硕士学位论文

可解释的序列分类与聚类算法

Interpretable Sequence Classification and Clustering

作 者 姓 名: 董俊杰

学　　号: 22217058

指 导 教 师:

学科、 专业: 软件工程

答 辩 日 期: 2025 年 5 月

大连理工大学

Dalian University of Technology

摘要

序列数据广泛应用于生物信息学、文本分析和金融等领域，其中分类与聚类任务在模式识别和数据挖掘中具有重要作用。然而，现有的序列分类方法主要依赖复杂的深度学习模型来提升准确率，但往往缺乏可解释性，导致模型决策过程难以理解。同样，现有的序列聚类方法侧重于优化聚类效果，而忽视了对聚类形成依据的直观解释。这种可解释性的缺失降低了分析结果的可信度，不利于领域专家深入理解模型决策机制。因此，提高序列分类与聚类方法的可解释性，同时保证其性能，是当前序列分析领域亟待解决的问题。

针对上述问题，本文主要完成了以下工作：(1) 提出 Hamming 编码器神经网络，用于可解释序列分类。该模型在一维卷积神经网络（1DCNN）框架下，引入卷积核权重与 k-mer 子串的对应二值化策略，并采用基于 Hamming 距离的相似度度量，使模型能够自动学习并提取关键判别模式。在提高分类准确率的同时，Hamming 编码器神经网络能够提供透明的判别依据，增强分类结果的可解释性。(2) 设计可解释序列聚类树（ISCT）算法。该算法利用序列模式和树状结构对数据进行层次化划分，通过构建树形结构将序列数据划分为 k 个簇（叶节点），并使用 $k-1$ 个判别模式作为分裂准则，从而清晰揭示聚类形成的依据。此外，在聚类过程中引入提升策略，对序列进行重新投影和优化，提高聚类的稳定性和可理解性。

本文在多个基准序列数据集上对上述方法进行了实验评估。结果表明，Hamming 编码器神经网络在保证高分类准确率的同时，能够清晰解释分类决策；ISCT 在多个数据集上的聚类性能优于传统方法，并通过树状结构直观展示聚类过程。综上所述，本文提出的序列分类与聚类方法在性能和可解释性上均取得了显著进展，为序列数据分析提供了一种透明、高效的解决方案。

关键词：可解释性；序列分类；序列聚类；模式挖掘；卷积神经网络

ABSTRACT

Sequence data is widely used in bioinformatics, text analysis, and finance, where classification and clustering play a crucial role. However, existing classification methods often rely on complex deep learning models, making decision processes difficult to interpret. Similarly, clustering methods focus on optimizing results but lack clear explanations for cluster formation. This lack of interpretability reduces credibility and hinders domain experts from understanding model decisions.

To address this, we propose: (1) Hamming Encoder Neural Network for interpretable sequence classification. This model integrates pattern mining into a one-dimensional convolutional neural network (1DCNN) by binarizing convolutional kernel weights and introducing a Hamming distance-based similarity metric. It enhances accuracy while making decision-making transparent. (2) Interpretable Sequence Clustering Tree (ISCT), which uses sequence patterns and a tree structure for hierarchical clustering. The algorithm employs k-1 discriminative patterns as splitting criteria, providing clear explanations of cluster formation while improving stability.

Experiments on multiple benchmark datasets show that our methods achieve high classification accuracy and superior clustering performance while enhancing interpretability. In summary, these methods offer a transparent and effective solution for sequential data analysis.

Key Words: Interpretability; Sequence Classification; Sequence Clustering; Pattern Mining; Convolutional Neural Network

目　　录

TABLE OF CONTENTS

图 目 录

表 目 录

1 绪论

1.1 研究背景与意义

离散序列是按特定顺序排列的符号列表，其中的符号代表具体对象的抽象表达。在多个领域，如生物信息学[1]、文本分析[2] 和金融数据分析[3]，序列数据分析正发挥着越来越重要的作用。例如，在生物信息学中，基因序列、蛋白质序列的分析对疾病研究、药物开发以及生物进化研究至关重要。在文本分析领域，情感分析、机器翻译以及语音识别等任务都依赖对文本序列的理解和建模。在金融领域，股票市场趋势预测、信用风险评估以及用户交易行为分析往往涉及时间序列数据，其有效的分类与聚类对金融决策具有重要影响。

在序列分析任务中，分类是极具挑战性的任务。这主要是因为序列数据通常缺乏显式特征，同时其潜在特征空间维度极高。现有的序列分类方法大多致力于提升模型性能和分类准确率，但这些方法通常依赖于复杂的深度学习模型或其他高维特征提取手段，涉及大量参数，导致模型的可解释性较低。然而，在许多实际应用场景中，模型的可解释性至关重要。例如，在医疗诊断领域，医生不仅需要模型给出疾病预测结果，还需要理解影响诊断的关键基因序列模式；在金融科技领域，银行需要了解信用评分模型的决策依据，以确保其符合监管要求。因此，提高序列分类模型的可解释性，同时保持高分类性能，是当前研究的重要方向。

除了分类，聚类分析也是序列分析中的关键任务之一，其目标是将相似的序列归为一类。然而，当前主流的序列聚类算法多侧重于优化聚类结果的准确性，而较少关注解释序列为何被分入同一簇，这在实际应用中可能影响结果的可理解性和可信度。例如，在基因功能研究中，研究人员希望知道为什么某些基因序列被聚为一类，而不仅仅是得到分类结果；在市场分析中，商家希望理解用户行为数据的聚类逻辑，以便进行精准营销。现有聚类方法通常采用基于距离度量的计算方式，计算复杂度较高，并且难以提供直观的聚类解释。因此，开发可解释的序列聚类方法，使其能够揭示聚类形成的依据，提高模型的可理解性，是当前序列数据挖掘的重要研究方向。

从科学研究的角度来看，理解模型的决策依据以及聚类结果的形成机制至关重要。例如，在生物信息学领域，研究人员不仅希望知道基因序列如何被分类，还需要理解分类的依据，以揭示基因功能和进化关系，从而指导后续实验设计。在金融科技领域，银行和投资机构希望通过可解释的模型来分析市场趋势，提高信用评估的透明度，从而优化决策过程。在医疗健康领域，医生需要理解患者病症数据是如何被分类的，以确保模型的预测结果符合医学知识。然而，现有的序列分类和聚类方法通常难以提供清晰的解释，使得结果的可信度受到限制。因此，本研究的目标是探索能够兼

顾高性能和可解释性的序列分类与聚类方法，并为实际应用提供一种透明、可靠的数据分析工具。

具体而言，本文提出了 Hamming 编码器神经网络用于可解释的序列分类。该方法结合深度学习与模式挖掘技术，在提升分类准确率的同时，通过卷积核权重的二值化和基于 Hamming 距离的相似性度量，实现分类决策的透明化。此外，本文设计了一种可解释序列聚类树（ISCT），该方法利用树状结构和判别模式优化策略，使得序列聚类结果更加直观，并提供了清晰的聚类依据。这些方法不仅能够提升分类与聚类任务的性能，同时也能增强模型的透明性，为科学研究和实际应用提供有力支持。

综上所述，本研究的必要性体现在：一是解决现有序列分类方法可解释性不足的问题，使得模型决策过程更加透明；二是给出可解释的序列聚类方法，使得用户能够直观理解不同簇的形成机制。

1.2 研究现状

在过去的十几年中，研究人员提出了多种序列分类和聚类算法，通常可分为三大类：基于特征的方法、基于距离的方法和基于模型的方法 [4]。这些方法在各自的应用领域取得了显著成果，但在可解释性方面仍然存在不少局限性。

在序列分类方面，基于特征的方法通过统计等方法从序列中提取序列特征（如 k-mer、序列模式等），然后使用传统的机器学习算法进行分类（支持向量机、决策树等）或聚类（K-Means 等）。随着深度学习的兴起，研究者开始尝试直接使用神经网络[5]（如 1DCNN、RNN、LSTM 等）自动学习序列的隐式特征，并且构建端到端的模型，这些方法取得了较好的分类性能。然而，随着特征提取复杂度的提升，这些方法往往面临着可解释性不足的问题。尽管这些深度学习模型能够有效提高分类准确率，但由于其黑箱特性，研究人员很难理解模型是如何从序列中提取有意义的特征，也无法清晰地解释模型的决策过程。因此，开发具有可解释性的序列分类神经网络成为当前研究的一个重要方向，以便更好地理解模型的决策过程，并提高科学研究中的数据利用度。

在序列聚类方面，经典的 K-Means 等方法可以看作是具有一定可解释性的聚类算法，因为它们通过计算每个序列与聚类中心的距离来决定分类。然而，由于每个特征在计算距离时的影响很难被明确量化，尽管聚类结果可以通过质心和可视化来直观呈现，但这些方法的可解释性依然远低于基于规则或基于树的方法。因此，如何提升序列聚类方法的可解释性，也是当前研究的重要课题。基于距离的方法则通过衡量序列之间的相似度，利用距离度量进行分类或聚类。尽管这些方法提供了相对简单的思路来度量序列之间的关系，但它们往往不直接揭示决策过程，且对相似性度量的依赖使得方法的可解释性较低，尤其是在使用复杂的距离函数时，决策过程更加晦涩难

懂[6]。此外，基于模型的方法，如隐马尔可夫模型（HMM）等，假设不同类别的序列服从不同的概率分布，依此进行分类或聚类。尽管这种方法在统计学上有其优势，但对于非专业人员来说，理解模型的工作原理并非易事，尤其是在数据的分布较为复杂情况下。

近年来，针对非序列数据的可解释性研究逐渐增多，例如基于规则的方法和基于树的方法取得了一定进展。但这些方法并不能直接应用于序列数据，这主要是由于序列数据通常缺乏显式特征，且其高维度特征空间使得传统的可解释性方法难以直接适用。因此，有必要开发专门针对序列数据的可解释性方法，使得能够提供透明、直观的决策过程和聚类结果。

1.3 本文的主要工作

针对上述问题，本文提出了两种创新的解决方案，旨在提升序列分类与聚类方法的可解释性和性能。

首先，针对序列分类，本文提出了一种新的判别模式挖掘方法——Hamming 编码器神经网络。与传统的先挖掘模式后构建特征的两步走方法不同，该方法借助神经网络的自动特征提取能力，在一维卷积神经网络（1DCNN）框架下，开发了一种将卷积核权重与 k-mers 子串一一对应的二值化函数。此外，本文还提出了一种基于 Hamming 距离的相似度函数，该函数与全局最大池化操作兼容，从而有效识别并挖掘序列中的判别模式组合。通过这种方法，Hamming 编码器神经网络不仅能够提高分类性能，还能保持较强的可解释性，揭示模型的判别依据。具体而言，该方法通过梯度优化技术，根据相似度度量变换后的序列特征来优化模式识别过程，从而显著提升分类准确性，并提供透明的决策过程。因此，Hamming 编码器神经网络在提高分类准确率的同时，能够有效解释其决策依据。

其次，针对序列聚类，本文提出了一种可解释序列聚类方法——可解释序列聚类树（ISCT）。该方法结合了序列模式和简洁可解释的树状结构，为每个聚类提供了清晰的直观解释。ISCT 通过生成 k 个叶节点，对应 k 个聚类，并利用 $k-1$ 个模式来指导聚类的形成。在每个节点，方法采用提升策略对序列进行重新投影和重新聚类，从而进一步优化聚类结果，并增强模型的可解释性。与传统的聚类方法相比，ISCT 不仅能够提供更精确的聚类结果，还能帮助用户理解为何某些序列会被划分到特定的聚类中。

1.4 本文组织结构

本文组织结构如下：

第一章绪论，介绍了可解释的序列分类和聚类的在科学研究中的背景和意义，并

且对本文提出的基于权重可解释的序列分类神经网络和可解释序列聚类树进行了梳理和介绍。

第二章相关工作，介绍了序列分类和聚类的相关工作，包括基于特征的方法、基于距离的方法和基于模型的方法，并指出在可解释分类和聚类方面的研究现状。

第三章基于权重可解释的序列分类神经网络，详细介绍了本文提出的基于权重可解释的序列分类神经网络的设计和实现，包括网络结构、训练方法和实验结果。

第四章可解释序列聚类树，详细介绍了本文提出的可解释序列聚类树的设计和实现，包括树结构、构建方法和实验结果。

第五章中总结了本文的工作，并且讨论了未来的研究方向。

2 相关工作

本章从序列学习、可解释学习以及基于神经网络的模式挖掘方法三个方面介绍了国内外相关工作。

2.1 序列学习

离散序列是按照特定顺序排列的符号列表 $s = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$，其中 $\sigma_i \in I$，$I = \{e_1, e_2, \ldots, e_m\}$ 表示一个有限的符号集合。序列学习侧重于从这些序列数据中提取并利用有效信息，包括序列分类、序列聚类等任务。接下来将介绍序列学习的相关工作。

