研究生。2015 年 3 月至 2016 年 3 月在法国约瑟夫傅里叶大学（Université Joseph Fourier，UJF）访问交流一年。

二、发表论文

[1] Jidong Yuan, Zhihai Wang, Meng Han and Yange Sun. A lazy associative classifier for time series. Intelligent Data Analysis, 2015, 19 (5). pp. 983-1002. (SCI WOS: 000361555000003; EI 索引号: 20153801287343)

[2] 原继东, 王志海, 韩萌, 游洋. 基于逻辑 Shapelets 转换的时间序列分类算法. 计算机学报, 2015，38 (7). pp.1448-1459.(EI 索引号: 20153101095313)

[3] 原继东, 王志海, 韩萌. 基于 Shapelets 剪枝和覆盖的时间序列分类算法. 软件学报, 2015, 26 (9). pp. 2311-2325. (EI 索引号: 20154101350196)

[4] 原继东, 王志海. 时间序列的表示与分类算法综述. 计算机科学, 2015, 42 (3). pp.1-7. (CSCD 核心期刊)

[5] Jidong Yuan, Zhihai Wang, Meng Han. A discriminative shapelets transformation for time series classification. International Journal of Pattern Recognition and Artificial Intelligence, 2014, 28(6). pp. 1450014:1-28. (SCI WOS:000343997500002; EI 索引号: 20144900297678)

[6] 原继东, 王志海, 韩萌. 时间序列关联式分类算法. 中国第五届数据挖掘会议 (CCDM 2014).

[7] Jidong Yuan, Zhihai Wang, Meng Han, Yashu Liu. Efficiently using different arrays in mining access patterns. Journal of Computational Information Systems, 2013, 9 (5). pp. 1791-1798. (EI 索引号: 20131616214511)

[8] Jidong Yuan, Zhihai Wang, Ahlame Douzal and Gustavo E.A.P.A. Batista. Locally-aware weighted dynamic time warping for an efficient time series nearest neighbor classification. The 16th IEEE International Conference on Data Mining (ICDM 2016), 2016. (Submitted)

[9] Meng Han, Zhihai Wang, Jidong Yuan. Mining closed and multi-supports-based sequential pattern in high dimensional dataset. International Arab Journal of Information Technology, 2015, 12 (4). pp. 360-369. (SCI WOS: 000359894600007)

[10] 韩萌, 王志海, 原继东. 一种基于时间衰减模型的数据流闭合模式挖掘方法. 计算机学报, 2015, 38 (7). pp. 1473-1483. (EI 索引号: 20153101095315)

[11] 韩萌, 王志海, 原继东. 基于高斯函数的衰减因子设置方法研究. 计算机研究与发展, 2015. 52(12). pp. 2834-2843. (EI 索引号: 20160501878299)

[12] Meng Han, Zhihai Wang, Jidong Yuan. Closed sequential pattern mining in high dimensional sequences. Journal of Software, 2013, 8 (6). pp. 1368-1373. (EI 索引号: 20132416421548)

[13] Meng Han, Zhihai Wang, Jidong Yuan. Mining constraint based sequential patterns and rules on restaurant recommendation system. Journal of Computational Information Systems, 2013, 9 (10). pp. 3901-3908. (EI 索引号: 20132416422420)

[14] Meng Han, Zhihai Wang, Jidong Yuan. Efficient method for mining patterns from highly similar

and dense database based on prefix-frequent-items. Journal of Software, 2014, 9 (8). pp. 2080-2086.

[15] Yange Sun, Zhihai Wang, Haiyang Liu, Chao Du, and Jidong Yuan. Online ensemble using adaptive windowing for data streams with concept drift. International Journal of Distributed Sensor Networks, 2016, 6(5). pp. 1-9. (SCI WOS: 000377352300001)

三、参与科研项目

[1] 中央高校基本科研业务费－时间序列的表示和分类算法研究（项目编号：2014YJS032）

[2] 中央高校基本科研业务费－高维时间序列的降维及分类算法研究（项目编号：2015YJS049）

[3] 北京市自然科学基金－多标记依赖关系发现方法及其在自动标签推荐中的应用研究（项目编号：4142042）

[4] 国家公派留学基金－Learning Metrics for Time Series Classification

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京交通大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：　　签字日期：　　年　　月　　日

学位论文数据集

表 1.1： 数据集页

关键词*	密级*	中图分类号	UDC	论文资助
时间序列，分类，可解释性	公开	TP391		北京市自然科学基金资助（4142042）
学位授予单位名称*		学位授予单位代码*	学位类别*	学位级别*
北京交通大学		10004	工学	博士
论文题名*		并列题名		论文语种*
时间序列分类算法研究				中文
作者姓名*	原继东		学号*	12112078
培养单位名称*		培养单位代码*	培养单位地址	邮编
北京交通大学		10004	北京市海淀区西直门外上园村 3 号	100044
学科专业*		研究方向*	学制*	学位授予年*
计算机科学与技术		数据挖掘	4	2016
论文提交日期*	2016.09			
导师姓名*	王志海		职称*	教授
评阅人	答辩委员会主席*		答辩委员会成员	
	史忠植		王树良，孙永奇，黄雅平，郎丛妍	
电子版论文提交格式 文本（√） 图像（ ） 视频（ ） 音频（ ） 多媒体（ ） 其他（ ） 推荐格式：application/msword；application/pdf				
电子版论文出版（发布）者		电子版论文出版（发布）地		权限声明
论文总页数*	113			
共 33 项，其中带*为必填数据，为 21 项。				