软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(2):799−816 [doi: 10.13328/j.cnki.jos.007481] [CSTR: 32375.14.jos.007481]

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

知识增强的时间序列异常检测算法自动选择*

梁志宇[1], 蔡东芮[1], 梁 晨[1], 梁 峥[1], 王宏志[1], 郑 博[2]

[1](哈尔滨工业大学 计算学部, 黑龙江 哈尔滨 150001)

[2](北京诺司时空科技有限公司, 北京 100020)

通信作者: 王宏志, E-mail: wangzh@hit.edu.cn

摘 要: 时间序列异常检测技术在许多实际应用中发挥着重要作用. 例如, 云原生数据库系统通过监测关键指标(如 CPU 和内存使用情况) 实现系统故障的及时识别. 尽管近年来已经提出了许多先进的时间序列异常检测算法, 但研究表明, 在异常检测准确率方面, 不同算法擅于应对不同的应用场景, 没有通用的最佳方法. 因此, 为了实现更高的异常检测准确率, 研究如何基于不同场景的数据特征自动选择最佳时间序列异常检测算法的问题尤为重要. 现有方法通常基于时间序列分类 (TSC) 技术来解决这一问题. 实现方法是利用历史任务积累的数据, 以时间序列为输入、对应的最准确异常检测算法为输出训练分类器, 从而预测未知时间序列的最佳异常检测算法. 尽管这类基于 TSC 的解决方案能有效提高异常检测准确率, 但现有的标准 TSC 算法未能充分利用来自异常检测历史任务的知识. 为弥补这一缺陷, 提出一个知识增强的时间序列异常检测框架. 在训练 TSC 模型时, 不仅使用现有方法普遍采用的、代表每个历史时间序列最佳检测算法的硬标签, 还利用历史数据上所有候选算法的准确率来估计输入时间序列的类别分布, 将其作为软标签来为算法选择器 (即 TSC 模型) 提供更多关于异常检测算法之间相互关系的知识. 与此同时, 设计了一个外部知识融合模块, 可以灵活地将各类外部知识 (例如时间序列的应用领域及数据与异常特点的描述) 融入 TSC 模型中. 所提方法能够作为插件无缝集成到任意架构的 TSC 模型中, 提高其在异常检测算法选择方面的性能. 在多种类型的时间序列数据集上进行大量实验, 验证所提方法的有效性.

关键词: 时间序列异常检测; 自动算法选择; 时序数据分析; 知识增强型系统; 面向数据库的人工智能

中图法分类号: TP311

中文引用格式: 梁志宇, 蔡东芮, 梁晨, 梁峥, 王宏志, 郑博. 知识增强的时间序列异常检测算法自动选择. 软件学报, 2026, 37(2): 799–816. http://www.jos.org.cn/1000-9825/7481.htm

英文引用格式: Liang ZY, Cai DR, Liang C, Liang Z, Wang HZ, Zheng B. Knowledge-enhanced Automatic Selection for Time Series Anomaly Detection Algorithm. Ruan Jian Xue Bao/Journal of Software, 2026, 37(2): 799–816 (in Chinese). http://www.jos.org.cn/1000-9825/7481.htm

Knowledge-enhanced Automatic Selection for Time Series Anomaly Detection Algorithm

LIANG Zhi-Yu[1], CAI Dong-Rui[1], LIANG Chen[1], LIANG Zheng[1], WANG Hong-Zhi[1], ZHENG Bo[2]

[1](Faculty of Computing, Harbin Institute of Technology, Harbin 150001, China)

[2](CnosDB Inc., Beijing 100020, China)

Abstract: Time series anomaly detection plays an important role in many real-world applications, such as monitoring key metrics (e.g., CPU and memory usage) in cloud-native database systems to detect system failures timely. Although many advanced time series anomaly detection algorithms have been proposed in recent years, it has been shown that different algorithms excel in different application scenarios in terms of anomaly detection accuracy, and there is no universally optimal method. Therefore, studying the problem of automatically selecting the most suitable time series anomaly detection algorithm based on the data characteristics of various scenarios is crucial to

* 基金项目: 智能电网重大专项 (2030) (2024ZD0802900)
收稿时间: 2024-06-06; 修改时间: 2025-04-28; 采用时间: 2025-06-02; jos 在线出版时间: 2025-12-03
CNKI 网络首发时间: 2025-12-04

achieving higher detection accuracy. Existing studies typically address this problem using time series classification (TSC) techniques, training a classifier on data from historical tasks, where the input is a time series, and the output is the predicted most accurate anomaly detection algorithm for that time series. Although TSC-based solutions improve detection accuracy, existing standard TSC algorithms fail to fully utilize the knowledge from historical anomaly detection tasks. This study proposes a knowledge-enhanced time series anomaly detection framework. Specifically, in addition to training the TSC model with hard labels that represent the best detection algorithm for each historical time series, the accuracy of all candidate algorithms evaluated on historical data is used to estimate the class distribution of the input time series. The distribution is treated as a soft label, providing the algorithm selector (i.e., the TSC model) with more knowledge about the relationships between the anomaly detection algorithms. Meanwhile, a module is designed to flexibly integrate various types of external knowledge (e.g., descriptions of the domain, characteristics of time series, and anomalies) into the TSC model. The proposed method is designed as a plugin that can be seamlessly integrated into any TSC model to enhance its performance in anomaly detection algorithm selection, regardless of the model architecture. Extensive experiments on various types of time series datasets validate the effectiveness of this approach.

Key words: time series anomaly detection; automatic algorithm selection; time-series data analysis; knowledge-enhanced system; artificial intelligence for database

时间序列异常检测[1]技术是诸多实际应用中的重要支撑技术. 例如在云原生数据库系统监控运维中[2], 通过时间序列异常检测算法对持续采集的时间序列数据, 如 CPU、内存和磁盘等资源的利用率、数据库调用情况等监控指标进行分析, 检测其中的异常值, 能帮助运维人员及早地发现复杂庞大的云原生数据库系统中可能存在的故障, 降低系统故障造成的损失并节约宝贵的人力和时间成本.

尽管近年来已有众多先进的时间序列异常检测算法被相继提出[1,3], 但大量研究表明, 由于时间序列数据特征的异构性和异常类型的多样性, 在检测准确率方面, 当前各类时间序列异常检测算法在不同应用场景 (如监控指标或业务负载) 上各有所长, 尚无通用的最佳方法[1,3,4]. 为保证不同应用场景上的检测效果, 一种直接的解决方案是通过集成学习[5]组合多种异常检测算法. 然而, 一方面, 此类方法每次检测时都需要运行多个算法, 计算代价较高. 另一方面, 现有研究表明[4], 多种时间序列异常检测算法的集成模型所能达到的检测准确率, 通常低于单独运行各个算法所能达到的最佳水平. 因此, 实践中亟需针对不同场景的数据特点, 选择最佳时间序列异常检测算法, 从而实现最优检测准确率.

由于不同业务中时间序列数据特点复杂多样[6], 为每类数据人工选择最佳异常检测算法不仅耗时耗力, 且需要工作人员具备丰富的数据分析经验, 难以满足实际应用的需求. 为此, 研究人员提出了时间序列异常检测算法的自动选择方法[4,7,8]. 其主流思想是基于元学习[9], 利用候选的异常检测算法在历史时间序列数据上的检测准确率离线构建时间序列分类[10]模型. 模型将输入时间序列映射为对应的类别, 即候选异常检测算法中具有最高检测准确率的算法. 分类模型能够根据新生成时间序列的特征在线预测最佳异常检测算法[4], 从而最大程度减少人工干预, 提升效率并降低成本.

尽管当前的时间序列异常检测算法自动选择方法能有效弥补单一异常检测算法难以应对不同场景的不足, 但现有方法存在一个严重缺陷, 即仅利用标准的时间序列分类技术[11,12]来构建算法预测模型, 缺乏对于时间序列异常检测相关知识的充分利用. 主要体现在两方面. 第一, 现有方法在训练面向异常检测的时间序列分类模型时, 对历史时间序列采用硬标签标注, 即将每个时间序列的类别标记为具有最高准确率的异常检测算法[4]. 这种标记方式忽略了除标记类别而外的其他异常检测算法在历史时间序列上的检测准确率信息, 使得模型难以充分学习有关各异常检测算法间关联性的知识[13]. 第二, 标准的时间序列分类方法仅能对原始时间序列和标注 (即异常检测算法的准确率) 进行学习, 无法有效利用时间序列异常检测任务相关的外部知识, 如应用领域、数据特征和异常情况的描述等. 上述缺陷造成现有方法选择最佳异常检测算法的能力有限, 异常检测的准确率不高.

针对上述问题, 本文提出了一种知识增强的时间序列异常检测算法自动选择方法, 充分利用时间序列异常检测有关的知识, 提升异常检测算法的选择能力, 从而提高异常检测的准确率. 本文的主要贡献包括以下内容.

(1) 基于候选的时间序列异常检测算法在历史时间序列上的准确率, 估计时间序列的类别分布, 以此作为软标签, 设计软标签分类损失函数来指导时间序列分类模型的训练, 从而使模型学习更多有关异常检测算法间关联的知识.

(2) 提出一种能灵活融入各类外部知识的时间序列异常检测知识融合模块, 将知识映射为统一的表征, 并通过对比学习最大化知识表征与时间序列表征间的互信息, 从而将时间序列异常检测相关的外部知识融入时间序列分类模型中.

(3) 本文的方法被设计为插件的形式, 使其能够无缝集成到任意时间序列分类网络模型的训练中, 提高模型选择最优异常检测算法的能力, 且带来的额外计算时间可以忽略不计.

(4) 多组对比实验验证了本文所提方法在时间序列异常检测算法自动选择问题上的有效性.

本文第 1 节介绍时间序列异常检测算法自动选择的相关方法和研究现状. 第 2 节介绍本文所需的基础知识, 包括本文用到的主要符号、时间序列异常检测算法自动选择问题的形式化定义, 以及基于时间序列分类的时间序列异常检测算法自动选择求解流程. 第 3 节介绍本文构建的知识增强的时间序列异常检测算法自动选择方法. 第 4 节通过对比实验验证了所提方法的有效性. 最后, 第 5 节总结全文.