2.1.1 序列分类

（1）基于特征/模式的方法

基于特征的方法通常首先将序列表示为 $k$mers（或 n-grams）的存在与否的向量，其中 k-mer 是长度为 k 连续片段。然而，随着 k 值的增加，k-mer 的数量呈指数级增长，因此需要使用特征选择方法来筛选出一些无关的 k-mer。

后续随着子序列模式的概念被提出，一些方法将子序列视为模式表示，并提供了几种用于过滤模式的标准度量[7] 。随后的研究旨在通过对挖掘模式的集合应用约束，减少冗余并提高分类器的准确性 [8–10]。这些方法的主要区别在于特征/模式选择的标准。Gokrimp [11] 基于最小描述长度原理，识别出一组非冗余的压缩序列模式。MiSeRe [12] 引入了规则模式模型空间，并在该模型空间内定义了先验分布。SCIP [9] 利用凝聚力和支持度来定义有趣的模式。iBCM [13] 使用行为约束模板，涵盖了多种约束，能够表达简单的出现模式、循环模式和基于位置的行为模式。Seq2Pat [14] 采用二分模式挖掘来寻找频繁出现并具有期望特征的序列模式。此外，还提出了多种相似度函数，用于计算模式与序列之间的距离，从而提高模式的利用率 [9, 15]。与上述方法不同，Sqn2Vec [16] 将序列视为句子，句中每个序列模式被视为一个单词，从而使得序列能够像 Word2Vec 一样进行特征表示。然而，这些方法对每个模式进行单独评估，可能会错过一些判别性模式的组合。因此，这些方法无法保证将序列转换为特征向量后的模式集合具有足够的判别能力。

为了解决上述问题，一些研究将模式挖掘与分类器构建相结合 [17, 18]。这些方法能够识别出在特定分类器下最优的模式，但是也可能遗漏一些重要的判别模式组合。

（2）基于距离的方法

基于距离的序列分类方法通过定义一个距离度量函数来衡量两个序列之间的相似性。一旦获得了这种距离度量函数，就可以使用现有的分类方法（如 K 近邻分类器

（KNN）和支持向量机（SVM）与局部对齐核）进行序列分类。一些经典的基于距离的序列分类方法包括基于动态时间规整（DTW）的算法和基于最近邻（k-NN）的算法。这些方法通过比较序列间的距离或相似度来判断样本所属类别，具有较好的适应性和较高的准确度。后续的工作为了提升计算效率，基于距离的方法被拓展为基于参考点的方法 [19]，通过统计方法筛选一些特定的序列，然后利用这些序列作为参考点，计算其他序列与参考点之间的距离，从而减少计算量。

（3）基于模型的方法

基于模型的方法通常假设序列来自不同的潜在分布，并通过构建统计模型来捕捉序列的内在结构和规律。这些方法包括隐马尔可夫模型（HMM）[20]、马尔可夫链模型以及贝叶斯网络等。隐马尔可夫模型（HMM）通过引入隐状态来建模序列数据中的时间依赖性，已被广泛应用于语音识别、自然语言处理和生物信息学等领域 [21]。

除了传统的 HMM，近年来基于深度学习的模型也被广泛应用于序列分类任务。例如，基于循环神经网络（RNN）和长短时记忆网络（LSTM）的方法 [22] 已经取得了显著的成功，能够很好地捕捉长时间依赖关系。此外，基于自注意力机制的 Transformer 模型也被广泛应用于处理序列数据 [23]。

与基于距离的方法相比，基于模型的方法往往可以更好地捕捉序列的全局特征，但在模型复杂度和计算资源上往往要求较高。此外，这些方法通常需要大量的标注数据进行训练，因此在数据稀缺的情况下可能面临一定的挑战。

2.1.2 序列聚类

序列聚类旨在将一组序列划分为不同的簇，以便在同一簇内的序列具有相似性，而不同簇之间的序列具有显著的差异性。根据不同的聚类方法，序列聚类可以分为几类：基于特征/模式的方法、划分方法、基于模型的方法和层次方法。下面将详细介绍每种方法的特点和应用。

（1）基于特征/模式的方法

基于特征或模式的序列聚类方法主要有两种策略。第一种方法是将序列转化为特征向量，然后应用现有的聚类方法（如 K-Means [24] 或 DBSCAN [25]）来获得最终的划分结果。这些方法之间的主要差异体现在如何将序列转化为特征向量。常见的转化方法包括使用序列模式 [26]、k-mers [27] 或嵌入技术 [25, 28]。

另一种策略是先基于发现的序列模式构建初步的聚类结果，然后扩展并优化这些聚类，以实现最终的精细聚类结果 [27]。一些聚类方法 [29] 还尝试通过规则提取来解释所得的聚类，这有助于提高本文对聚类结果的理解。然而，这些方法通常没有提供清晰直观的聚类过程。

（2）划分方法

划分方法 [30] 直接将序列集合分成多个簇。这些方法并不依赖于将序列向量化再

应用传统的基于向量数据的聚类算法，而是直接对序列进行划分以获得聚类结果。

值得注意的是，某些方法既可以被认定为划分方法，也可以理解成基于特征的方法。因为基于特征的方法在特征提取后可能会使用各种聚类技术，比如 K-Means 聚类算法。按照本文的分类标准，只有那些直接将序列划分为簇而不显式进行特征提取的方法，才被视为划分方法。根据这个标准，基于特征和划分方法之间不会有交集。

（3）基于模型的方法

基于模型的聚类方法假设来自不同簇的序列遵循不同的概率分布。现有的基于模型的方法通常采用马尔可夫模型，通过构建考虑前序子序列的马尔可夫链来描述序列类别的发生，或使用隐马尔可夫模型（HMM），引入隐藏状态来表示序列的变化 [20]。

（4）层次方法

层次方法主要依赖于所有序列之间的距离。通常，首先需要计算序列之间的成对相似度矩阵。然后，采用聚合法（Agglomerative）或分裂法（Divisive）逐步构建层次结构。这些方法的主要区别在于它们使用不同的距离函数来计算相似度矩阵。常见的距离函数包括 Kullback-Leibler 距离 [31]、编辑距离（edit distance）[32] 和基于子序列的 Jaccard 相似度 [33]。

值得一提的是，层次方法已经具备了一定的可解释性，因为聚类结果可以基于样本间的距离进行解释。然而，这种解释可能不够直观，因为计算距离涉及复杂的序列元素组合，非专业研究者往往难以直接理解。

2.2 可解释学习

可解释学习关注于构建具有透明决策过程的机器学习模型，使人类能够理解模型的推理机制和决策依据。随着人工智能系统在医疗诊断、金融风控等关键领域的广泛应用，模型的可解释性变得日益重要。与仅注重预测准确性的传统黑盒模型不同，可解释学习方法旨在在保持较高性能的同时，提供清晰、直观且人类可理解的决策过程。本节将介绍可解释分类和可解释聚类两个方面的相关工作。

2.2.1 可解释分类

可解释分类方法近年来在机器学习领域中备受关注，特别是在需要高透明度和可理解性的应用场景中。当前主流的可解释分类方法包括基于原型的方法、基于树方法和基于规则的方法。

（1）基于原型的方法

基于原型的方法[34] 的目标是通过选取一组最小的样本子集，提供数据集的精简或凝练视图。与后处理分析方法不同，原型基分类器并不依赖于解释性分析，而是基于给定样本与选定原型之间的相似度做出决策。这种方法在解释性上具有优势，因为它符合认知科学中的直观方法[35–38]，使得每个样本的分类决策都可以通过与原型的

相似性来理解和解释。

（4）基于树的方法

树基分类方法因其结构直观、逻辑清晰而广泛应用于机器学习中，且具有很高的可解释性。决策树[39, 40] 提供了具体且透明的解释，清晰地展示了决策路径，帮助用户识别影响分类结果的关键因素。决策树的每个分支节点和叶节点都对应着明确的决策标准，因此分类结果的推理过程对于用户来说非常直观且易于理解。树基方法在许多需要可解释性的实际应用中，如医疗诊断、金融风控等领域，取得了显著成效。

（4）基于规则的方法

基于规则的方法因其直观、透明的推理过程在可解释学习领域占据重要地位，具有极高的可解释性。决策集和决策列表[41] 是此类方法中的典型代表。由于决策集中的规则可以独立应用而无需考虑顺序，相较于具有顺序依赖性的决策列表，决策集通常提供了更直接的可解释性[42]。这些方法通过明确的"IF-THEN"（如果-那么）规则结构呈现决策逻辑，使得即使是非专业用户也能轻松理解每个分类决策背后的推理过程，从而增强了模型的可信度和实用价值。

可解释决策集的发现通常被表述为一个二阶段过程：规则生成和规则选择。在这一过程中，首先通过挖掘数据模式来生成候选规则，接着通过优化一个非单调的子模函数来选择最终的规则集[42]。此外，一些研究还将规则生成与选择整合到一个统一的优化过程中[43]，进一步提升了方法的有效性和可扩展性。

2.2.2 可解释聚类

（1）基于模糊规则的方法

基于模糊规则的方法旨在从数据中抽取模糊 IF-THEN 规则，形式如下：

$$\text{规则 } R_i\text{: IF } x_1 \text{ is } \phi_{11} \text{ and } \ldots \text{ and } x_n \text{ is } \phi_{jj} \text{ THEN 簇 } c_i.$$

其中，x_n 表示样本 x 的第 n 个特征，ϕ_{jj} 表示该特征所需满足的模糊集合。通过发现满足同一规则的一组样本，聚类算法的决策过程能够以高可解释性呈现。根据隶属函数是否与数据相关，模糊规则方法可大致分为自适应划分和固定网格划分两种[44]。

在自适应划分方法中，数据集的信息会贯穿于整个划分过程中。如 FRCGC [45] 是一种基于粒计算的快速模糊规则聚类方法。它从样本描述中选取示例描述来指导数据粒化，为每个簇提供一个模糊规则。固定网格划分方法在模糊划分过程中并不考虑数据集信息，常用的隶属函数包括三角和梯形函数 [46]。

（2）基于多面体的方法

基于多面体的方法通过超平面构造聚类边界，这些超平面的组合形成多面体，并通过对超平面的描述来构建可解释的规则。这种方法与支持向量机（SVM）类似，二者都使用超平面作为决策边界。但与 SVM 不同的是，基于多面体的聚类方法不依赖

标签数据，因此寻找聚类边界的超平面更具挑战性。

从这一角度出发，研究者提出了多种优化算法。例如，有研究[47] 提出了以超矩形作为聚类边界的方法。由于初始生成的超矩形可能存在重叠区域，该方法引入了“软尾”机制来计算样本对不同矩形的隶属度，并通过交替最小化技术确定最终的聚类边界。随后，判别性矩形混合模型（DReaM）[48] 被提出，具有更高的灵活性，它允许用户明确区分两类特征：规则生成特征和聚类维持特征。DReaM 仅利用规则生成特征构建可解释的聚类规则，而聚类的内部结构则由聚类维持特征来决定。与前述基于矩形边界的方法不同，文献[49] 提出了一种新颖的方法，通过在每个簇周围构建多面体并将决策边界转换为可解释规则，从而实现了更为直观的聚类结果表示。

（3）基于阈值树的方法

阈值树 [50] 是一种可解释的聚类方法，它通过二叉树结构将数据集划分为若干个子集。树中的每个节点会根据特定特征的阈值对数据进行迭代划分。这样，数据集能够通过 $k-1$ 个节点被划分成 k 个部分，这种方法具有高度直观和可解释的特点。其清晰性的一大原因在于，决策树聚类方法通过创建空间中的超矩形区域，使得每个维度的阈值划分具有很好的直观性。换句话说，阈值树提供了一种简单的过程，通过少数几个特征即可解释为什么某个点属于其所在的聚类 [50]。此外，树的特征划分和决策路径有助于本文理解每个叶节点内成员的独特特征 [51]。

文献 [52] 提出了一种能够生成树结构的层次聚类方法，并给出了四种属性选择度量以及两种数据分割算法。与直接构建树结构的方法不同，CUBT [53] 通过在样本空间中随机添加辅助样本作为第二类别，以获取用于计算信息增益的标签，并通过调整分裂模式来优化决策过程。FDTC [54] 则结合了模糊规则与决策树结构，并以模糊规则作为节点分裂的依据。