1 时间序列异常检测算法自动选择相关工作

时间序列异常检测技术在数据库监控运维[2,14]、故障实时预警[15]等众多领域具有广泛应用. 针对不同场景中时间序列数据的特点, 研究者提出了多种类型的时间序列异常检测算法[1,3]. 例如, 基于距离的算法[16–19]检索时间序列中与近邻子序列具有较低相似度的子序列. 这些子序列代表了时间序列中“不和谐”的异常片段. 基于邻近性 (proximity) 的方法[20–22]针对数据空间中正常模式分布紧凑而异常模式较为稀疏的场景, 通过密度估计或空间划分等策略区分正常模式与异常模式. 基于预测的方法通过循环神经网络[23]等时间序列预测模型, 根据过去一段时间的数据预测当前时间段内的数据, 并以预测值作为当前时间段的正常值, 通过真实观测值与预测值之间的误差大小来判断是否发生异常. 基于重建的方法通过自编码器[24]、生成对抗网络[25]等生成模型学习正常时间序列的重构方式, 并假定异常模式难以通过学习到的生成模型进行重构, 因而能通过重构误差的大小来判断输入序列是否为异常.

在真实应用中, 由于时间序列数据特征和异常模式复杂多样, 目前尚未存在一种时间序列异常检测算法能在所有数据上表现良好[1,3,4]. 尽管通过集成学习[5]组合多种异常检测算法可以弥补单一算法准确率不足的缺陷, 但该方案需要运行所有候选的算法, 计算代价会随着异常检测算法数量的增加而显著增大. 此外, 许多场景下, 集成多种异常检测算法所取得的准确率往往低于单独执行这些算法所能达到的最佳水平[4]. 因此, 如何根据时间序列数据的特点自动选择最优异常检测算法, 从而提高检测准确率, 是当前一项重要研究课题.

现有的时间序列异常检测算法自动选择方法主要包含两类: 基于代理指标的方法[26]和基于历史经验的方法[4,7,8].

基于代理指标的方法根据时间序列异常检测的特点, 设计无监督的代理指标来评估异常检测结果的准确性, 并以代理指标为依据选择最准确的算法. 与集成方法一样, 这类方法通常需要运行所有候选的异常检测算法, 并利用相应的检测结果来计算代理指标, 因而面临严重的效率问题. 不仅如此, 通过代理指标估计异常检测算法的真实准确率具有较大误差, 影响算法选择的效果.

基于历史经验的方法利用元学习的思想, 通过构建机器学习模型, 从已有的时间序列异常检测任务中学习有关知识, 指导新任务上的算法选择. 该过程被抽象为一类特殊的时间序列分类[10]问题, 即构建一个映射函数, 将输入时间序列映射到相应的最佳时间序列异常检测算法上. 由此, 任何时间序列分类技术, 如现有的基于时间序列相似度[12,27]、基于特征[12]和深度学习方法[11], 都能用来解决时间序列异常检测算法的自动选择问题. 相较于基于代理指标的方法, 利用历史经验构建的时间序列分类模型能直接为给定时间序列选择最佳异常检测算法, 无需通过运行候选算法来评估其检测准确率, 因而具有更高计算效率. 且以异常检测算法在历史任务上的真实准确率为依据, 有利于识别并选择更准确的异常检测算法, 从而提高目标任务上的异常检测准确率. 因此, 该类方法是当前及未来时间序列异常检测算法自动选择方法的主流[4].

2 基础知识

本文所提的时间序列异常检测算法自动选择方法在基于历史经验和时间序列分类技术的方法框架之上构建

而成. 下面就相关概念和基本知识予以介绍.

2.1 符号定义

记 $T \in R^n$ 为长度为 n 的时间序列, 即 n 个随时间有序排列的实数值 $T=(T_1,T_2,\ldots,T_n)$. 记 $T_{i,l} \in R^l$ 为 T 中起始点为 i 长度为 l 的子序列, 即 $T_{i,l}=(T_i,T_{i+1},\ldots,T_{i+l-1})$.

记 A 为一个时间序列异常检测算法. 对于输入 T, 算法 A 输出长度为 n 的异常分数序列, 表示时间序列 T 中每个数据点的异常分数, 记为 $S_T=A(T)=(S_{T,1},S_{T,2},\ldots,S_{T,n})$, 其中 $S_{T,i} \in [0,1]$. 记 $L \in \{0,1\}^n$ 为 T 的真实异常标记序列, 其中 0 和 1 分别表示对应的数据点为正常和异常. 记 $Acc(A(T),L)$ 为异常检测算法 A 在时间序列 T 上的异常检测准确率 (通过精度-召回率曲线下的面积 AUC-PR[28]、精度-召回率调和均值 $F1$-score[1]等指标度量), 用于度量算法检测的异常分数 S_T 与真实异常 L 的接近程度 (越接近则检测准确率越高).

表 1 对本文使用的主要符号进行了说明.

表 1 主要符号说明

符号	说明	符号	说明
T, L	时间序列及对应的真实异常标签	$\boldsymbol{x}_{T,i,l}$	$T_{i,l}$的时序表征
$T_{i,l}$	序列 T 中起始点为 i 长度为 l 的子序列	p_i	预测为 $T_{i,l}$ 选择各个算法的概率
S_T	序列 T 的异常分数	$s(\cdot)$	Softmax函数
A	时间序列异常检测算法	$p_{\text{hard},i}(p_{\text{soft},i})$	$T_{i,l}$的算法选择硬 (软) 标签
$\mathcal{B}$	时间序列异常检测算法候选集	t_{soft}	软标签温度系数
$Acc(A(T),L)$	算法 A 在序列 T 上的检测准确率	$\boldsymbol{x}_{K,i,l}$	$T_{i,l}$对应的知识表征
f	时间序列异常检测算法选择器	$g_T(g_K)$	时序 (知识) 表征的映射函数
$\mathcal{T}_l$	序列 T 的长度为 l 的不重叠子序列集合	$\boldsymbol{z}_{T,i,l}(\boldsymbol{z}_{K,i,l})$	共享空间上的时序 (知识) 表征
$O(T)$	序列 T 的真实最佳异常检测算法	τ	对比损失温度系数
E_T	时序特征提取器	$D_T/D_K/D$	时序表征/知识表征/共享空间维度
P	线性分类器	$\alpha(\lambda)$	软硬标签 (知识融入) 重要性

2.2 时间序列异常检测算法自动选择问题的定义

时间序列异常检测算法自动选择问题的定义如下.

定义 1. 时间序列异常检测算法自动选择. 给定时间序列异常检测算法的候选集合 $\mathcal{B}=\{A_1,A_2,\ldots,A_m\}$, 时间序列异常检测算法自动选择的目标是构建一个映射函数 f, 输入时间序列 T, 返回 $\mathcal{B}$ 中具有最高准确率的异常检测算法. 表示为:

$$f(T)=\arg\max_{A\in\mathcal{B}}\{Acc(A(T),L)\} \tag{1}$$

其中, L 为 T 的真实异常标签.

当前主流方法[4]基于时间序列分类技术求解定义 1 中的时间序列异常检测算法自动选择问题, 其方法流程将在第 2.3 节中详细介绍.

2.3 基于历史经验和时间序列分类技术的问题求解框架

根据定义 1, 算法选择函数 f 可以视作一类特殊的时间序列分类[11,12]模型, 能够将输入时间序列映射为具有最高准确率的异常检测算法所对应的类别. 因此, 基于元学习[9]思想, 可以将历史积累的时间序列及相应的异常检测算法的准确率作为训练数据, 通过时间序列分类技术来学习 f, 从而为未知时间序列预测最佳异常检测算法.

为了应对真实场景中时间序列长短不一且直接对长序列建模的计算复杂度过高的问题, Sylligardos 等人[4]提出一种通用的异常检测算法自动选择流水线, 将所有时间序列分割为固定长度的子序列. 具体来说, 对于任意时间序列 $T \in R^n$, 给定子序列长度 l ($l \leqslant n$), 将 T 分割为彼此相邻但不重叠的子序列集合 $\mathcal{T}_l$. 特别地, 若 T 无法被 l 等分, 则最前序的两个子序列间有一定重叠. $\mathcal{T}_l$ 的数学表示如下:

$$\mathcal{T}_l=\begin{cases}\left\{T_{1+i\times l,l}|i\in\left\{0,1,\ldots,\left\lceil\frac{n}{l}\right\rceil\right\}\right\}, & \left\lceil\frac{n}{l}\right\rceil=\frac{n}{l}\\ \{T_{1,l}\}\cup\left\{T_{n-\lceil\frac{n}{l}\rceil+i\times l,l}|i\in\left\{0,1,\ldots,\left\lceil\frac{n}{l}\right\rceil-1\right\}\right\}, & \left\lceil\frac{n}{l}\right\rceil>\frac{n}{l}\end{cases} \tag{2}$$

上述划分有 3 个优势. 第一, 当前的时间序列分类方法大多面向固定长度时间序列而设计. 将原始时间序列划分为固定长度子序列, 便于与现有时间序列分类方法兼容. 第二, 固定序列的长度有利于 GPU 等硬件分批并行处理, 从而提高计算效率. 第三, 将超长时间序列分割为较短的子序列能降低序列建模的计算复杂度.

在上述划分基础上, 利用标准的时间序列分类技术, 以历史积累的时间序列数据及对应的类别标签作为训练数据来学习函数 f. 具体地, 给定候选异常检测算法在历史时间序列 T 上的检测准确率 $Acc(A_1(T),L)$, $Acc(A_2(T),L),\ldots,Acc(A_m(T),L)$, 其类别标签对应具有最高准确率的异常检测算法, 表示为:

$$O(T)=\arg\max_{i=1,\ldots,m}\{Acc(A_i(T),L)\} \tag{3}$$

时间序列 T 分割出的所有固定长度子序列 $\mathcal{T}_l$ 与 T 具有相同类别标记. 即:

$$O(T_{i,l})=O(T),\ \forall T_{i,l}\in\mathcal{T}_l \tag{4}$$

经上述预处理的训练数据可以与任意时间序列分类方法相结合来构建分类函数 f, 即时间序列异常检测算法选择器. 在测试阶段, 对于未知时间序列 T_{test}, 首先根据公式 (2) 将其分割为同样长度的子序列; 再通过 f 分别预测每个子序列的类别; 最后通过多数表决的方式进行投票, 选择得票最多的类别对应的异常检测算法对 T_{test} 进行异常检测. 图 1 展示了基于时间序列分类的时间序列异常检测算法自动选择的流程.