此外，树的构建过程还可以利用传统聚类方法生成的伪标签 [55, 56]。这些研究从理论层面进行探讨，并以 K-Means 和 k-medians 的目标函数作为聚类质量评价指标，证明单次阈值切分能够在均值和中值上达到常数因子近似。后续工作进一步提升了近似比。与这些方法需要依赖 K-Means 聚类结果和预先确定的聚类数目类似，本文第四章节的方法也依赖序列模式来分裂节点。与基于数值特征进行节点分裂的传统方法相比，在序列数据场景中，基于模式的分裂方式更易于理解。

2.3 基于二值化神经网络的模式挖掘

近年来，深度神经网络（DNN）在许多领域表现出了强大的性能。然而，随着网络规模的不断增大，DNN 通常需要大量的存储和计算资源。为了解决低资源和嵌入式设备上的计算效率问题，许多参数量化方法应运而生 [57, 58]。其中，二值化技术作为一种 1 位量化方法，将数据限制为两个可能的值：-1（或 0）和 $+1$，通过量化网络

权重，显著降低了内存和计算需求，同时几乎不牺牲性能。

然而，传统的反向传播（BP）和随机梯度下降（SGD）方法并不适用于二值化网络，因为二值化函数不可微且其导数为零。为了解决这一问题，提出了一种名为“直通估计器（STE）”的启发式方法 [59]，通过绕过当前层的梯度，直接将二值权重的梯度传递给实值权重，从而使得像 SGD 或 Adam 等优化策略能够应用于更新实值权重。根据任务的不同，已经提出了多种不同的梯度估计方法。

尽管这些二值化网络结构通过将权重和输入之间的连接离散化，能够在一定程度上降低计算复杂度，但它们并未直观地学习与输入相关的可解释结构。在本研究中，本文借鉴了二值化神经网络（BNN）的方法，训练了一个基于卷积神经网络（CNN）的自编码器，以挖掘具有可解释性的模式结构。

在模式挖掘领域，二值化神经网络（BNN）已经广泛应用，尤其在分类规则挖掘方面取得了显著成果[60–63]。CRS[60] 通过梯度下降和随机二值化技术高效学习规则集，但在处理大规模数据集时存在局限性。为解决这一问题，研究者提出了基于规则的表示学习器（RRL）[62]，该方法在保证效率的同时成功扩展至大规模数据集应用场景。NR-Net [61] 则通过构造特殊的神经网络训练问题，学习准确且可解释的决策规则集，并设计了一种能够直接映射为可解释规则集的两层网络架构。在此基础上，CT-Net [63] 引入了阈值设置机制，进一步提升了表格数据分类任务的性能。

尽管这些方法在非序列数据上取得了显著的进展，但它们并未专门针对序列数据进行优化。随着神经网络技术的发展，近年来也出现了一些针对序列模式挖掘的神经网络方法 [64–66]，然而这些方法并未专门解决序列分类问题。

在序列分类任务中，CR2N [67] 结合了 1D 卷积神经网络（1DCNN）和基础规则模型架构，能够发现局部或全局模式的分类规则。与 CR2N 方法类似，本文的方法同样采用了 1DCNN 架构，但主要的不同之处在于，本文挖掘的是一组判别性子串，而非 CR2N 中的全局析取规则。为了实现这一目标，本文提出了一种特殊的二值化策略，使得每个卷积核的权重在训练过程中得以更新，并且这些权重可以被可解释地转换为一段 k-mers 子串，从而更好地揭示模型的判别依据。

3 权重可解释的序列分类神经网络

本章提出了一种基于可解释二值化权重的序列分类神经网络，通过特殊的二值化策略将神经网络的权重转化为可理解的 k-mer 模式，在保持较高分类性能的同时，提供了模型决策的清晰解释，并给出了相应的基于 Hamming 距离的序列相似性度量方法。

3.1 问题定义

在序列分类的神经网络架构中，1DCNN 是一个常用的模型。给定一个权重矩阵 **w** 和一个 one-hot 序列矩阵 **x**：

$$\mathbf{x}_{i,j} = \begin{cases} 1, & \text{if } \sigma_j \text{ is the } i\text{-th item of } I, \\ 0, & \text{Otherwise.} \end{cases} \tag{3.1}$$

其中，**x** 的大小为 $|I| \times |s|$，其中 $|\mathcal{I}|$ 是集合 I 中唯一项目的数量，$|s|$ 是序列 s 的长度。卷积操作后的输出 **z** 可以通过公式3.2得到：

$$\mathbf{z} = \mathbf{w} \otimes \mathbf{x} + \mathbf{b}, \tag{3.2}$$

其中，$\otimes$ 表示卷积操作。在这种情况下，权重矩阵 **w** 是一个卷积核，用于提取序列 **x** 中的特征。然而，经典的 1DCNN 神经网络卷积核的权重是实数，对于人类来说，这些权重是高度复杂、冗余并且不可解释的。

为了压缩模型、减小冗余并且提高模型的适用性，训练二值化神经网络的策略被提出，即将权重矩阵 **w** 二值化为 $-1(0)$ 和 $+1$ 。对于二值化，现有的 BNNs 通常应用一个 $sign$ 函数：

$$sign(x) = \begin{cases} +1, & \text{if } x \geq 0, \\ -1, & \text{otherwise.} \end{cases} \tag{3.3}$$

通过量化，输出 **z** 可以表示为：

$$\mathbf{z} = sign(\mathbf{w}) \otimes sign(\mathbf{x}) + \mathbf{b}. \tag{3.4}$$

然而，这种二值化策略并不能生成可解释的结构，虽然对比实值的全精度神经网络，二值化神经网络可以减少模型的大小和计算量，但是其卷积核的权重仍然是只是一堆由 -1 和 $+1$ 组成的矩阵。由于每个矩阵依然是多个模式的复杂组合，这些矩阵对于人类来说是不可理解和不可解释的。因此，本章提出了一种新的基于可解释二值

化权重的序列分类神经网络，并且给出了对应的基于 Hamming 距离的序列相似性度量方法。

3.2 算法设计

对于给定的序列数据库，本方法的目的是找到一组具有区分性的 k-mer 用于序列分类。特别地，本方法希望尽可能找到一些具有高区分性的模式组合。在神经网络结构中，编码器通常用于提取下游任务的特征，而基于卷积神经网络的编码器已经展示了其强大的特征提取能力用于序列分类。然而，神经网络中输入和神经元之间的连接通常是非符号和非线性的，这使得神经网络对于人类来说难以理解。

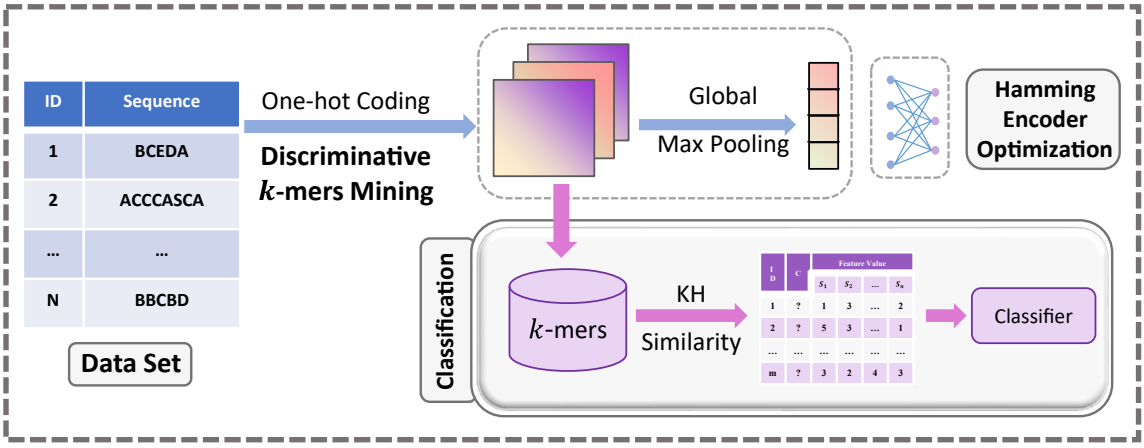


图 3.1　Hamming 编码器的工作流程

Fig. 3.1　Workflow of the Hamming Encoder

本章提出 Hamming 编码器，其中权重在前向传播过程中被二值化为 0 和 1。这是一个可解释的 CNN 编码器，其中每个卷积核可以被转换为一个 k-mer。因此，本文将挖掘具有区分性的模式作为 CNN 编码器的优化问题。Hamming 编码器的工作流程如图3.1所示。数据集中的每个序列首先被转换为一个 $|I| \times |s|_{max}$ 的 one-hot 矩阵，其中 $|\mathcal{I}|$ 是项目集的大小，$|s|_{max}$ 表示数据集中的最大序列长度。为了获得一个强大的 Hamming 编码器，本方法添加了一个全连接层，因为它代表了互相组合后的特征信息。在使用损失函数优化 Hamming 编码器之后，其权重被转换为一个 k-mer 集合。随后，本方法采用 KH 相似度[68]，该相似度计算序列和具有区分性的 k-mer 之间的相似性以生成特征向量。KH 相似度等价于网络结构中使用的全局最大池化操作，通过将具有更多信息的特征向量转换为 k-mer，增强了挖掘到的 k-mer 的区分能力。

3.2.1 Hamming 编码器

Hamming 编码器的结构是一个特殊的二值化 CNN 层，后面跟着一个全精度的 FNN 层。二值化 CNN 层的每个卷积核（或通道）在前向过程中具有二进制权重 $\mathbf{w}_b$，

Algorithm 3.1 Hamming 编码器挖掘

输入: 序列数据库 D，初始权重 $\mathbf{w}^1$，迭代次数 e_{max}，批大小 N，项目集 I。
输出: 具有区分性的 k-mer 集合 P。

$\mathbf{t} \leftarrow 1$ ▷ 步骤
$\mathbf{x} \leftarrow$ one-hot(I, D)
for $e = 1 \ldots e_{max}$ **do** ▷ 迭代次数
  for $i = 1 \ldots n$ **do** ▷ 对每个批次
    //前向传播
    $\mathbf{w}^t_{bcnn} \leftarrow \mathbf{Q}(\mathbf{w}^t_{cnn})$; ▷ 二值化权重
    $\mathbf{z} \leftarrow \mathbf{w}^t_{bcnn} \otimes \mathbf{x}_i$; ▷ 卷积
    $\mathbf{k} \leftarrow$ GlobalMaxPooling$(\mathbf{z})$;
    $\mathbf{y_{logits}} \leftarrow \mathbf{k}\mathbf{w}^T_{fnn}$ ▷ FNN 层
    $\hat{\mathbf{y}} \leftarrow$ SoftMax$(\mathbf{y}_{logits})$
    $\mathcal{L} \leftarrow$ loss$(D[i]; \hat{\mathbf{y}})$; ▷ 计算损失
    //反向传播
    $g_{\mathbf{w}_{cnn}} \leftarrow$ backward$(\mathcal{L})$;
    $\mathbf{w}^{t+1}_{cnn} \leftarrow$ update$(\mathbf{w}^t_{cnn}, g_{\mathbf{w}_{cnn}})$; ▷ 优化
  end for
end for
$P \leftarrow$ inverse one-hot$(I, \mathbf{w}_{bcnn})$ ▷ 提取 k-mer
输出: P

在反向传播过程中具有连续版本的权重 $\mathbf{w}$，其中二进制权重 $\mathbf{w}_b$ 可以被转换为 k-mer。通过优化编码器，可以获得具有区分性的 k-mer 集合。算法3.1展示了 Hamming 编码器训练过程的细节。接下来的章节描述了如何在前向传播和反向传播过程中进行二值化。

3.2.2 前向传播

在本方法中，权重矩阵中的具体值并不重要。具体来说，对于给定的权重矩阵 $\mathbf{w} \in \mathcal{R}^{|\mathcal{I}|\times k}$，本方法使用其二值化权重 $\mathbf{w}_b$ 如公式3.5，其中 $\mathbf{w}_b$ 的元素在原始矩阵 $\mathbf{w}$ 的每一列中的最大值对应位置被设置为 1，否则为 0，如下所示：

$$\mathbf{Q}(\mathbf{w})_{i,j} = \begin{cases} 1, & \text{if } i = \arg\max_k \mathbf{w}_{k,j} \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