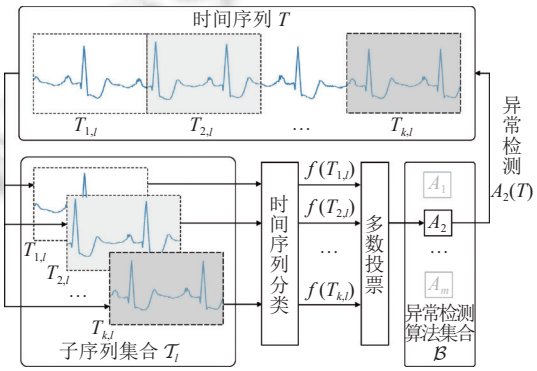


图 1 基于时间序列分类的时间序列异常检测算法自动选择流程

3 知识增强的时间序列异常检测算法自动选择方法

3.1 方法概述

本文所提的知识增强的时间序列异常检测算法自动选择方法总体框架如图 2 所示. 与现有方法[4]一样, 本文方法的核心目标是构建面向异常检测的时间序列分类函数 (即算法选择器) f. 算法选择器由两部分组成.

时序特征提取器 E_T 从时间序列 $T_{i,l}$ 中提取时序表征 (特征向量) $\boldsymbol{x}_{T,i,l}\in R^{D_T}$, 表示为:

$$\boldsymbol{x}_{T,i,l}=E_T(T_{i,l}) \tag{5}$$

其中, E_T 可以是任意面向时间序列分类的网络架构, 如被广泛验证具有优越性能的卷积神经网络[29]和 Transformer[30]等.

线性分类器 P 以时序表征 $\boldsymbol{x}_{T,i,l}$ 为输入, 预测 $T_{i,l}$ 属于每个类别 (即选择相应的异常检测算法) 的概率 $p_i\in R^m$, 表示为:

$$p_i=(\Pr(A_1),\Pr(A_2),\ldots,\Pr(A_m))=P(\boldsymbol{x}_{T,i,l})=s(\boldsymbol{x}_{T,i,l}\cdot W) \tag{6}$$

其中, $W\in R^{D_T\times m}$ 为权重系数. $\forall i, s(\boldsymbol{x})=\dfrac{e^{x_i}}{\sum_j \boldsymbol{x}_j}$ 为 Softmax 函数[31]. $\Pr(A_k)$ 表示选择异常检测算法 A_k 的概率.

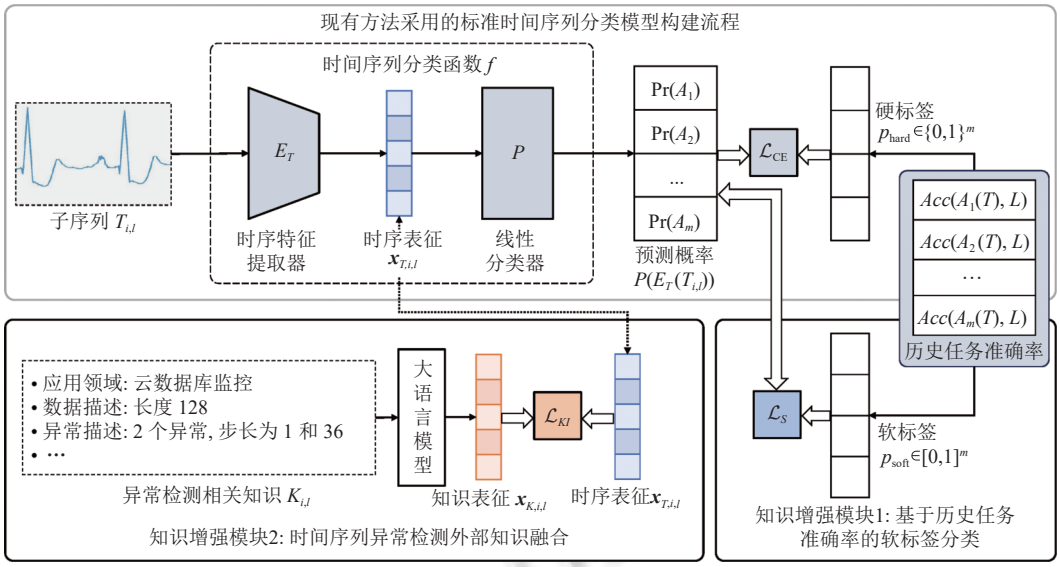


图 2 知识增强的时间序列异常检测算法自动选择方法总体框架图

f 的输出为具有最大预测概率的类别, 表示为:

$$f(T_{i,l}) = \arg\max_{k=1,\ldots,m} \{\Pr(A_k)\} \tag{7}$$

如图 2 灰色线框部分所示, 现有方法仅利用历史时间序列 $T_{i,l}$ 和第 2.3 节定义的标记 $O(T_{i,l})$ (等价于第 3.2 节所述的硬标签) 作为训练样本对, 使用标准时间序列分类模型训练技术来构建 f, 即以最小化预测概率 p_i 与标记 $O(T_{i,l})$ 间的交叉熵 (cross-entropy) 损失[31]为目标, 学习 f 的最优参数. 损失函数表示为:

$$\mathcal{L}_{CE} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_{CE,i} \tag{8}$$

其中, N 为训练样本的个数. $\mathcal{L}_{CE,i}$ 表示单个样本的损失, 定义为:

$$\mathcal{L}_{CE,i} = \log p_{i,j},\ j = O(T_{i,l}) \tag{9}$$

这种标准时间序列分类模型构建方式缺乏对于历史任务中时间序列异常检测有关知识的有效挖掘和利用. 针对这一问题, 本文提出两类知识增强模块: 基于历史任务的软标签分类模块利用每个历史时间序列 $T_{i,l}$ 上所有异常检测算法的准确率估计其真实类别分布, 以此作为软标签来为 f 的学习提供更多关于候选异常检测算法间关系的知识; 时间序列异常检测外部知识融合模块利用预训练的大语言模型[32]获取外部知识的通用表征, 并通过最大化时序表征与知识表征间的互信息, 将外部知识融入算法选择器 f 中. 两个模块分别在第 3.2 和 3.3 节中详细介绍.

3.2 基于历史任务准确率的软标签分类

现有方法仅利用历史任务上取得最高准确率的异常检测算法 $O(T_{i,l}) = O(T)$ 作为时间序列 $T_{i,l}$ 的标注. 这是一种硬标签标注方式, 即算法选择器 f 的预测目标为二元取值的概率向量, 记为 $p_{hard} \in \{0,1\}^m$, 其中类别 $O(T)$ 对应的概率为 1, 其余类别的概率值为 0, 表示为:

$$p_{hard,j} = \begin{cases} 1, & j = O(T) \\ 0, & j \neq O(T) \end{cases},\ j = 1,\ldots,m \tag{10}$$

这种硬标签标注方式将最佳异常检测算法外的所有候选算法不加区别地对待 (目标概率均置为 0), 忽略了异常检测算法间的准确率大小存在多样化差异这一重要知识. 为了弥补这一缺陷, 本文利用每个历史时间序列 T 上所有异常检测算法候选的准确率来估计真实目标概率向量, 以此作为软标签来指导 f 的训练.

记软标签为 p_{soft}, 它应满足两个条件:

(1) 概率化. p_{soft} 应为概率向量, 即满足 $p_{\text{soft}} \in [0,1]^m$ 且各维度的值之和为 1.

(2) 单调性. p_{soft} 各维度中的概率值应与其对应算法的准确率呈正相关, 即准确率越高的算法对应的输出概率越大.

为了满足上述条件, 本文将软标签 p_{soft} 定义为:

$$p_{\text{soft}} = s\left(\left(\frac{Acc(A_1(T),L)}{t_{\text{soft}}}, \frac{Acc(A_2(T),L)}{t_{\text{soft}}}, \ldots, \frac{Acc(A_m(T),L)}{t_{\text{soft}}}\right)\right) \tag{11}$$

其中, $t_{\text{soft}} > 0$ 为温度系数超参数, 用来控制软标签的分布平滑程度. t_{soft} 越小, p_{soft} 的分布形状越尖锐, 即不同类别间概率差异越大. 当 t_{soft} 趋于 0 时, p_{soft} 的极限值即为 p_{hard}.

图 3 通过简单的示例说明了现有方法采用的硬标签和本文提出的基于历史任务准确率的软标签的差异. 3 个候选的时间序列异常检测算法在该训练数据上的准确率分别为 0.2、0.5 和 0.3, 对应的硬标签 $O(T)$ 为 2, 等价于 $p_{\text{hard}} = (0,1,0)$, 表示算法 A_2 在时间序列 T 上的异常检测准确率最高. 与之相比, 由公式 (11) 计算出的软标签为 $p_{\text{soft}} = (0.042, 0.844, 0.114)$. p_{soft} 不仅反映了历史任务上最准确的异常检测算法, 同时也量化了所有候选算法间准确率的大小关系. 以 p_{soft} 作为输出的目标值, 可以为算法选择器 f 提供有关于各个时间序列异常检测算法之间关联关系的信息, 有助于 f 学习到更多算法选择的相关知识, 提高新任务上的检测准确率.

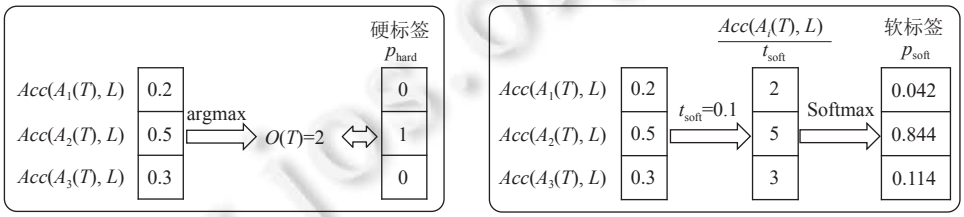


图 3 现有方法采用的硬标签和本文提出的基于历史任务准确率的软标签示例

与第 2.3 节介绍的流水线相同, 每个历史时间序列 T 的所有定长子序列 $T_{i,l}$ 与 T 共享相同的软标签, 表示为 $p_{\text{soft},i} = p_{\text{soft}}$. 基于软标签的算法选择器训练的优化目标是最小化当前模型输出 $p_i = P(E_T(T_{i,l}))$ 与软标签 $p_{\text{soft},i}$ 之间的交叉熵损失, 定义为:

$$\mathcal{L}_{S,i} = \sum_{j=1}^{m} p_{\text{soft},i,j} \log p_{i,j} \tag{12}$$

3.3 时间序列异常检测外部知识融合

除训练标准时间序列分类模型 (即算法选择器) 所需的时间序列数据和异常检测算法的准确率外, 时间序列异常检测的历史任务中亦包含许多额外信息[3], 如异常检测任务的应用场景、历史数据的异常类型和分布等. 本文将其统称为外部知识. 将这些外部知识融入算法选择器中, 有助于进一步提高选择器对异常检测算法的判别能力. 然而, 如何有效实现外部知识的融合颇具挑战. 主要原因有两个.