因此，对于一个序列 $\mathbf{x} \in \mathcal{R}^{|\mathcal{I}|\times|s|_{max}}$ 的 one-hot 序列矩阵，二值化 CNN 层的输出为：

$$\mathbf{z} = \mathbf{w_b} \otimes \mathbf{x}. \tag{3.6}$$

随后，对二值化 CNN 层的输出应用全局最大池化操作，以获得特征值 α。本方法将展示该值等价于根据第3.2.5节中定义的 KH 相似度计算的特征值。

$$\alpha = \max_j(\mathbf{z}_j). \tag{3.7}$$

对于多个卷积核，本方法获得多个对应的值，这些值被连接起来形成特征向量 **a**。然后，特征向量 **a** 被输入到一个全连接神经网络（FNN）中，以获得输出 $\mathbf{y}_{logtis}$：

$$\mathbf{y}_{logtis} = \mathbf{a}\mathbf{w}_{fnn}^T + \mathbf{b}. \tag{3.8}$$

为了将 $\mathbf{y}_{logits}$ 转换为概率，本方法应用 softmax 函数。最终输出 $\hat{\mathbf{y}}$ 如下：

$$\hat{\mathbf{y}}_i = \frac{\exp(\mathbf{y}_{logits,i})}{\sum_j \exp(\mathbf{y}_{logits,j})}. \tag{3.9}$$

3.2.3 反向传播

为了挖掘具有区分性的 k-mer，本方法最小化损失函数 $\mathcal{L}$。通过将输出 $\hat{\mathbf{y}}$ 和真实标签 **y** 作为输入，交叉熵损失函数定义如下：

$$\mathcal{L} = -\sum_i \mathbf{y}_i \log(\hat{\mathbf{y}}_i). \tag{3.10}$$

通过批次的损失值，本方法可以计算损失函数相对于二值化权重 $\mathbf{w}_b$ 的梯度。然而，计算权重 **w** 的梯度仍然是一个问题，因为二值化函数是不可微的。为了更新权重，本方法采用了 Straight-Through Estimator (STE) [59] 来估计量化的反向梯度。STE 将传入的梯度分配给阈值操作的传出梯度，通过在反向传播过程中将二值化函数的梯度设置为 1，简化了过程。因此，本方法有：

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial \mathbf{w}_b} \cdot \frac{\partial \mathbf{w}_b}{\partial \mathbf{w}}, \tag{3.11}$$

其中，在 STE 下，$\frac{\partial \mathbf{w}_b}{\partial \mathbf{w}} = 1$。因此，梯度简化为：

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial \mathbf{w}_b}. \tag{3.12}$$

STE 操作应用于 Hamming 编码器层的权重矩阵 **w**，损失函数相对于 Hamming 编码器层的权重的梯度计算如下：

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_b} = \underbrace{\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{y}}}}_{\text{Cross-Entropy Loss}} \cdot \underbrace{\frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{y}_{logits}}}_{\text{SoftMax}} \cdot \underbrace{\frac{\partial \mathbf{y}_{logits}}{\partial \mathbf{a}}}_{\text{FNN layer}} \cdot \underbrace{\frac{\partial \mathbf{a}}{\partial \mathbf{z}}}_{\text{Global Max Pooling}} \cdot \underbrace{\frac{\partial \mathbf{z}}{\partial \mathbf{w}_b}}_{\text{Convolution}}. \tag{3.13}$$

考虑全局最大池化位置，梯度的详细表达式如下：

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_b} = \left(\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{y}}} \cdot \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{y}_{logits}} \cdot \frac{\partial \mathbf{y}_{logits}}{\partial \mathbf{a}} \cdot \frac{\partial \mathbf{z}_{\max}}{\partial \mathbf{w}_b} \right). \tag{3.14}$$

其中，全局最大池化层的梯度定义如下：

$$\frac{\partial \mathbf{a}}{\partial \mathbf{z}_j} = \begin{cases} 1, & \text{if } j = \arg\max_j(\mathbf{z}_j), \\ 0, & \text{otherwise.} \end{cases} \tag{3.15}$$

最后，考虑到只有全局最大池化选择的索引位置会对梯度更新产生影响，本方法可以简化梯度为：

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_b} = \left(\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{y}}} \cdot \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{y}_{logits}} \cdot \frac{\partial \mathbf{y}_{logits}}{\partial \mathbf{a}} \cdot \frac{\partial \mathbf{z}_{\max}}{\partial \mathbf{w}_b} \right). \tag{3.16}$$

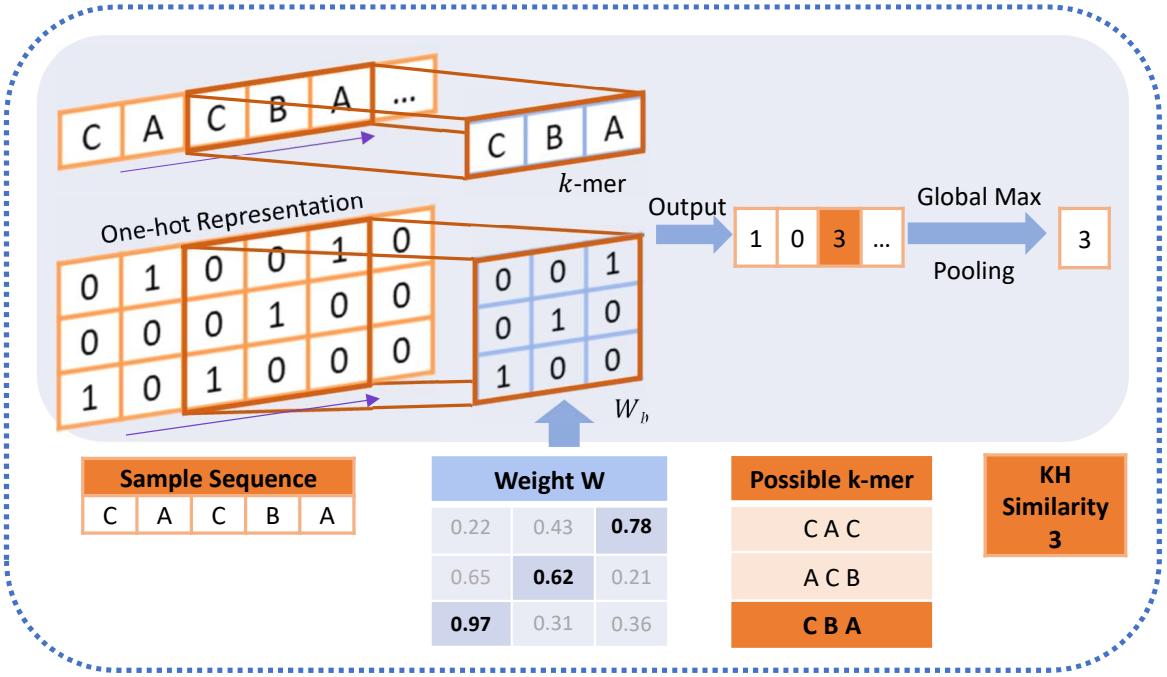


图 3.2　KH 相似度与全局最大池化的一致性示例

Fig. 3.2　Illustration of the consistency between KH similarity and global max pooling

3.2.4　基于 k-mer 的序列特征提取

对于一个优化完成的 Hamming 编码器，本方法首先从网络参数中读取权重 $\mathbf{w}$，然后对其应用二值化策略 $\mathbf{Q}(\cdot)$。这使本方法可以获得在前向传播过程中计算的二值化权重，记为 $\mathbf{w}_b$。随后，本方法根据项目集 I 将矩阵的权重转换为 k-mer，遵循与将输入序列转换为 one-hot 矩阵相同的顺序。因此，对于每个 $\mathbf{w}_b$，本方法可以找到一个对应

的 k-mer p：

$$p = \langle i_{j_1}, i_{j_2}, \ldots, i_{j_k} \rangle \quad \text{where} \quad \mathbf{w}_{j_1,1} = 1, \mathbf{w}_{j_2,2} = 1, \ldots, \mathbf{w}_{j_k,k} = 1, \tag{3.17}$$

其中，i_j 表示元素 I 中的第 j 个元素，$\mathbf{w}_{j,n}$ 表示权重矩阵 $\mathbf{w}_b$ 的第 j 行和第 n 列的元素。该公式表示对于矩阵 $\mathbf{w}_b$ 的每个第 n 列，找到值为 1 的第 j 行，并将对应于该行的元素添加到 k-mer p 的第 n 个位置。

下面是将权重矩阵 $\mathbf{w}_{cnn}$ 转换为 k-mer p 的示例。

$$\mathbf{w}_{cnn} = \begin{pmatrix} 0.22 & 0.43 & \mathbf{0.78} \\ 0.65 & \mathbf{0.62} & 0.21 \\ \mathbf{0.97} & 0.31 & 0.36 \end{pmatrix}, \mathbf{w}_b = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

对于给定的项目集 $I = \{A, B, C\}$，权重矩阵 $\mathbf{w}_b$ 被转换为 $\langle CBA \rangle$。

通常，在处理序列数据时，找到具有区分性的 k-mer 首先涉及生成一个可能非常大的 k-mer 集合。然后，使用这个 k-mer 集合通过相应的相似性函数将所有序列转换为向量，之后可以进行特征选择。然而，尝试找到更长的 k-mer 可能会导致内存问题，使得转换变得不可行。

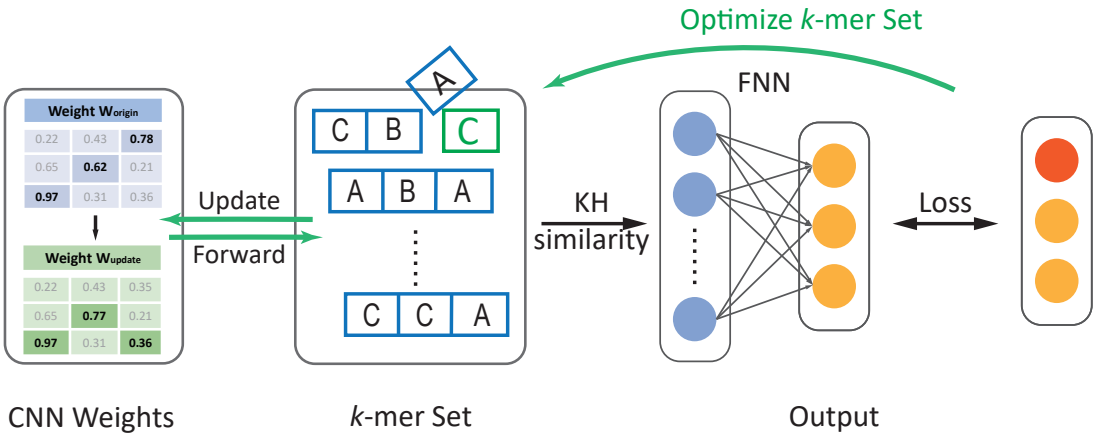


图 3.3　Hamming 编码器的 k-mer 集合搜索过程

Fig. 3.3　The k-mer set search process of the Hamming Encoder

相比之下，本方法提出的方法利用前向传播期间的梯度更新来优化 CNN 权重，从而避免了一次性将数据集转换为所有 k-mer 的需求。在本方法的二值化方法下，CNN 层的权重可以直接转换为特定的 k-mer，因此找到具有区分性的 k-mer 集合的任务被转换为优化 CNN 层。在 Hamming 编码器的前向传播过程中，首先将序列输入到 CNN 层，然后使用全局最大池化策略。然后，附加全连接层利用这些特征进行分类。

找到具有区分性的 k-mer 集合的具体过程如图3.3所示。CNN 层的权重将根据每个批次的损失进行局部更新，这对应于 k-mer 的局部变化（从 $\langle CBA\rangle$ 到 $\langle CBC\rangle$）。通过利用神经网络的梯度优化，本方法最终得到了一组具有强大区分能力的固定长度的 k-mer。

然而，本方法也有一些局限性。当所有输入序列特别长或项目集特别大时，卷积核的矩阵可能会变得非常大，导致时间复杂度很高。同时，选择 k 是非常重要的，需要手动指定，因为这个参数对模型的性能有很大影响。