(1) 外部知识种类多样, 不同历史任务和时间序列数据对应的知识类型不一而足. 例如, 有些任务包含丰富的异常产生原因和机理知识, 如云原生数据库系统的内存溢出、磁盘损坏、网络阻塞导致的系统崩溃等性能异常. 而有些任务仅能获取异常的表层信息, 如异常模式的数量、持续的步长等. 针对每类知识的特点设计特定的知识融入方法显然是不现实的.

(2) 与历史任务不同, 在算法选择和异常检测的执行阶段, 在线新增的异常检测任务往往缺乏外部知识. 倘若直接将时间序列和外部知识进行融合建模, 会导致算法选择器在原有时间序列分类模型基础上增加额外参数, 以便学习外部知识与时间序列的联合特征. 这不仅会增大在线测试阶段的计算代价, 更会因外部知识缺失造成的数据分布差异, 导致算法选择器的性能下降甚至失效.

针对上述挑战, 本文构建了一个能灵活统一处理各类知识、深入理解知识间关联的外部知识融合模块, 并将

外部知识与时间序列的处理解耦合, 使得训练阶段可以利用外部知识增强算法选择器的学习. 而在线测试阶段与第 2.3 节介绍的流水线完全相同, 仅利用时间序列分类模型进行算法选择, 从而避免因测试阶段缺乏知识而影响算法选择的效果. 不仅如此, 将外部知识处理模块解耦合, 能使其与任意时间序列分类网络架构无缝集成, 提高普适性. 且在测试阶段, 相较于现有的无知识融入方法不会增加额外的计算.

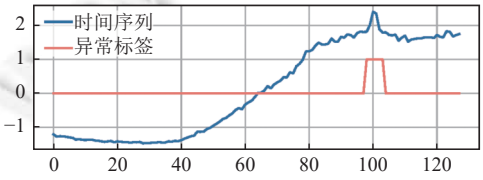
如图 2 所示, 外部知识融合模块主要包含两部分. 首先, 将各类时间序列异常检测知识统一表述为提示词(prompt)[33], 并通过预训练的大语言模型[32]对提示词进行编码, 利用大语言模型的通用理解能力, 将外部知识转换为向量化的知识表征. 接着, 基于时间序列与外部知识间的对比学习[34], 最大化时间序列表征与知识表征的互信息, 从而利用时间序列异常检测有关的外部知识指导时间序列特征的学习. 训练得到的时间序列分类模型 f 作为算法选择器, 实现异常检测算法的自动选择. 该模块的详细介绍如下.

为了实现各类时间序列异常检测知识的灵活运用, 利用提示词模板, 将外部知识统一表述为提示词. 提示词模板的示例见表 2.

表 2 表述外部知识的提示词模板示例

知识种类	模板
应用领域	这个时间序列来自{领域名称}. {应用场景描述}.
时间序列特征	该序列的长度为{长度}. 平均值为{均值}…
异常特征	该序列包含{数目}处异常. 持续长度为{长度1}, {长度2}, …
…	…

图 4 以一个云原生数据库存储设备监控应用为例说明了利用提示词模板描述的时间序列异常检测外部知识. 该应用中的时间序列记录了存储设备的每秒 I/O 操作. 图 4(a) 所展示的序列为原始时间序列分割而成的一个长为 128 的子序列, 其在第 99–104 个采样点上 I/O 操作过于频繁, 表示系统出现异常. 基于表 2 中的提示词模板, 该时间序列对应的异常检测有关外部知识表述如图 4(b) 所示.



(a) 长度为 128 的子序列 $T_{i,128}$ 及其异常标签

这个时间序列来自{云原生数据库存储设备监控}. {记录存储设备的每秒 I/O 操作. 反映了该云原生数据库上网络服务的规模、质量和机器的运行状况}. 该序列的长度为{128}. 平均值为{0.153 8}. 该序列包含{1}处异常. 持续长度为{6}.

(b) 外部知识

图 4 时间序列、异常标签及利用提示词模板表述的外部知识示例

采用上述的提示词方式, 仅需根据业务知识对提示词模板进行简单调整, 即可灵活描述各类时间序列异常检测有关的外部知识.

以提示词为基础, 考虑到外部知识的语义复杂性和多样性, 知识融合模块利用预训练的大语言模型的通用理解能力, 将知识映射为统一的表征, 以便于将其融入算法选择器 f.

由于外部知识是对相应时间序列数据中所蕴含的关键信息的凝练, 二者在决定最优时间序列异常检测算法这个维度应具备语义一致性. 从该视角出发, 本文通过时间序列与外部知识间的对比学习来最大化其互信息[34], 从而利用外部知识提高编码器 E_T 对时间序列特征的表达能力. 给定时间序列 $T_{i,l}$ 及其表征 $\boldsymbol{x}_{T,i,l} \in R^{D_T}$, 记 $T_{i,l}$ 对应的知识表征为 $\boldsymbol{x}_{K,i,l} \in R^{D_K}$, 首先通过映射函数 g_T 和 g_K, 将 $\boldsymbol{x}_{T,i,l}$ 和 $\boldsymbol{x}_{K,i,l}$ 分别映射到共享的空间. 表示为:

$$\boldsymbol{z}_{T,i,l} = g_T(\boldsymbol{x}_{T,i,l}) \in R^D \tag{13}$$

$$\boldsymbol{z}_{K,i,l} = g_K(\boldsymbol{x}_{K,i,l}) \in R^D \tag{14}$$

继而, 在共享空间 R^D 上, 最小化时间序列表征 $\boldsymbol{z}_{T,i,l}$ 和其对应的知识表征 $\boldsymbol{z}_{K,i,l}$ 之间的距离, 同时最大化 $\boldsymbol{z}_{T,i,l}$ 和

其他所有时间序列的表征 $\boldsymbol{z}_{T,j,l}$ $(j \neq i)$ 之间的距离, 以此实现时间序列和外部知识的对比学习. 该步骤通过最小化对比损失来完成. 由此, 知识融入模块优化的损失函数 $\mathcal{L}_{KI}$ 定义为:

$$\mathcal{L}_{KI} = \mathcal{L}_{CL} \tag{15}$$

其中, $\mathcal{L}_{CL}$ 可以由任意对比损失实现. 简单起见, 本文采用对比学习领域广泛使用的 InfoNCE 损失[34]来实现知识融入, 表示为:

$$\mathcal{L}_{KI,i} = -\log \frac{\exp\left(sim(\boldsymbol{z}_{T,i,l}, \boldsymbol{z}_{K,i,l})/\tau\right)}{\sum_{j \neq i} \exp\left(sim(\boldsymbol{z}_{T,i,l}, \boldsymbol{z}_{T,j,l})/\tau\right)} \tag{16}$$

其中, τ 为温度系数超参数, 用来控制对难学习样本的关注程度. $sim(\boldsymbol{u}, \boldsymbol{v})$ 表示向量 $\boldsymbol{u}$ 和 $\boldsymbol{v}$ 的余弦相似度, 即:

$$sim(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{|\boldsymbol{u}||\boldsymbol{v}|} \tag{17}$$

由于预训练的大语言模型具备通用的语言理解能力, 其参数在训练过程中保持冻结. 因此, 知识融入模块中仅对函数 g_K 进行优化更新.

3.4 方法总结与分析

综上所述, 本文所提方法的最终学习目标是最小化如下损失函数:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \left[(1-\alpha)\mathcal{L}_{\mathrm{CE},i} + \alpha \mathcal{L}_{S,i} + \lambda \cdot \mathcal{L}_{KI,i}\right] \tag{18}$$

其中, N 为训练样本总数. 超参数 λ 控制知识融入损失的重要性. $\alpha \in (0,1]$ 用于平衡原有的基于硬标签的分类损失和本文提出的软标签损失. 训练过程由经典的反向传播算法[35]实现: 给定输入数据, 根据公式 (18) 计算当前的损失值 $\mathcal{L}$, 继而通过链式求导法则得到 $\mathcal{L}$ 关于全部模型参数的梯度, 并通过梯度下降法对模型参数进行更新. 以上步骤迭代执行, 直到达到中止条件.

所提方法的计算复杂度由时间序列编码器和大语言模型的架构决定. 特别地, 当算法选择器 f 采用相同架构时, 本文所提方法相较于第 2.3 节介绍的现有方法, 训练阶段增加的主要计算在于利用大语言模型处理外部知识. 当前的大语言模型采用 Transformer 架构[30,32], 计算复杂度为 $\mathcal{O}\left(N_{\mathrm{layer}}(l_{\mathrm{seq}}^2 D_{\mathrm{repr}} + l_{\mathrm{seq}} D_{\mathrm{repr}}^2)\right)$, 其中 N_{layer} 为注意力层的数目, l_{seq} 为输入长度, D_{repr} 为嵌入的维度. 由于算法仅使用冻结参数的大语言模型, 且 Transformer 架构具有卓越的并行计算能力[30], 所提方法在实际环境下的训练时间与现有方法相差甚微 (详见第 4.5 节). 而在测试 (即模型推理或算法选择) 阶段, 本文方法与现有方法的计算逻辑和计算量完全相同.

4 实验分析

4.1 实验数据

使用最近发布的时间序列异常检测评测基准 TSB-UAD[3]中的数据集进行实验验证. 包含 16 个来自不同应用场景、具有不同数据特征和异常情况的公开数据集. 数据集信息如后文表 3 所示.

为与现有方法公平比较, 采用 Sylligardos 等人[4]提出的方法划分训练和测试数据, 其中训练数据包含上述所有数据集中的时间序列, 测试数据来自其中的 14 个子集. 使用表 2 中的模板 (无序列均值) 提取外部知识.

4.2 基准方法与评测指标

采用多种在时间序列异常检测算法自动选择任务上表现出色的方法作为基准. 这些方法通过标准时间序列分类技术构建异常检测算法选择器. 大致分为 4 类.

(1) 基于特征的方法. 使用时间序列特征提取工具 TSFresh[36]从分割后的时间序列中提取特征, 并在特征之上构建传统机器学习模型作为异常检测算法选择器[4]. 机器学习方法包括 K 最近邻 (KNN) [37]、支持向量分类器 (SVC)[38]、梯度提升决策树 (AdaBoost)[39]和随机森林 (RF)[40].

(2) 基于卷积的方法. 基于卷积模块[31]构建的时间序列分类模型, 包含一个由 3 层卷积层、最大池化层和分类

层堆叠而成的卷积神经网络 ConvNet (记作 CN)[41]; 带有残差连接的残差网络 ResNet[4]; 以及使用不同尺寸卷积核的残差网络作为基模块的集成网络 InceptionTime (记作 IT)[29].

(3) 基于 Transformer 的方法. 使用 Sylligardos 等人[4]构建的基于 SiT-stem[42]的 Transformer 模型 (记作 Trans) 作为异常检测算法选择器. 这是一种基于卷积模块和多头自注意力机制的时间序列分类模型.

(4) 基于 Rocket 的方法. MiniRocket[43]是一种特殊的时间序列分类算法. 该方法利用大量具有随机参数的卷积核和预先设计的池化操作, 将时间序列转换成特征向量. 进而在特征向量上构建线性分类器. 为了与现有研究[4]一致, 本文将 MiniRocket 方法简称为 Rocket.

表 3 时间序列异常检测数据集信息

数据集	序列数量	平均序列长度	平均异常数量	平均异常长度	应用场景描述
Daphnet	45	21 760	7.6	2 841	帕金森病患者臀部和腿部3个加速度传感器的带注释读数. 这些患者在步行任务期间会出现步态冻结 (FoG)
Dodgers	1	50 400	133	5 612	洛杉矶101号北高速公路格伦代尔入口匝道的环形传感器数据. 异常现象代表道奇队比赛后的异常交通
ECG	52	230 351.9	195.6	15 634	标准心电图数据. 异常代表室性早搏. 通过识别信号的周期性, 将一个长度约为1E7的长序列拆分为47个序列
Genesis	6	16 220	3	50	来自一款便携式拾放演示器的数据. 该设备使用储气罐为所有抓取和存储单元供电
GHL	126	200 001	1.2	388.8	柴油加热循环数据, 包含3个储层的状态, 例如温度和液位. 异常表明最高温度或泵频率发生变化
IOPS	58	102 119.2	46.5	2 312.3	机器性能指标监控数据集, 反映了Web服务的规模、质量和机器的健康状态
KDD21	250	77 415.06	1	196.5	在最近的SIGKDD 2021中发布的复合数据
MGAB	10	100 000	10	200	由具有非平凡异常的Mackey-Glass时间序列组成, 表现出人眼难以区分的混乱行为
MITDB	32	650 000	210.1	72 334.3	摘录了48个时长为30 min的双通道动态心电图记录, 这些记录来自BIH心律失常实验室在1975–1979年间研究的47名受试者
NAB	58	6 301.7	2	575.5	由真实世界和人工时间序列组成, 包括AWS服务器指标、在线广告点击率、实时流量数据以及Twitter提及的大型上市公司的集合
Occu.	10	5 725.8	18.3	1 414.5	根据温度、湿度、光和二氧化碳进行二元分类 (房间占用) 的实验数据. 地面实况占用率通过每分钟拍摄的带时间戳的照片获得
OPP	465	31 616.9	2	1 267.3	包括用户执行典型日常活动时记录的运动传感器读数. 旨在对人类活动识别算法 (例如分类、自动数据分割、传感器融合和特征提取) 进行基准测试
Sen.Sco.	23	27 038.4	11.2	6 110.4	从典型的分层传感器测量系统收集的环境数据的集合, 例如温度、湿度和太阳辐射
SMD	281	25 562.3	10.4	900.2	从一家大型互联网公司收集的为期5周的服务器性能检测数据, 包含来自28台不同机器的3组实体
SVDB	115	230 400	208	27 144.5	包括78个时长为30 min的心电图记录, 用于补充MIT-BIH心律失常数据库中的室上性心律失常示例
YAHOO	367	1 561.2	5.9	10.7	雅虎公司某生产系统生产流量的真实和合成时间序列

时间序列异常检测算法自动选择的目的是提高未知数据上的异常检测准确率. 因此, 本文以算法选择器在测试数据上所选算法的异常检测准确率为标准来比较所提方法与基准方法的性能优劣. 异常检测准确率的评测指标使用 AUC-PR[28]. 该指标不受类别不平衡的影响, 并且独立于预测阈值, 因此被广泛用于异常检测任务中. AUC-PR 衡量的是精度-召回率 (Precision-Recall) 曲线下方的面积. 该曲线以精度作为横坐标、召回率作为纵坐标在二维空间中绘制. 面积值越大, 所评测算法的准确率越高.

4.3 实验方法

本文所提方法基于 Python 3.8 和 PyTorch 库实现. 实验在一台拥有 Platinum 8260 CPUs 和 Ubuntu 20.04 LTS

操作系统的服务器上运行, 并使用单个英伟达 GTX 3090 GPU. 与 Sylligardos 等人[4]的设置一致, 实验采用 12 个不同类型的时间序列异常检测算法作为候选, 包括 8 个完全无监督方法: IForest、IForest1、LOF、MP、NormA、PCA、HBOS、POLY 和 4 个半监督方法: OC-SVM、AE、LSTM-AD、CNN. 算法简介见表 4.

表 4 候选的时间序列异常检测算法

异常检测算法	描述
IForest	基于随机空间分裂构造二叉树, 到根的路径较短的节点 (即输入的子序列) 有更大概率为异常
IForest1	与IForest使用相同算法, 但输入子序列的长度为1, 即以单个时间步的值为输入进行检测
LOF	通过相邻密度与局部密度的比例判断异常
MP	检测具有最大近邻距离的子序列作为异常
NormA	利用聚类来识别正常模式
PCA	利用主成分分析将数据投影到低维超平面, 并将距离超平面较远的数据点判断为异常
HBOS	构建数据的直方图, 并使用组距 (Bin) 高度的倒数作为数据点的异常分数
POLY	用历史数据拟合多项式模型来预测当前子序列, 基于预测误差判断异常值
OC-SVM	通过One-Class SVM拟合正常数据找到正常数据的边界, 将边界之外的数据判断为异常
AE	利用自编码器学习正常数据的低维特征表示和重构方法, 并假设重建误差大的序列为异常
LSTM-AD	在历史数据训练长短时记忆网络 (LSTM) 来预测当前子序列, 基于预测误差判断异常值
CNN	在历史数据训练卷积神经网络 (CNN) 来预测当前子序列, 基于预测误差判断异常值

依第 2.3 节所述流程, 计算各个异常检测算法在所有时间序列上的检测准确率, 其中训练数据的准确率用来生成算法选择器训练所需的标签. 测试数据上的检测准确率用来评估算法选择器的最优算法选择能力.

为了公平比较, 时间序列异常检测算法及算法自动选择基准方法的超参数设置与 Sylligardos 等人[4]的研究一致. 每个基准方法均使用子序列长度 $l \in \{16, 32, 64, 128, 256, 512, 768, 1024\}$ 分别测试准确率并报告最佳结果.

对于本文所提方法, 使用基准方法中的残差网络 ResNet 作为算法选择器. 子序列长度 l 固定为 128. 大语言模型使用预训练的 BERT[32], 模型通过开源工具 Hugging Face (https://huggingface.co/) 实现, 版本为基础版 (BERT-Base). g_T 和 g_K 分别由两个单隐层的多层感知机 (MLP)[31]实现, 隐藏层维度为 256, 激活函数为 ReLU, 输出层维度从{64, 256}中选择. τ 的值设置为 0.1. α 的值从{0.2, 0.4, 1.0}中选择, 分别表示软标签相较硬标签具有较低、适中和较高重要性. 对本文方法在 t_{soft} 和 λ 不同取值下的验证准确率进行实验探究. 结果显示, t_{soft} 取值区间为 [0.2, 0.25] 时, 本文方法整体表现较好, 且准确率相对 t_{soft} 波动较明显 (见后文图 5(a)). 因此, 后续实验中 t_{soft} 从{0.2, 0.22, 0.25}中选择. 类似地, 如后文图 5(b) 所示, λ 取值在 [0.78, 1] 时, 本文方法整体具有较高准确率, 且其相较 λ 波动平缓, 故 λ 取值仅从区间端点{0.78, 1.0}中选择, 从而实现超参数寻优空间规模与算法选择效果的平衡. 为提高效率, 通过随机搜索[44]策略优化超参数. 其他设置与基准方法一致.

4.4 实验结果分析

本文方法与基准方法所选定的时间序列异常检测算法在各数据集上的准确率 (AUC-PR) 结果如表 5 所示. 每个数据集上的最高准确率用粗体展示. 为了进行全面比较, 在准确率结果基础上统计了几项关键指标. 其中, “最佳次数”表示每个方法在所有数据集上取得最高准确率的次数. “平均排名” 计算每个方法在各个数据集上准确率排名的平均值. “Wilcoxon p 值” 表示用 Wilcoxon 符号秩检验的 p 值 (p-value) 度量的本文方法与每个基准方法在各个数据集上准确率排名的差异, 其中下划线标示了小于 0.05 的 p 值, 表示在 95% 的置信区间内, 两种方法的准确率具有统计显著性差异. “一对一比较” 将本文方法与各个基线方法单独比较, 并统计本文方法的准确率胜/平/负于被比较对象的数据集个数.

实验结果显示, 本文方法 (Ours) 在 14 个数据集的 8 个子集中达到当前最高的检测准确率. 在余下 6 个数据集上, 本文方法的准确率相较于 9 个基准方法仍具有较高的准确率排名: Daphnet 排名第 4、GHL 排名第 2、NAB 排名第 2、OPP 排名第 4、SMD 排名第 5、YAHOO 排名第 2. 所有数据集上准确率排名的平均值为 2.43, 超过了全部基准方法. 在此基础之上, 结合本文方法与基准算法在各个数据集上平均排名的 Wilcoxon 符号秩检验结果,

可以看出本文方法在 0.05 的统计学水平上显著优于 KNN、SVC、AdaBoost、RF、CN、IT、Trans 和 Rocket 方法. 与每个基准方法的一对一准确率对比结果也显示, 本文方法在大多数据集 (至少 10 个) 上异常检测准确率高于或持平对比方法. 以上实验结果表明, 相较于现有的时间序列异常检测算法自动选择方法, 本文所提方法能在多种不同领域和应用中选择具有更高检测准确率的算法, 体现了其优越性.