3.2.5 基于 Hamming 距离的序列相似性度量

为了确保 k-mer 转换为特征向量和在全局最大池化操作后网络中获得的值之间的一致性，本方法采用了 k-mer Hamming (KH) 相似度度量[68]。该度量涉及计算滑动窗口的子串和 k-mer $p \in P$ 之间的 Hamming 距离，如下所示：

$$\mathrm{Sim}(p, s) = k - \min_{t \in S_k(s)} \{H(t, p)\}, \tag{3.18}$$

其中，$S_k(s)$ 表示实例 s 中所有长度为 k 的子串的集合，$H(t, p)$ 表示 k-mer 序列 p 和子串 t 之间的 Hamming 距离：

$$H(t, p) = \sum_{i=1}^{k} (t[i] \oplus p[i]), \tag{3.19}$$

其中，$\oplus$ 表示异或操作。

定理 1. 给定一个卷积核权重 $\mathbf{w}_b$ 及其对应的 k-mer p 在 one-hot 编码下，对于任意给定的序列 **s** 及其 one-hot 矩阵 **x**，KH 相似度等于 **x** 与 $\mathbf{w}_b$ 进行卷积操作后的全局最大池化输出：

$$\mathrm{Sim}(p, s) = \max_{j}(\mathbf{z}_j). \tag{3.20}$$

证明. 假设 $\mathbf{w}_b = \{w_1, w_2, \ldots, w_k\}$, 其中 w_i 是 $\mathbf{w}_b$ 的第 i 列向量，可以根据公式 (3.17) 转换为 p_i。同样，**x** 也可以表示为列向量的组合 $\mathbf{x} = \{x_1, x_2, \ldots, x_{|s|}\}$，那么卷积操作的输出为，

$$\mathbf{z}_i = \sum_{i=1}^{|s|-k+1} \sum_{j=0}^{k-1} x_{i+j} \cdot w_i. \tag{3.21}$$

已知 x_i 是 one-hot 向量，权重矩阵 $\mathbf{w}_b$ 可以使用相同的编码关系转换为 p。基于 one-hot 编码的正交性，两个 one-hot 向量的点积当且仅当两个向量相同时等于 1。也

就是说，卷积操作实际上是在计算卷积核宽度 k 中相同向量的数量：

$$\mathbf{z}_i = k - H(\mathbf{x}_{i:i+k-1}, \mathbf{w}_b). \tag{3.22}$$

随后，根据公式 (3.7) 采用最大池化操作：

$$\alpha = \max_{i \in |s|-k+1} \{\mathbf{z}_i\} = k - \min_{i \in |s|-k+1} \{H(\mathbf{x}_{i:i+k-1}, \mathbf{w}_b)\}. \tag{3.23}$$

通过滑动窗口，$\mathbf{x}_{i:i+k-1}$ 实际上可以转换为序列中的 k-mer t，$\mathbf{w}_b$ 也可以逆向编码为 p。因此，本文提出的 KH 相似度等价于全局最大池化操作。 □

在图3.2中，本方法说明了如何计算 KH 相似度及其与全局最大池化操作的一致性。具体来说，本方法将权重矩阵 **w** 二值化以获得 $\mathbf{w_b}$ 矩阵，该矩阵用于计算从序列 $CACBA$ 的 one-hot 编码导出的矩阵 **x** 的输出 **z**。通过对输出 **z** 执行全局最大池化操作，本方法获得特征值 α，该特征值等价于 KH 相似度度量中的特征值。对于这个例子，其中 $k = 3$，$p = \langle CBA \rangle$，$s = CACBA$，$S_k(I) = \{CAC, ACB, CBA\}$，Hamming 距离计算得到 KH 相似度值为 3。

3.3 实验

在本节中，本方法首先在十一个基准数据集上评估本方法提出的方法的性能，并将其与几种基于特征的序列分类算法进行比较。为了研究 Hamming 编码器与基线 CNN 以及广泛使用的 BCNN 之间的区别，本方法还进行了一系列实验。实验在一台配备 NVIDIA A6000 GPU 和 Intel Xeon Gold 6226R CPU 的 Ubuntu 服务器上进行，内存为 512 GB。所有报告的准确性都是通过重复 5 折交叉验证 10 次来记录平均结果获得的。在评估 Seq2Pat 和 CR2N-based 方法处理超过 2 类数据集时，本方法采用 One-vs-One 策略来扩展二元分类器。

为了评估所提出方法的性能，本方法使用以下十一个数据集作为基准：Aslbu [69]，Auslan2 [69]，Context [70]，Epitope [71]，Gene [72]，Pioneer [69]，Question，Robot [73]，Skating [69]，Reuters [73]，和 Unix [73]。这些数据集的详细信息总结在表3.1中，其中 $|\mathcal{D}|$ 表示序列的数量，$|\mathcal{I}|$ 是项目的数量，min$l$ 和 max$l$ 分别是序列的最小和最大长度，$|C|$ 是类的数量。

3.3.1 算法性能

本文通过将提出的方法与以下最先进的序列分类方法进行比较来评估其性能：MiSeRe [12]、Sqn2Vec [16]、Seq2Pat [14]、SGT [28]、Gokrimp [74]、iBCM [13]、SeqDT [17]、SCIP [9] 和 CR2N [67]。选择对比的序列分类方法的主要基于以下依据。首先，本文选

表 3.1　性能比较中使用的数据集特征
Tab. 3.1　Characteristics of 数据集 s used in performance comparison

数据集	$\|\mathcal{D}\|$	$\|\mathcal{I}\|$	min$l$	max$l$	$\|C\|$
Aslbu	424	250	2	54	7
Auslan2	200	16	2	18	10
Context	240	94	22	246	5
Epitope	2392	20	9	21	2
Gene	2942	5	41	216	2
Pioneer	160	178	4	100	3
Question	1731	3612	4	29	2
Robot	4302	95	24	24	2
Skating	530	82	18	240	7
Reuters	1010	6380	4	533	4
Unix	5472	1697	1	1400	4

择相对较新的方法，以确保相关性并反映该领域的当前进展。其次，本文选择了与所提出方法类似、能够提取可被不同分类器利用的特征的方法。这确保了比较保持在同一类别内，因为本方法同样专注于用于分类的特征提取。

选择超参数的原则包括遵循推荐的设置和控制与本文的方法相同数量的模式。由于一些方法具有不同的超参数要求，本文相应地做了调整。对于像 SGT、Sqn2Vec、SeqDT、CR2N、HMM 和 SCIP 的四个变体这样提供默认参数的方法，本方法使用了它们手稿中推荐的默认参数。Gokrimp 是无参数的。对于 MiSeRe，本方法控制了挖掘模式的最大数量，使用与本方法相同的数量。对于 iBCM 和 Seq2Pat，本方法选择了适当的支持值，以避免生成过多的模式。这些方法的详细参数设置如下：

MiSeRe 首先从训练数据中挖掘序列规则，然后将其用作向量数据分类算法的输入。$execution\_time$ 和 max_rule 分别固定为 5 分钟和 1024。

Sqn2Vec 是一种序列嵌入方法，使用挖掘的模式来训练向量表示。本方法为其两个变体 Sqn2VecSEP 和 Sqn2VecSIM 设置 $min\_sup = 0.05$，$d = 128$ 和 $maxgap = 4$。

SGT 是一种基于图的特征嵌入方法，用于将序列转换为特征向量，本方法使用默认参数。

iBCM 基于行为模板挖掘序列模式进行序列分类。本方法为所有数据集设置 $min\_sup = 0.1$，$no\_win = 1$ 和 $reduce\_feature\_space = True$。

SeqDT 集成了相关的序列模式挖掘和决策树构建。本方法使用默认参数，其中 $maxL = 4$，$g = 1$，$pru = ture$，$\epsilon = 1$，$minS = 0.0$，$minN = 2$，$maxD = 0$。

SCIP 是一种基于模式的序列分类算法，有四个变体：SCII_HAR、SCII_MA、SCIS_HAR 和 SCIS_MA。实验中固定以下参数：$minint = 0.02$，$minsup = 0.05$，$maxsize = 3$，$conf = 0.5$ 和 $topk = 11$。

HMM 模型基于[75] 中的开源 Python 实现。与其一致，本文为每个类别训练了一

个单独的 HMM，并通过选择具有最高概率的 HMM 对序列进行分类。HMM 使用 Baum-Welch 算法进行训练，停止阈值为 0.001，最大迭代次数为 10^6。

CR2N 是一个用于发现序列数据中的局部或全局模式的 1D 卷积神经网络，同时学习二元分类规则。对于局部模型和全局模型，$lr = 0.1$，$epochs = 200$，$\lambda = 10^{-5}$。

Seq2Pat 是一个用于序列到模式生成的研究库，旨在挖掘数据集中正负类别之间的模式。在本方法的实验中，本方法设置参数如下：$min\_freq = 0.2$，$max\_len = 5$，$max\_span = 10$。

Gokrimp 是一种无参数的方法，它基于最小描述长度（MDL）原则识别一组非冗余的压缩序列模式。

在本文中初始的 k-mer 数量设置为 1024，批大小设置为 64。k-mer 的长度是一个用户指定的参数，可能会显著影响特定数据集上的模型性能，对于本方法的实验，本方法评估了 $k = 2$ 和 $k = 5$ 时不同模型性能。本方法建议根据序列长度分布选择值。对于具有相对较多长序列的数据集，本方法建议选择较大的 k，对于具有相对较多短序列的数据集，本方法建议选择较小的 k。使用 Adam 优化器，学习率为 0.0003，权重衰减为 $1e-5$。为了防止数据泄漏，所有训练过程都是在训练集上独立进行的。在训练过程中，报告每个时代的平均交叉熵损失，并选择训练过程中损失最低的模型。将迭代次数设置为 100。本方法使用 Scikit-Learn 库中的支持向量机、决策树（SVM、DT）、K-最近邻（KNN）和朴素贝叶斯（NB）评估使用不同技术挖掘的序列模式的可区分性。

不同基于特征的方法的分类准确率列在表3.2中。其中，SGT 的部分结果为空，这是因为其内存需求超过了实验计算机的上限；而 Gokrimp 的一些结果为空，则是由于它未能成功输出序列模式。此外，本方法还采用端到端方式将所提出的方法与其他报告单一值的方法进行了比较，这些实验结果可在表3.3中查看。根据实验结果，有以下关键发现。

首先，本方法在大多数数据集上表现出相对竞争力。为了评估整体性能，本方法计算了四种不同分类器的平均准确率和排名，这些结果显示在表3.2的最后两行中。结果表明，与其他方法相比，本方法在 SVM 和 NB 分类器上表现出竞争力。然而，对于 DT 和 KNN 分类器，本方法并没有明显的性能提升。

为了量化本方法的性能表现，本研究采用了 Bonferroni-Dunn 检验[76] 来验证一个零假设：即本方法与各竞争算法在分类准确率方面具有相同的性能水平。在显著性水平为 0.05 且数据集数量为 11 的条件下，临界差值 (CD) 为 3.18。当两种方法的平均排名差异超过 CD 值时，可认为它们之间的性能差距在统计学上具有显著性。图3.4展示了各” 端到端” 方法的平均排名及显著性测试结果。从图中可以看出，虽然 Hamming 编码器未能在统计上显著优于所有对比方法，但它仍然明显优于 CR2N（全局）、CR2N

表 3.2　Hamming 编码器、MiSeRe、iBCM、Sqn2Vec、Seq2Pat、Gokrimp 和 SGT 的分类准确率（标准差）

Tab. 3.2　Performance comparison of Hamming Encoder, MiSeRe, iBCM, Sqn2Vec, Seq2Pat, Gokrimp, and SGT in terms of classification accuracy (standard deviation)