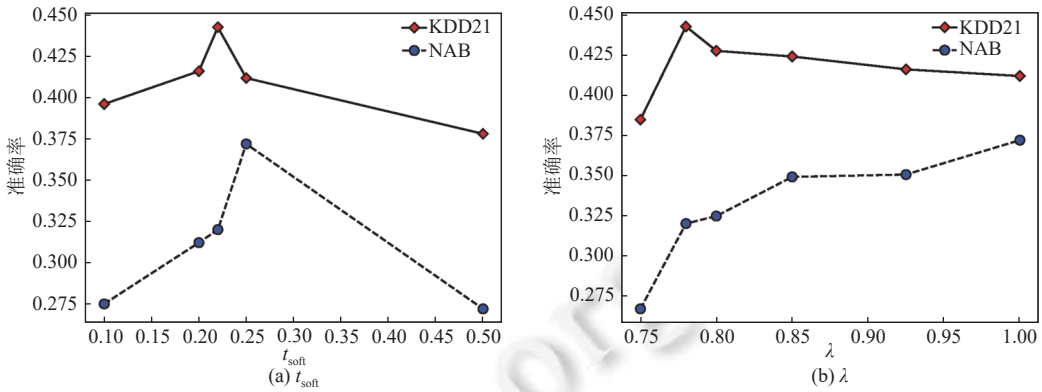


图 5 本文方法在超参数 t_{soft} 和 λ 的不同取值下的准确率

表 5 本文方法与基准方法的异常检测准确率对比

数据集	KNN	SVC	AdaBoost	RF	CN	ResNet	IT	Trans	Rocket	Ours
Daphnet	0.1197	0.1888	0.1197	0.1854	0.2445	0.2994	**0.3129**	0.3070	0.1397	0.2873
ECG	0.6842	0.6842	0.6842	0.6842	0.6154	0.6717	0.6187	0.6176	0.6842	**0.6897**
Genesis	0.0017	0.0017	0.0017	0.0017	**0.3617**	**0.3617**	0.1796	0.2957	0.1796	**0.3617**
GHL	0.3071	0.3042	0.3071	**0.3335**	0.2571	0.2573	0.2637	0.2851	0.3068	0.3071
IOPS	0.0734	0.1807	0.0734	0.0686	0.2732	0.2765	0.2235	0.2714	0.1430	**0.3090**
KDD21	0.4132	0.3154	0.3876	0.3808	0.3874	0.3595	0.2987	0.3616	0.3372	**0.4426**
MGAB	**0.6140**	0.0122	0.0122	0.0122	**0.6140**	**0.6140**	**0.6140**	**0.6140**	**0.6140**	**0.6140**
MITDB	**0.4856**	**0.4856**	**0.4856**	**0.4856**	0.4132	0.3123	0.4298	0.3739	0.4562	**0.4856**
NAB	0.1867	0.2783	0.2127	0.1739	0.3263	0.3507	0.3019	**0.4195**	0.3558	0.3730
OPP	0.3238	0.3116	0.3157	0.3092	**0.4080**	0.4075	0.3849	0.4013	0.3472	0.3995
Sen.Sco.	0.2857	0.2941	0.2755	0.2941	0.335	0.3365	0.3350	0.3191	0.2521	**0.3844**
SMD	0.2479	0.3497	0.2278	0.3358	**0.5004**	0.4857	0.4832	0.4819	0.3175	0.4576
SVDB	0.6317	0.6317	0.6317	0.6317	0.6043	0.6339	0.5940	0.6369	0.6391	**0.6408**
YAHOO	0.3144	0.1934	0.2718	0.2681	0.7385	0.7447	**0.7616**	0.7091	0.3474	0.7558
最佳次数	2	1	1	2	4	2	3	2	1	**8**
平均排名	6.36	6.89	7.00	6.89	4.96	4.36	5.5	4.71	5.89	**2.43**
Wilcoxon p值	0.0033	0.0014	0.0022	0.0021	0.0494	0.0636	0.0117	0.0156	0.0014	—
一对一比较 (胜/平/负)	11/3/0	13/1/0	12/2/0	12/1/1	10/2/2	9/2/3	10/1/3	9/1/4	13/1/0	—

为了进一步验证本文所研究的算法自动选择方法的优越性和必要性, 将本文所提方法与表 4 中的异常检测算法及其集成方法进行一对一准确率比较, 其中集成方法 AvgEnsemble 运行所有异常检测算法并以输出异常分数的平均值作为最终检测结果. 从图 6(a)–(l) 中可以看出, 本文提出的时间序列异常检测算法自动选择方法在绝大多数情况下准确率高于或持平单独使用一种异常检测算法, 且在准确率低于单个异常检测算法的情形中, 本文方法的准确率也与被比较对象十分接近. 特别地, 图 6(m) 中的结果显示, 本文所提的算法自动选择方法在各个数据集中一致地优于所有异常检测算法的集成方法. 这更加说明时间序列异常检测算法自动选择的必要性和本文方法的有效性.

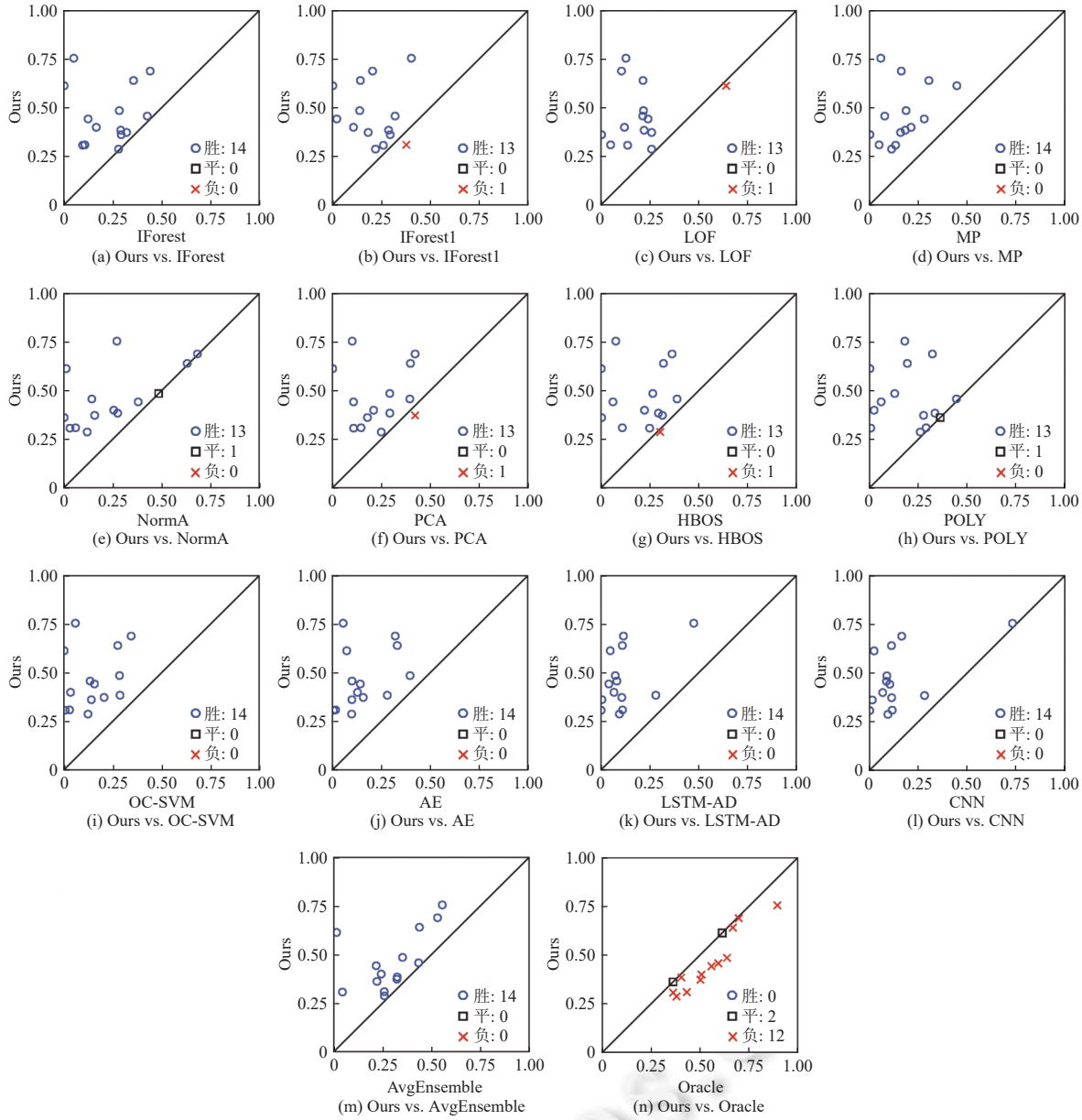


图 6　本文方法与单个异常检测算法、所有算法的集成以及神谕 (Oracle) 算法选择方法的准确率对比

最后, 将本文方法与算法自动选择的神谕 (Oracle) 方法进行对比. 后者根据所有候选异常检测算法在测试集上的检测准确率真值, 为每个时间序列选择具有最高准确率的算法, 代表的是异常检测算法自动选择方法所能达到的理论上限. 图 6(n) 中的结果显示, 本文方法在 2 个数据集上与神谕方法准确率持平, 另有 3 个数据集上准确率接近神谕方法, 而在其余 9 个数据集上仍存在一定差距. 结果表明, 在本文方法的基础之上, 针对时间序列异常检测算法的自动选择问题, 仍存在进一步优化空间.

4.5　消融实验

本节通过消融实验进一步探究所提的两个知识增强模块的有效性. 为此, 将本文方法与其 3 个变体进行比较. 所有方法采用相同的超参数配置. 为便于比较, 在第 4.3 节设置的基础上将 g_T 和 g_K 的输出维度固定为 64. t_{soft} 固

定为 0.22. 参与比较的方法包括以下几种.

(1) 无模块 1. 算法选择器训练中不使用知识增强模块 1, 即基于历史任务准确率的软标签分类模块.

(2) 无模块 2. 算法选择器训练中不使用知识增强模块 2, 即时间序列异常检测外部知识融合模块.

(3) 无模块 1 和 2. 不使用本文提出的两个知识增强模块, 对应于现有的标准时间序列分类方法.

(4) 两模块皆有. 同时使用本文提出的两个知识增强模块, 是本文所提方法的默认配置.

表 6 显示的是本文方法 (两模块皆有) 与各变体的异常检测准确率. 可以看出, 同时使用本文提出的两个知识增强模块相较于仅使用其一或不使用这些模块, 在大部分数据集上能取得更高检测准确率. 其准确率的平均排名为 1.75, 高于其他 3 个变体, 且排名的优势在 0.05 水平上具备统计显著性 (Wilcoxon p 值小于 0.05). 模块 1 和 2 均不使用的变体 (仅基于第 3.1 节所述的标准时间序列分类模型构建方法) 在各项统计指标上的表现最差. 与之相比, 单独使用本文提出的任意一种知识增强模块, 都能有效提高所选择算法的异常检测准确率. 注意到无模块 2 的变体整体表现略优于无模块 1 的变体, 这表明相较于软标签分类模块, 外部知识融合模块对于算法选择准确率的提升更小. 造成这一结果的原因可能是实验使用的外部知识相对浅显. 在实践中, 如果能利用更多真实应用场景中积累的业务知识作为外部知识, 有希望更好地发挥本文提出的外部知识融合模块的效果, 带来进一步的准确率提升.

表 6 本文方法与其变体的检测准确率

数据集	无模块1	无模块2	无模块1和2	两模块皆有 (默认配置)
Daphnet	**0.3014**	0.2873	0.2873	0.2873
ECG	0.6259	**0.6897**	0.6624	**0.6897**
Genesis	**0.3617**	**0.3617**	**0.3617**	**0.3617**
GHL	0.2303	**0.3071**	0.1932	0.3035
IOPS	**0.3090**	0.2843	0.2843	**0.3090**
KDD21	0.3125	0.3875	0.2902	**0.4426**
MGAB	**0.6140**	**0.6140**	**0.6140**	**0.6140**
MITDB	0.3123	**0.4856**	0.3355	**0.4856**
NAB	0.3137	**0.3319**	**0.3319**	0.3279
OPP.	**0.4031**	0.3886	0.3886	0.3995
Sen.Sco.	0.3350	0.3350	0.3350	**0.3844**
SMD	0.4501	0.4561	0.4561	**0.4576**
SVDB	0.6215	0.6212	0.6212	**0.6337**
YAHOO	0.7535	0.7370	0.7370	**0.7558**
最佳次数	5	6	3	**10**
平均排名	2.68	2.50	3.07	**1.75**
Wilcoxon p值	0.0384	0.0427	0.0069	—
一对一比较 (胜/平/负)	9/3/2	7/5/2	10/3/1	—

最后, 对本文方法及其 3 个变体在所有时间序列数据上的整体运行时间进行测试. 结果如表 7 所示. 可以看出, 4 类方法的训练时间和测试 (即模型推理或算法选择) 时间均非常接近. 表明本文提出的两个知识融合模块的额外计算代价很小, 几乎可以忽略不计.

表 7 本文方法及其变体的运行时间 (s)

阶段	无模块1	无模块2	无模块1和2	两模块皆有 (默认配置)
训练	16923.22	16825.20	16913.44	16921.70
测试 (算法选择)	34.27	35.34	35.46	35.07

4.6 案例研究

本节以一个云原生数据库监控运维场景中的典型应用为例, 深入探究本文方法取得更高检测准确率的原因.

实验采用 IOPS 数据集, 其时间序列记录的是一台机器存储设备的每秒 I/O 操作 (input/output operations per second, IOPS). 测试数据集包含 17 条记录, 其中 9 条记录中存在异常. 单条记录上异常数目最大为 63, 最小为 5. 其他 8 条记录无异常.

选择在 IOPS 上取得最高准确率的基准方法 CN 作为对照. 为便于比较, 对异常检测算法在测试数据集中每条时间序列记录上的真实检测准确率及本文方法与对照方法的算法选择结果进行了可视化, 如图 7 所示.

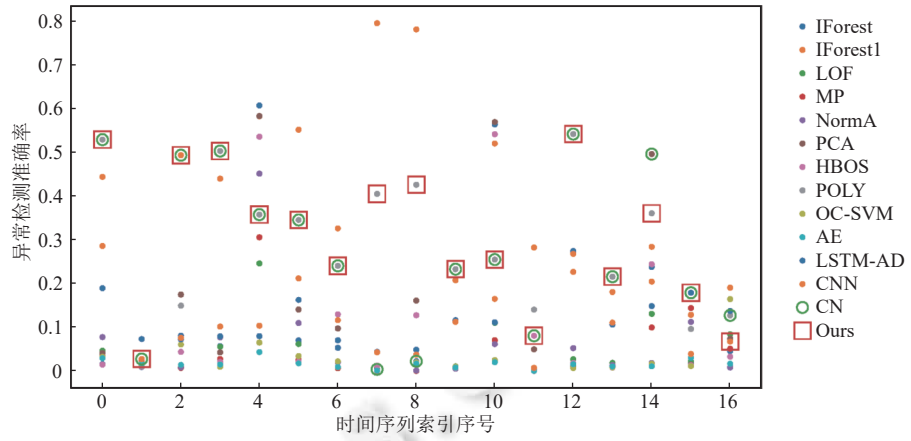


图 7　本文方法与基准方法为 IOPS 数据集中的时间序列选择的异常检测算法及其相应的检测准确率

结果显示, 12 个候选的异常检测算法在各个时间序列上的准确率表现参差不齐. 本文方法与对照方法分别在 7 和 8 条记录上精准选择了具有最高准确率的异常检测算法, 并且二者在 13 条记录上选择了相同的算法. 然而, 对比两种方法做出不同选择的 4 条记录 (对应序号 7、8、14 和 16) 可以发现, 现有的 CN 方法可能选择准确率非常低的算法, 如序号 7 和 8. 造成这一现象的主要原因是现有方法仅使用对应最佳算法的硬标签, 而将所有非最佳算法不加区别地对待, 导致算法选择器难以从非最佳算法中辨别出较优的候选. 相比之下, 本文方法通过历史任务准确率估算的软标签来强化不同异常检测算法的关系, 且异常检测外部知识的融入也有助于算法选择器捕获更多算法选择相关的特征. 这使得本文方法更倾向于选择所有候选中准确率排名较高的算法, 因而能取得更高的检测准确率.

5　总结与展望

时间序列异常检测算法的自动选择问题具有重要研究意义. 针对当前主流的基于标准时间序列分类的算法选择器构建方法难以有效利用历史知识的问题, 本文提出知识增强的时间序列异常检测算法自动选择方法. 方法包含两个模块. 基于历史任务的软标签分类模块利用历史时间序列对应的所有异常检测算法的准确率估计序列的真实类别分布, 以此作为软标签, 为算法选择器的学习提供更多有关异常检测算法间关系的知识. 时间序列异常检测外部知识融合模块利用预训练的大语言模型获取外部知识的通用表征, 并通过对比学习最大化时序表征与知识表征间的互信息, 以此将外部知识融入算法选择器中. 本文在来自不同领域的多个时间序列数据集上进行了广泛实验. 结果表明, 本文所提方法相较于现有的时间序列异常检测算法自动选择方法具有更高的检测准确率, 且带来的额外计算时间可以忽略不计.

在本文研究的基础上, 未来研究可以从以下 3 个方向展开. 首先, 针对时间序列异常检测算法选择问题的特点, 研究泛化能力更强的算法选择模型架构和训练方法, 进一步提升异常检测算法选择的精确性; 其次, 面向时间序列数据的动态演化特性, 研究增量式元学习框架, 实现算法选择方法的在线动态演化; 最后, 面向海量时间序列异常检测场景的需要, 研究适用于端侧的轻量化异常检测算法选择架构, 提升工业互联网等重要应用场景下的实时检测效率.

References

[1] Schmidl S, Wenig P, Papenbrock T. Anomaly detection in time series: A comprehensive evaluation. Proc. of the VLDB Endowment, 2022, 15(9): 1779–1797. [doi: 10.14778/3538598.3538602]

[2] Dong HW, Zhang C, Li GL, Feng JH. Survey on cloud-native databases. Ruan Jian Xue Bao/Journal of Software, 2024, 35(2): 899–926 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6952.htm [doi: 10.13328/j.cnki.jos.006952]

[3] Paparrizos J, Kang YH, Boniol P, Tsay RS, Palpanas T, Franklin MJ. TSB-UAD: An end-to-end benchmark suite for univariate time-series anomaly detection. Proc. of the VLDB Endowment, 2022, 15(8): 1697–1711. [doi: 10.14778/3529337.3529354]

[4] Sylligardos E, Boniol P, Paparrizos J, Trahanias P, Palpanas T. Choose wisely: An extensive evaluation of model selection for anomaly detection in time series. Proc. of the VLDB Endowment, 2023, 16(11): 3418–3432. [doi: 10.14778/3611479.3611536]

[5] Zhou ZH. Ensemble learning. In: Machine Learning. Singapore: Springer, 2021. 181–210. [doi: 10.1007/978-981-15-1967-3_8]

[6] Liang ZY, Zhang JF, Liang C, Wang HZ, Liang Z, Pan LJ. A shapelet-based framework for unsupervised multivariate time series representation learning. Proc. of the VLDB Endowment, 2023, 17(3): 386–399. [doi: 10.14778/3632093.3632103]

[7] Zhao Y, Rossi RA, Akoglu L. Automatic unsupervised outlier model selection. In: Proc. of the 35th Int'l Conf. on Neural Information Processing Systems. Curran Associates Inc., 2021. 4489–4502.

[8] Ying YX, Duan JY, Wang CL, Wang YJ, Huang CR, Xu BX. Automated model selection for time-series anomaly detection. arXiv:2009.04395, 2020.