		Hamming Encoder $k=2$	Hamming Encoder $k=5$	MiSeRe	iBCM	Sqn2Vec SEQ	Sqn2Vec SIM	Seq2Pat	Gokrimp	SGT
aslbu	SVM	**0.639** (0.011)	0.580 (0.016)	0.574 (0.012)	0.400 (0.038)	0.438 (0.004)	0.602 (0.008)	0.451 (0.007)	0.479 (0.000)	0.431 (0.008)
	NB	**0.588** (0.010)	0.494 (0.019)	0.536 (0.008)	0.531 (0.042)	0.540 (0.006)	0.548 (0.010)	0.449 (0.012)	0.497 (0.000)	0.383 (0.001)
	KNN	0.593 (0.016)	0.509 (0.022)	0.560 (0.017)	0.501 (0.044)	0.519 (0.011)	0.570 (0.005)	0.384 (0.026)	0.495 (0.000)	0.412 (0.015)
	DT	**0.547** (0.016)	0.478 (0.025)	0.541 (0.011)	0.492 (0.058)	0.441 (0.018)	0.484 (0.010)	0.445 (0.012)	0.472 (0.002)	0.513 (0.014)
auslan2	SVM	0.273 (0.016)	0.237 (0.018)	**0.285** (0.027)	0.274 (0.021)	0.256 (0.016)	0.255 (0.019)	0.070 (0.014)	0.230 (0.000)	0.265 (0.017)
	NB	0.295 (0.009)	0.232 (0.011)	0.283 (0.017)	0.281 (0.027)	0.274 (0.019)	**0.305** (0.016)	0.112 (0.011)	0.230 (0.000)	0.285 (0.017)
	KNN	0.251 (0.025)	0.201 (0.026)	**0.293** (0.019)	0.252 (0.026)	0.257 (0.015)	0.270 (0.019)	0.145 (0.004)	0.160 (0.000)	0.240 (0.015)
	DT	0.264 (0.018)	0.221 (0.012)	**0.304** (0.021)	0.270 (0.031)	0.280 (0.035)	0.250 (0.022)	0.088 (0.013)	0.225 (0.000)	0.281 (0.018)
context	SVM	0.896 (0.011)	**0.949** (0.007)	0.917 (0.009)	0.906 (0.007)	0.927 (0.006)	0.928 (0.011)	0.930 (0.009)	0.863 (0.000)	0.927 (0.006)
	NB	0.843 (0.011)	0.834 (0.015)	**0.907** (0.007)	0.891 (0.008)	0.862 (0.010)	0.874 (0.010)	0.890 (0.007)	0.863 (0.000)	0.905 (0.007)
	KNN	0.852 (0.012)	**0.927** (0.012)	0.902 (0.007)	0.663 (0.014)	0.876 (0.016)	0.833 (0.017)	0.883 (0.013)	0.833 (0.000)	0.719 (0.012)
	DT	0.826 (0.017)	0.846 (0.019)	**0.880** (0.017)	0.802 (0.024)	0.590 (0.031)	0.524 (0.038)	0.849 (0.021)	0.830 (0.013)	0.833 (0.016)
epitope	SVM	0.858 (0.004)	**0.924** (0.004)	0.820 (0.004)	0.715 (0.006)	0.846 (0.003)	0.850 (0.004)	0.803 (0.004)	N.A	0.921 (0.005)
	NB	0.715 (0.004)	0.718 (0.004)	0.583 (0.002)	0.570 (0.001)	0.726 (0.007)	0.721 (0.004)	0.679 (0.003)	N.A	**0.729** (0.001)
	KNN	0.809 (0.004)	**0.877** (0.004)	0.810 (0.005)	0.703 (0.012)	0.847 (0.004)	0.843 (0.006)	0.767 (0.006)	N.A	0.822 (0.004)
	DT	0.863 (0.006)	**0.869** (0.006)	0.856 (0.005)	0.793 (0.009)	0.783 (0.009)	0.783 (0.009)	0.854 (0.004)	N.A	0.861 (0.006)
gene	SVM	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	N.A	**1.000** (0.000)
	NB	**1.000** (0.000)	0.961 (0.064)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	N.A	**1.000** (0.000)
	KNN	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	N.A	**1.000** (0.000)
	DT	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	0.999 (0.000)	0.960 (0.006)	0.963 (0.005)	**1.000** (0.000)	N.A	**1.000** (0.000)
pioneer	SVM	0.964 (0.009)	**0.989** (0.007)	0.958 (0.004)	0.804 (0.013)	0.959 (0.014)	0.984 (0.005)	0.938 (0.035)	0.881 (0.000)	0.879 (0.016)
	NB	0.923 (0.007)	0.935 (0.009)	0.787 (0.004)	0.848 (0.019)	0.909 (0.019)	**0.940** (0.010)	0.899 (0.044)	0.794 (0.000)	0.775 (0.006)
	KNN	0.959 (0.013)	**0.989** (0.006)	0.959 (0.010)	0.661 (0.023)	0.951 (0.013)	0.959 (0.009)	0.902 (0.032)	0.825 (0.000)	0.611 (0.036)
	DT	0.982 (0.016)	0.938 (0.019)	0.976 (0.012)	0.956 (0.017)	0.834 (0.045)	0.781 (0.015)	0.956 (0.014)	0.931 (0.005)	**0.984** (0.013)
question	SVM	0.943 (0.003)	**0.955** (0.003)	0.898 (0.005)	0.842 (0.003)	0.803 (0.006)	0.884 (0.003)	0.611 (0.005)	0.837 (0.000)	0.936 (0.009)
	NB	0.910 (0.004)	**0.923** (0.003)	0.852 (0.001)	0.831 (0.000)	0.715 (0.013)	0.779 (0.010)	0.606 (0.001)	0.794 (0.000)	0.861 (0.012)
	KNN	**0.924** (0.005)	0.921 (0.005)	0.843 (0.003)	0.844 (0.004)	0.841 (0.006)	0.876 (0.003)	0.591 (0.002)	0.756 (0.000)	0.899 (0.013)
	DT	0.932 (0.004)	**0.940** (0.005)	0.897 (0.005)	0.866 (0.005)	0.728 (0.010)	0.826 (0.007)	0.607 (0.005)	0.843 (0.000)	0.899 (0.007)
robot	SVM	0.912 (0.002)	**0.952** (0.002)	0.936 (0.003)	0.916 (0.002)	0.925 (0.002)	0.926 (0.002)	0.831 (0.002)	0.873 (0.000)	0.933 (0.002)
	NB	0.748 (0.003)	0.779 (0.005)	0.829 (0.001)	0.838 (0.002)	0.772 (0.006)	0.785 (0.003)	0.719 (0.006)	0.784 (0.000)	**0.861** (0.001)
	KNN	0.909 (0.003)	**0.935** (0.002)	0.919 (0.002)	0.894 (0.002)	0.937 (0.002)	0.934 (0.002)	0.819 (0.003)	0.849 (0.000)	0.894 (0.002)
	DT	0.876 (0.003)	0.892 (0.005)	0.890 (0.006)	0.936 (0.002)	0.783 (0.007)	0.769 (0.005)	0.843 (0.004)	0.851 (0.002)	**0.899** (0.004)
skating	SVM	0.293 (0.011)	**0.342** (0.018)	0.259 (0.010)	0.241 (0.016)	0.319 (0.012)	0.324 (0.019)	0.290 (0.009)	0.258 (0.000)	0.316 (0.017)
	NB	0.282 (0.012)	0.275 (0.011)	0.258 (0.016)	0.250 (0.009)	**0.285** (0.017)	**0.285** (0.021)	0.240 (0.008)	0.236 (0.000)	0.250 (0.014)
	KNN	0.243 (0.008)	**0.308** (0.014)	0.232 (0.018)	0.242 (0.009)	0.299 (0.011)	0.302 (0.011)	0.248 (0.014)	0.226 (0.000)	0.228 (0.012)
	DT	0.272 (0.017)	0.270 (0.010)	**0.280** (0.011)	0.238 (0.014)	0.200 (0.015)	0.204 (0.019)	0.259 (0.016)	0.220 (0.008)	0.257 (0.019)
reuters	SVM	0.956 (0.004)	0.975 (0.002)	0.953 (0.003)	0.832 (0.003)	**0.980** (0.001)	0.978 (0.002)	0.868 (0.005)	0.741 (0.000)	N.A
	NB	0.923 (0.006)	**0.927** (0.003)	0.865 (0.002)	0.869 (0.003)	0.927 (0.005)	0.917 (0.005)	0.847 (0.003)	0.733 (0.000)	N.A
	KNN	0.934 (0.005)	0.948 (0.004)	0.893 (0.006)	0.762 (0.011)	**0.956** (0.003)	0.940 (0.004)	0.854 (0.003)	0.660 (0.000)	N.A
	DT	0.906 (0.007)	**0.908** (0.008)	0.905 (0.007)	0.886 (0.007)	0.786 (0.018)	0.687 (0.017)	0.843 (0.002)	0.732 (0.001)	N.A
unix	SVM	0.891 (0.002)	0.716 (0.002)	0.898 (0.001)	0.866 (0.001)	**0.909** (0.001)	0.895 (0.002)	0.609 (0.001)	0.741 (0.000)	N.A
	NB	0.838 (0.004)	0.627 (0.015)	0.728 (0.006)	**0.890** (0.001)	0.829 (0.008)	0.775 (0.006)	0.443 (0.006)	0.733 (0.000)	N.A
	KNN	0.853 (0.007)	0.685 (0.007)	0.863 (0.003)	0.829 (0.002)	**0.891** (0.002)	0.887 (0.003)	0.447 (0.001)	0.660 (0.000)	N.A
	DT	0.875 (0.002)	0.701 (0.002)	**0.902** (0.002)	0.887 (0.005)	0.778 (0.007)	0.760 (0.005)	0.602 (0.001)	0.732 (0.001)	N.A
平均准确度(排名)	SVM	**0.784** (3.591)	**0.784 (2.500)**	0.773 (3.773)	0.709 (5.591)	0.760 (4.000)	0.784 (2.955)	0.673 (5.591)	/	/
	NB	**0.733 (3.182)**	0.700 (4.273)	0.693 (4.091)	0.709 (4.000)	0.712 (3.727)	0.721 (2.818)	0.626 (5.909)	/	/
	KNN	0.757 (3.818)	0.755 **(2.773)**	0.752 (3.591)	0.668 (5.682)	0.761 (3.364)	**0.765** (3.000)	0.640 (5.773)	/	/
	DT	0.758 (2.364)	0.733 (3.364)	**0.766 (2.000)**	0.739 (3.636)	0.651 (5.727)	0.639 (6.000)	0.668 (4.909)	/	/

(局部)、SCII_CBA、HMM 和 SCII_MA 等方法。

同时，本研究也通过此测试分析了先进的基于特征的方法与 Hamming 编码器在不同分类器上的性能表现。在排除 SGT 和 Gokrimp 后的 11 个数据集上的结果如图3.5所示，其中 $CD \simeq 2.43$。显著性测试结果表明，除了相对于 Seq2Pat 外，Hamming 编码器相比其他现有方法并未展现出统计上的显著优势。值得注意的是，当使用 SVM 和 KNN 作为分类器时，Hamming 编码器明显优于 iBCM；同样，当使用 DT 作为分类器时，本方法也优于 Sqn2Vec 的两个变体实现。

从图3.6可以看出，随着 epoch 的增加，损失函数呈现稳定的下降趋势。对于大多数数据集，当损失下降到大约 40 时，损失趋于稳定。这表明 STE 策略对于这种二值

表 3.3　性能比较：嵌入分类器的方法和 Hamming 编码器的端到端设置

数据集 s	Hamming Encoder (k=5)	SeqDT	SCII_CBA	SCII_MA	SCIS_HAR	SCIS_MA	HMM	CR2N Local	CR2N Global
aslbu	**0.642** (0.009)	0.561 (0.010)	0.604 (0.076)	0.606 (0.087)	0.599 (0.067)	0.575 (0.064)	0.524 (0.053)	0.409 (0.037)	0.383 (0.031)
auslan2	0.285 (0.021)	0.262 (0.020)	0.150 (0.081)	0.170 (0.084)	0.215 (0.100)	0.210 (0.099)	**0.290** (0.064)	0.207 (0.023)	0.197 (0.018)
context	**0.932** (0.007)	0.830 (0.018)	0.712 (0.098)	0.788 (0.076)	0.767 (0.097)	0.829 (0.063)	0.822 (0.050)	0.282 (0.041)	0.219 (0.021)
epitope	**0.889** (0.006)	0.834 (0.010)	0.685 (0.028)	0.717 (0.026)	0.707 (0.030)	0.727 (0.028)	0.790 (0.018)	0.536 (0.034)	0.596 (0.045)
gene	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	0.941 (0.048)	0.851 (0.090)
pioneer	0.982 (0.008)	**0.996** (0.009)	0.969 (0.042)	0.969 (0.042)	0.975 (0.041)	0.975 (0.041)	0.816 (0.068)	0.698 (0.049)	0.699 (0.045)
question	**0.956** (0.004)	0.934 (0.003)	0.860 (0.025)	0.850 (0.023)	0.846 (0.022)	0.858 (0.034)	0.192 (0.024)	0.724 (0.013)	0.777 (0.016)
robot	0.904 (0.004)	0.881 (0.003)	0.790 (0.012)	0.819 (0.013)	0.818 (0.009)	0.822 (0.012)	0.862 (0.010)	**0.983** (0.015)	0.757 (0.059)
skating	**0.344** (0.007)	0.269 (0.016)	0.247 (0.052)	0.230 (0.063)	0.272 (0.054)	0.283 (0.060)	0.273 (0.045)	0.164 (0.007)	0.148 (0.017)
reuters	**0.978** (0.003)	0.894 (0.006)	0.946 (0.023)	0.950 (0.018)	0.960 (0.023)	0.945 (0.018)	0.170 (0.012)	0.644 (0.058)	0.627 (0.056)
unix	**0.923** (0.002)	0.904 (0.002)	0.854 (0.015)	0.850 (0.014)	0.861 (0.016)	0.846 (0.018)	0.789 (0.009)	0.521 (0.081)	0.492 (0.052)
平均准确度（排名）	**0.803** (**1.727**)	0.760 (3.273)	0.711 (5.500)	0.723 (5.045)	0.729 (4.591)	0.734 (4.318)	0.593 (5.273)	0.555 (7.091)	0.522 (8.182)