[9] Li FC, Liu Y, Wu PX, Dong F, Cai Q, Wang Z. A survey on recent advances in meta-learning. Chinese Journal of Computers, 2021, 44(2): 422–446 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2021.00422]

[10] Yang YM, Pan R, Pan JL, Yang Q, Li L. A comparative study on time series classification. Chinese Journal of Computers, 2007, 30(8): 1259–1266 (in Chinese with English abstract). [doi: 10.3321/j.issn:0254-4164.2007.08.007]

[11] Ismail Fawaz H, Forestier G, Weber J, Idoumghar L, Muller PA. Deep learning for time series classification: A review. Data Mining and Knowledge Discovery, 2019, 33(4): 917–963. [doi: 10.1007/s10618-019-00619-1]

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

[13] Gou JP, Yu BS, Maybank SJ, Tao DC. Knowledge distillation: A survey. Int'l Journal of Computer Vision, 2021, 129(6): 1789–1819. [doi: 10.1007/s11263-021-01453-z]

[14] Jin LY, Li GL. AI-based database performance diagnosis. Ruan Jian Xue Bao/Journal of Software, 2021, 32(3): 845–858 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6177.htm [doi: 10.13328/j.cnki.jos.006177]

[15] Ellefsen AL, Han PH, Cheng X, Holmeset FT, Æsøy V, Zhang HX. Online fault detection in autonomous ferries: Using fault-type independent spectral anomaly detection. IEEE Trans. on Instrumentation and Measurement, 2020, 69(10): 8216–8225. [doi: 10.1109/TIM.2020.2994012]

[16] Fu AWC, Leung OTW, Keogh E, Lin J. Finding time series discords based on Haar transform. In: Proc. of the 2nd Int'l Conf. on Advanced Data Mining and Applications. Xi'an: Springer, 2006. 31–41. [doi: 10.1007/11811305_3]

[17] Keogh E, Lonardi S, Ratanamahatana CA, Wei L, Lee SH, Handley J. Compression-based data mining of sequential data. Data Mining & Knowledge Discovery, 2007, 14(1): 99–129. [doi: 10.1007/s10618-006-0049-3]

[18] Linardi M, Zhu Y, Palpanas T, Keogh E. Matrix profile goes MAD: Variable-length motif and discord discovery in data series. Data Mining and Knowledge Discovery, 2020, 34(4): 1022–1071. [doi: 10.1007/s10618-020-00685-w]

[19] Yeh CCM, Zhu Y, Ulanova L, Begum N, Ding YF, Dau HA, Silva DF, Mueen A, Keogh E. Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In: Proc. of the 16th IEEE Int'l Conf. on Data Mining. Barcelona: IEEE, 2016. 1317–1322. [doi: 10.1109/ICDM.2016.0179]

[20] Breunig MM, Kriegel HP, Ng RT, Sander J. LOF: Identifying density-based local outliers. In: Proc. of the 2000 ACM SIGMOD Int'l Conf. on Management of Data. Dallas: ACM, 2000. 93–104. [doi: 10.1145/342009.335388]

[21] Liu FT, Ting KM, Zhou ZH. Isolation forest. In: Proc. of the 8th IEEE Int'l Conf. on Data Mining. Pisa: IEEE, 2008. 413–422. [doi: 10.1109/ICDM.2008.17]

[22] Ma HR, Ghojogh B, Samad MN, Zheng DY, Crowley M. Isolation Mondrian forest for batch and online anomaly detection. In: Proc. of the 2020 IEEE Int'l Conf. on Systems, Man, and Cybernetics (SMC). Toronto: IEEE, 2020. 3051–3058. [doi: 10.1109/SMC42975.2020.9283073]

[23] Malhotra P, Vig L, Shroff G, Agarwal P. Long short term memory networks for anomaly detection in time series. In: Proc. of the 23rd European Symp. on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges, 2015. 89–94.

[24] Sakurada M, Yairi T. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In: Proc. of the 2nd Workshop on

Machine Learning for Sensory Data Analysis. Gold Coast: ACM, 2014. 4–11. [doi: 10.1145/2689746.2689747]

[25] Li YF, Peng XY, Zhang J, Li ZY, Wen M. DCT-GAN: Dilated convolutional Transformer-based GAN for time series anomaly detection. IEEE Trans. on Knowledge and Data Engineering, 2023, 35(4): 3632–3644. [doi: 10.1109/TKDE.2021.3130234]

[26] Goswami M, Challu C, Callot L, Minorics L, Kan A. Unsupervised model selection for time-series anomaly detection. arXiv:2210.01078, 2023.

[27] Abanda A, Mori U, Lozano JA. A review on distance based time series classification. Data Mining and Knowledge Discovery, 2019, 33(2): 378–412. [doi: 10.1007/s10618-018-0596-4]

[28] Davis J, Goadrich M. The relationship between precision-recall and ROC curves. In: Proc. of the 23rd Int'l Conf. on Machine Learning. Pittsburgh: ACM, 2006. 233–240. [doi: 10.1145/1143844.1143874]

[29] Ismail Fawaz H, Lucas B, Forestier G, Pelletier C, Schmidt DF, Weber J, Webb GI, Idoumghar L, Muller PA, Petitjean F. InceptionTime: Finding AlexNet for time series classification. Data Mining and Knowledge Discovery, 2020, 34(6): 1936–1962. [doi: 10.1007/s10618-020-00710-y]

[30] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A, Kaiser L. Attention is all you need. In: Proc. of the 31st Int'l Conf. on Neural Information Processing Systems. Long Beach: Curran Associates Inc., 2017. 6000–6010.

[31] Goodfellow I, Bengio Y, Courville A. Deep Learning. Cambridge: MIT Press, 2016.

[32] Devlin J, Chang MW, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional Transformers for language understanding. In: Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1 (Long and Short Papers). Minneapolis: Association for Computational Linguistics, 2019. 4171–4186. [doi: 10.18653/v1/N19-1423]

[33] Liu PF, Yuan WZ, Fu JL, Jiang ZB, Hayashi H, Neubig G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM Computing Surveys, 2023, 55(9): 195. [doi: 10.1145/3560815]

[34] van den Oord A, Li YZ, Vinyals O. Representation learning with contrastive predictive coding. arXiv:1807.03748, 2019.

[35] Chauvin Y, Rumelhart DE. Backpropagation: Theory, Architectures, and Applications. New York: Psychology Press, 1995. [doi: 10.4324/9780203763247]

[36] Christ M, Braun N, Neuffer J, Kempa-Liehr AW. Time series feature extraction on basis of scalable hypothesis tests (tsfresh—A Python package). Neurocomputing, 2018, 307: 72–77. [doi: 10.1016/j.neucom.2018.03.067]

[37] Ramaswamy S, Rastogi R, Shim K. Efficient algorithms for mining outliers from large data sets. In: Proc. of the 2000 ACM SIGMOD Int'l Conf. on Management of Data. Dallas: ACM, 2000. 427–438. [doi: 10.1145/342009.335437]

[38] Zhang R, Zhang SY, Muthuraman S, Jiang JM. One class support vector machine for anomaly detection in the communication network performance data. In: Proc. of the 5th Conf. on Applied Electromagnetics, Wireless and Optical Communications. Tenerife: World Scientific and Engineering Academy and Society (WSEAS), 2007. 31–37.

[39] Tripathi AM, Baruah RD. Anomaly detection in multivariate time series using fuzzy AdaBoost and dynamic naive Bayesian classifier. In: Proc. of the 2019 IEEE Int'l Conf. on Systems, Man and Cybernetics (SMC). Bari: IEEE, 2019. 1938–1944. [doi: 10.1109/SMC.2019.8914477]

[40] Breiman L. Random forests. Machine Learning, 2001, 45(1): 5–32. [doi: 10.1023/A:1010933404324]

[41] Wang ZG, Yan WZ, Oates T. Time series classification from scratch with deep neural networks: A strong baseline. In: Proc. of the 2017 Int'l Joint Conf. on Neural Networks (IJCNN). Anchorage: IEEE, 2017. 1578–1585. [doi: 10.1109/IJCNN.2017.7966039]

[42] Ma NY, Goldstein M, Albergo MS, Boffi NM, Vanden-Eijnden E, Xie SN. SiT: Exploring flow and diffusion-based generative models with scalable interpolant Transformers. In: Proc. of the 18th European Conf. on Computer Vision. Milan: Springer, 2024. 23–40. [doi: 10.1007/978-3-031-72980-5_2]

[43] Dempster A, Schmidt DF, Webb GI. MiniRocket: A very fast (almost) deterministic transform for time series classification. In: Proc. of the 27th ACM SIGKDD Conf. on Knowledge Discovery & Data Mining. ACM, 2021. 248–257. [doi: 10.1145/3447548.3467231]

[44] Bergstra J, Bardenet R, Bengio Y, Kégl B. Algorithms for hyper-parameter optimization. In: Proc. of the 25th Int'l Conf. on Neural Information Processing Systems. Granada: Curran Associates Inc., 2011. 2546–2554.

附中文参考文献

[2] 董昊文, 张超, 李国良, 冯建华. 云原生数据库综述. 软件学报, 2024, 35(2): 899–926. http://www.jos.org.cn/1000-9825/6952.htm [doi: 10.13328/j.cnki.jos.006952]

[9] 李凡长, 刘洋, 吴鹏翔, 董方, 蔡奇, 王哲. 元学习研究综述. 计算机学报, 2021, 44(2): 422–446. [doi: 10.11897/SP.J.1016.2021.00422]

[10] 杨一鸣, 潘嵘, 潘嘉林, 杨强, 李磊. 时间序列分类问题的算法比较. 计算机学报, 2007, 30(8): 1259–1266. [doi: 10.3321/j.issn:0254-

4164.2007.08.007]
[14] 金连源, 李国良. 基于人工智能方法的数据库智能诊断. 软件学报, 2021, 32(3): 845–858. http://www.jos.org.cn/1000-9825/6177.htm [doi: 10.13328/j.cnki.jos.006177]

作者简介

梁志宇, 博士, 副研究员, CCF 专业会员, 主要研究领域为时序数据分析, 时序数据库.
蔡东芮, 硕士生, CCF 学生会员, 主要研究领域为时间序列异常检测.
梁晨, 博士生, 主要研究领域为数据挖掘, 自动机器学习.
梁峥, 博士生, 主要研究领域为数据集成, 实体识别, 异常检测.
王宏志, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为数据库管理系统, 大数据分析与治理.
郑博, 硕士, CCF 专业会员, 主要研究领域为时序数据库管理系统.