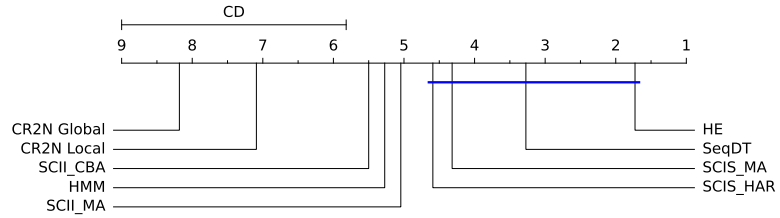


图 3.4　Hamming 编码器与基线方法的比较 (Bonferroni-Dunn 检验)

Fig. 3.4　Comparison of Hamming Encoder (HE) against each competing method based on the Bonferroni-Dunn test

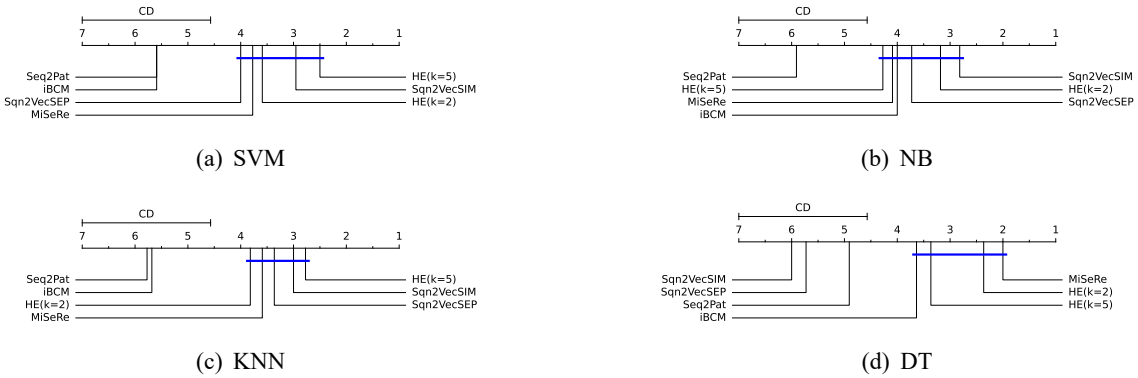


图 3.5　Hamming 编码器与基于特征的方法的比较 (Bonferroni-Dunn 检验)

Fig. 3.5　Comparison of Hamming Encoder (HE) against each feature-based method coupled with four 分类器 s based on the Bonferroni-Dunn test

化方法表现良好，并且在模型训练过程中，提出的可解释量化器稳定收敛。

先前的分类准确性实验清晰地表明，这种二值化策略能够有效提取用于序列分类的区分性 k-mer 集合。更重要的是，这种可解释的量化方法为理解 1DCNN 模型提供了宝贵的见解。在之前的研究中，本方法采用了 SteHeaviside 作为二值化量化器，该方法训练出的 CNN 权重矩阵特点是大多数位置为 0，只有少数关键位置为 1。这种结构本身就具有一定的可解释性，因为这些权重可以被视为多个模式的组合表达。然而，SteHeaviside 可能在单个卷积核中同时提取多个模式，这使得深入分析和理解变

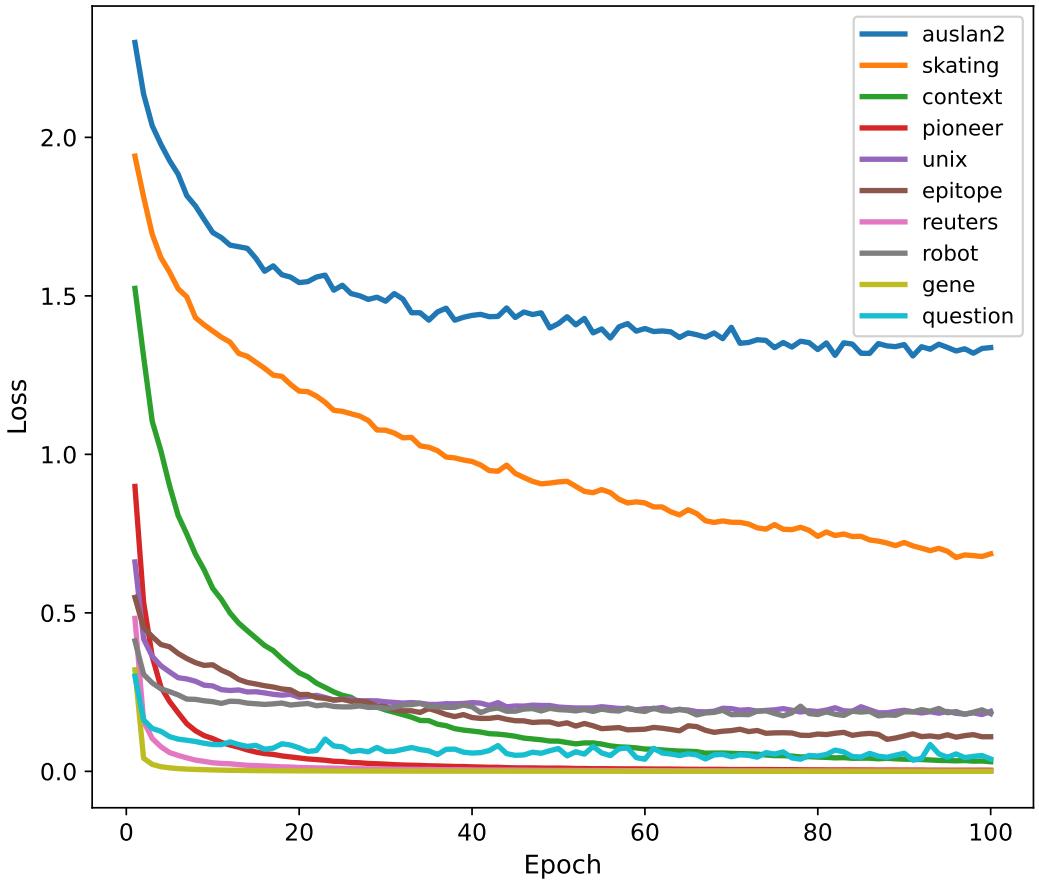


图 3.6　Hamming 编码器的训练损失与 Epoch 的关系
Fig. 3.6　Training loss vs. epochs for the Hamming Encoder

得相当困难。

另一方面，本文提出的 Hamming 编码器二值化量化器在训练过程中增加了约束，使得每个内核只表示一个 k-mer。实验证明，Hamming 编码器收敛良好，并且能够产生区分性的 k-mer 集。

3.3.2　特征数量分析

图3.7展示了不同方法在各基准数据集上所使用的特征数量。可以观察到，SeqDT、CR2N（局部）和 CR2N（全局）在所有数据集上均使用了相对少量的特征。这主要归因于它们将模式挖掘过程与分类器构建过程有机结合。特别值得注意的是，CR2N（局部）对特定局部模式表现出高度敏感性，能够通过极少量的模式（规则）实现显著高的分类准确率。例如，在 “robot” 数据集上，CR2N（局部）仅使用平均 1.6 个模式就达到了 98.3% 的分类准确率。而 CR2N（全局）在大多数数据集上检测到的模式数量极少，在某些情况下（如 “gene” 和 “epitope” 数据集）甚至未能识别出任何规则，这直接限制了其在这些数据集上的表现效果。

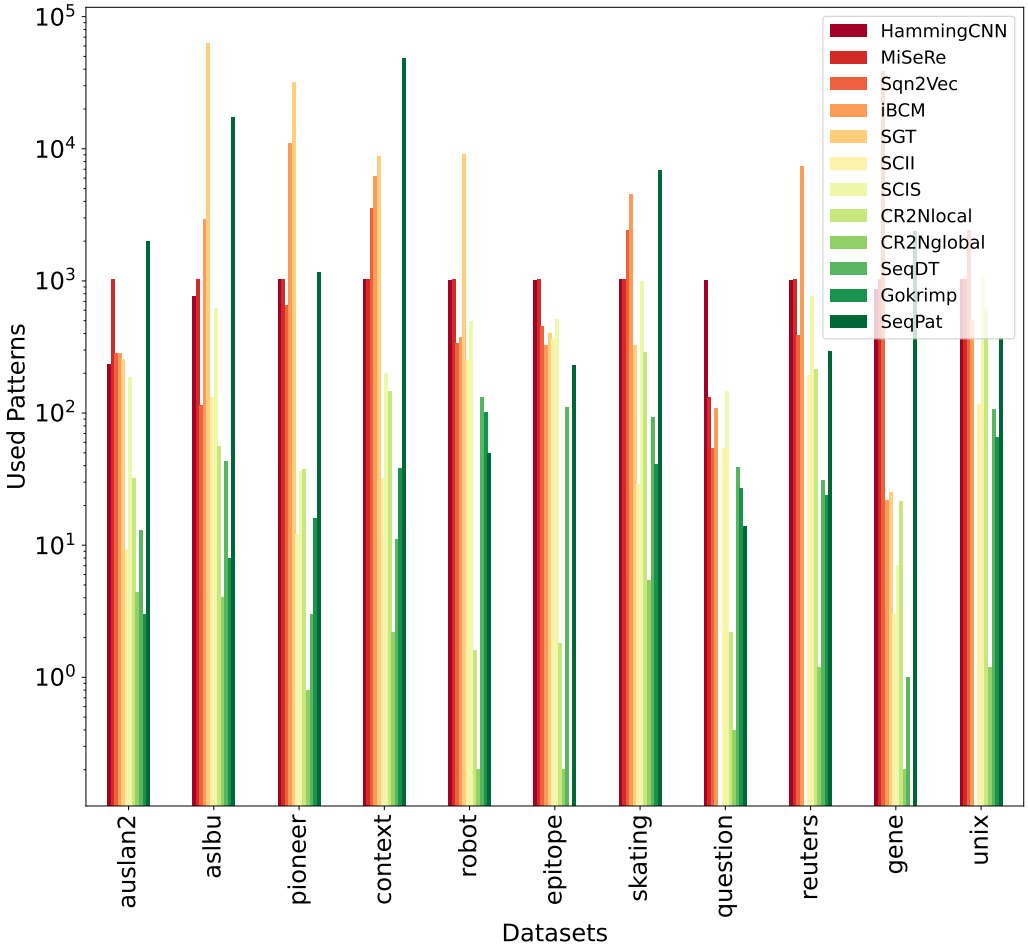


图 3.7　不同方法在特征数量方面的比较

Fig. 3.7　Comparison of different methods in terms of the number of features

图3.8展示了随着模式数量的增加，Hamming 编码器的平均分类准确率的趋势。可以看到，随着模式数量的增加，预测性能通常会提高。然而，在使用的模式数量达到几百个之后，性能的提升就会趋于稳定。图3.9展示了各种方法随着特征数量增加而产生的挖掘时间变化。从图中可以观察到，大部分的方法随着设置模式数量的增长，其挖掘时间并没有呈现出显著的变化。虽然整体趋势是随着模式数量的增加挖掘时间也相应增加，但这种增长相对平缓。在检查序列信息表以及本方法提出的方法的性能时，可以看到该方法通常在具有更均匀序列长度分布的较长序列的数据集上表现良好，例如“question”和“robot”。然而，在具有不均匀序列长度分布的较短序列的数据集上，例如“auslan2”和“unix”，它可能表现不佳。这是因为本方法使用了固定长度的 k-mer，这可能不足以捕获较短序列的足够信息。

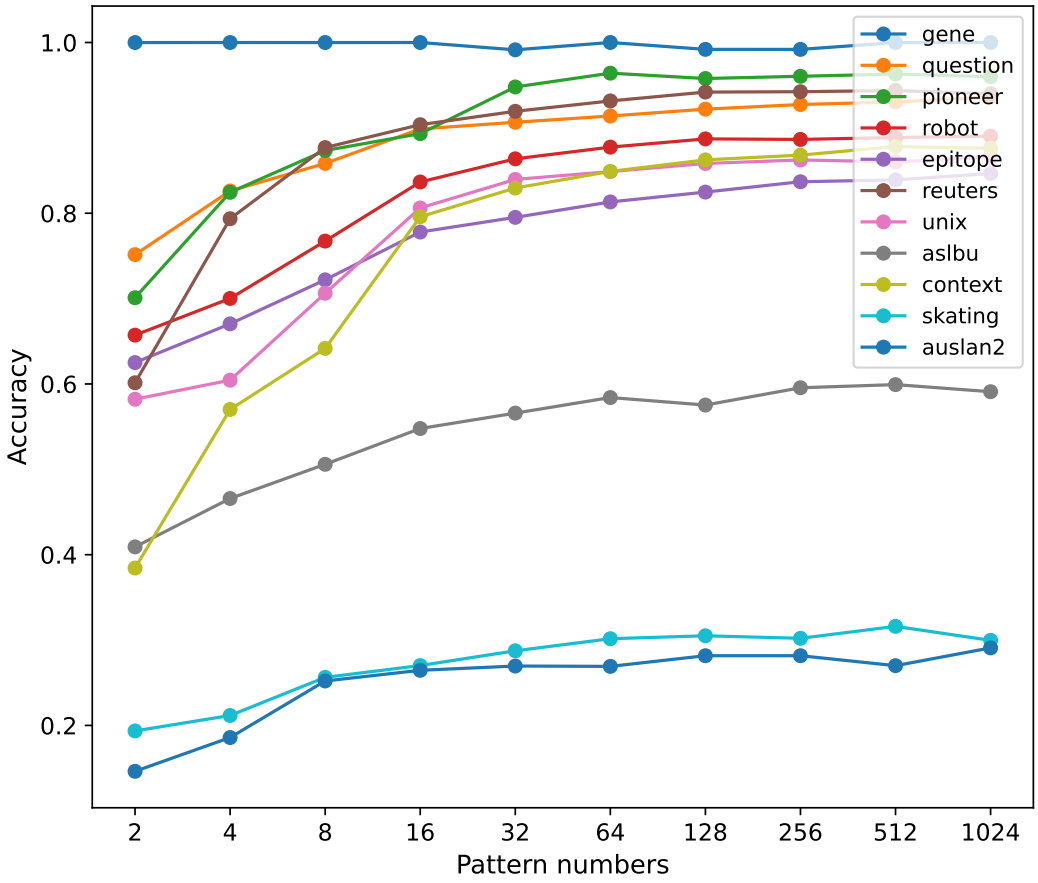


图 3.8　每个数据集的准确率波动与挖掘模式数量的关系

Fig. 3.8　Fluctuation of accuracy per dataset vs. the number of mined patterns

3.3.3　时间效率分析

图3.10提供了 Hamming 编码器和其他序列模式挖掘算法的运行时间比较。图中显示，本方法在大多数数据集上与其他算法相当。然而，在“unix”和“reuters”数据集上表现不佳。这可以归因于本方法利用 1DCNN 结构进行 one-hot 映射和填充矩阵，基于最大序列长度。因此，输入矩阵的高度等于项目集的大小，这意味着对于“reuters”这样的数据集，卷积核的大小更大。此外，对于具有较大最大序列长度的数据集，例如“unix”，需要执行更多的卷积操作。因此，对于具有相对均匀序列长度分布和不太多项目的数据集，Hamming 编码器在挖掘区分性 k-mer 集方面相对较快。

3.3.4　消融实验

为了验证整体优化模式集相比单独搜索区分性模式的优势，我们设计了以下实验。当 $k=2$ 时，多数序列数据集中的 k-mer 数量相对较少，这便于获取所有可能的 k-mer，并将序列转换为特征向量来比较不同方法的效果。在实验中，本文首先从训练集提取全部可能的 k-mer，然后应用不同的特征选择方法进行对比。其中，随机森

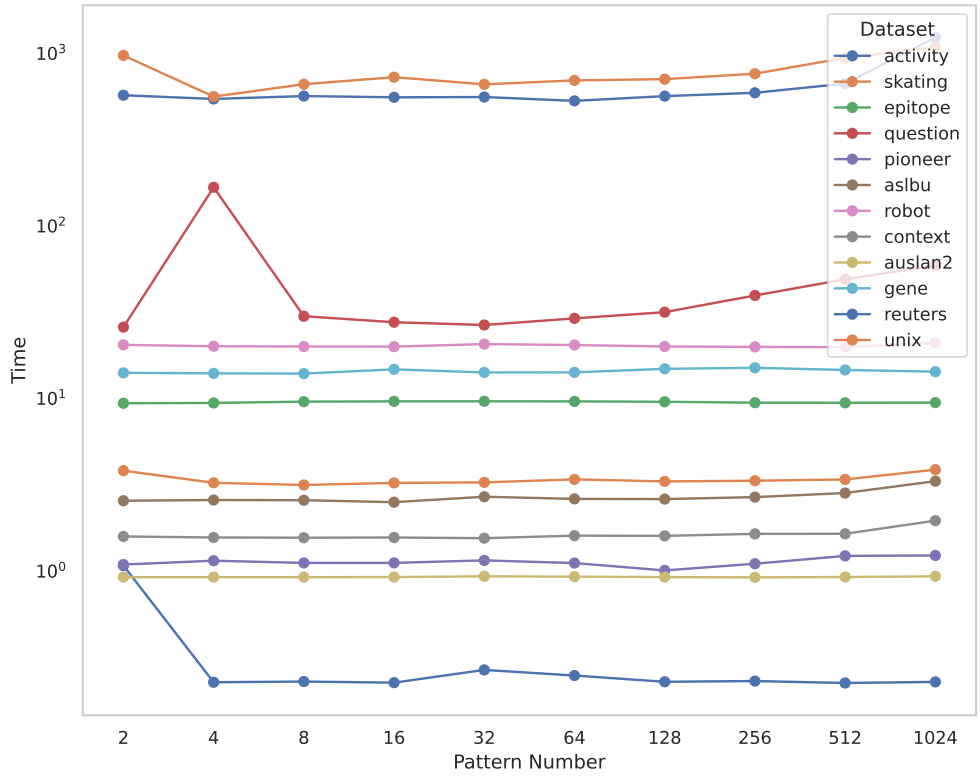


图 3.9　特征数量增加与挖掘时间的关系

Fig. 3.9　Relationship between the number of features and the mining time

林（RF）被用于发现特征组合的能力，而 Chi2 [77] 作为一种过滤方法则用于单独选择具有区分性的特征。详细比较结果见表3.4。

如表3.4所示，Hamming 编码器和 RF 在大多数数据集中优于 Chi2 特征选择方法和没有特征选择的方法。这表明优化整个区分模式集是有益的。此外，与 RF 特征选择相比，本方法稍微有优势。

为了系统比较从头训练二值化神经网络与在训练标准 1D CNN 后应用二值化的效果差异，以及对比随机内核与从现有数据随机选择 k-mer 的性能表现，本研究进行了深入的消融实验。在实验设计中，本文保持标准 CNN 的参数与结构与前述实验完全一致，随后在模型训练完成后的最终阶段，应用我们提出的二值化函数提取卷积核中蕴含的 k-mer 模式，并结合标准分类器进行综合评估。表3.5详细呈现了实验结果，数据清晰表明直接二值化与 Hamming 编码器在性能上显著优于其他两种基于随机 k-mer 选择的方法。值得注意的是，虽然 Hamming 编码器与直接二值化方法在单项指标上差异不显著，但从所有分类器的平均排名来看，Hamming 编码器仍然展现出微弱但一致的优势。

表 3.4　Hamming 编码器与随机森林及 Chi2 特征选择方法的性能比较

Tab. 3.4　Performance comparison of Hamming Encoder, Random Forest, and Chi2 feature selection methods

数据集	分类器	Hamming Encoder	RF Selection	Chi2 Selection	All 2-mer
aslbu	SVM	**0.639** (0.011)	0.631 (0.009)	0.602 (0.008)	0.627 (0.011)
	NB	**0.588** (0.010)	0.553 (0.010)	0.528 (0.008)	0.515 (0.008)
	KNN	**0.593** (0.016)	0.573 (0.012)	0.575 (0.012)	0.548 (0.012)
	DT	0.547 (0.016)	0.548 (0.017)	**0.558** (0.013)	0.533 (0.020)
auslan2	SVM	0.273 (0.016)	**0.277** (0.013)	0.262 (0.024)	0.273 (0.018)
	NB	**0.295** (0.009)	0.282 (0.018)	0.292 (0.020)	0.290 (0.013)
	KNN	0.251 (0.025)	0.275 (0.022)	0.266 (0.037)	**0.278** (0.020)
	DT	0.264 (0.018)	**0.280** (0.018)	0.278 (0.016)	0.276 (0.014)
context	SVM	0.896 (0.011)	**0.901** (0.007)	0.897 (0.008)	**0.901** (0.009)
	NB	**0.843** (0.011)	0.813 (0.012)	0.800 (0.021)	0.800 (0.012)
	KNN	0.852 (0.012)	**0.875** (0.010)	0.837 (0.014)	0.830 (0.009)
	DT	0.826 (0.017)	0.823 (0.021)	0.826 (0.014)	**0.830** (0.007)
epitope	SVM	0.858 (0.004)	0.843 (0.006)	**0.859** (0.006)	0.858 (0.006)
	NB	0.715 (0.004)	**0.717** (0.003)	0.712 (0.002)	0.712 (0.003)
	KNN	0.809 (0.004)	0.800 (0.006)	0.811 (0.006)	**0.813** (0.006)
	DT	0.863 (0.006)	0.868 (0.006)	0.869 (0.006)	**0.871** (0.009)
gene	SVM	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)
	NB	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)
	KNN	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)
	DT	**1.000** (0.000)	1.000 (0.001)	**1.000** (0.000)	1.000 (0.001)
pioneer	SVM	0.964 (0.009)	**0.982** (0.008)	0.949 (0.009)	0.922 (0.009)
	NB	**0.923** (0.007)	0.892 (0.013)	0.882 (0.011)	0.894 (0.010)
	KNN	**0.959** (0.013)	0.951 (0.011)	0.932 (0.004)	0.925 (0.018)
	DT	0.982 (0.016)	0.984 (0.009)	**0.989** (0.008)	0.985 (0.012)
question	SVM	**0.943** (0.003)	0.926 (0.002)	0.909 (0.003)	0.901 (0.003)
	NB	**0.910** (0.004)	0.799 (0.004)	0.760 (0.006)	0.743 (0.004)
	KNN	**0.924** (0.005)	0.885 (0.006)	0.901 (0.005)	0.836 (0.006)
	DT	0.932 (0.004)	**0.943** (0.004)	0.931 (0.003)	0.941 (0.003)
robot	SVM	0.912 (0.002)	**0.923** (0.002)	0.907 (0.002)	0.921 (0.002)
	NB	0.748 (0.003)	**0.787** (0.002)	0.780 (0.001)	0.778 (0.001)
	KNN	0.909 (0.003)	0.920 (0.002)	0.902 (0.002)	**0.926** (0.002)
	DT	0.876 (0.003)	0.881 (0.004)	0.877 (0.003)	**0.883** (0.003)
skating	SVM	**0.293** (0.011)	0.258 (0.013)	0.251 (0.015)	0.245 (0.009)
	NB	**0.282** (0.012)	0.254 (0.008)	0.255 (0.009)	0.243 (0.011)
	KNN	**0.243** (0.008)	0.225 (0.012)	0.204 (0.016)	0.201 (0.013)
	DT	**0.272** (0.017)	0.271 (0.013)	0.280 (0.017)	0.269 (0.025)
reuters	SVM	**0.956** (0.004)	0.940 (0.003)	0.884 (0.004)	**0.956** (0.003)
	NB	**0.923** (0.006)	0.911 (0.003)	0.749 (0.010)	0.835 (0.003)
	KNN	**0.934** (0.005)	0.921 (0.004)	0.897 (0.004)	0.881 (0.005)
	DT	0.906 (0.007)	0.901 (0.009)	0.899 (0.004)	**0.908** (0.009)
unix	SVM	**0.891** (0.002)	0.882 (0.001)	0.854 (0.001)	0.876 (0.001)
	NB	**0.838** (0.004)	0.797 (0.003)	0.713 (0.003)	0.673 (0.007)
	KNN	0.853 (0.007)	**0.865** (0.002)	0.857 (0.007)	0.839 (0.005)
	DT	0.875 (0.002)	0.883 (0.002)	0.876 (0.002)	**0.886** (0.002)
平均准确度（排名）	SVM	**0.784** (**1.909**)	0.778 (2.000)	0.761 (3.227)	0.771 (2.864)
	NB	**0.733** (**1.500**)	0.710 (2.227)	0.679 (2.955)	0.680 (3.318)
	KNN	**0.757** (**2.045**)	0.754 (2.227)	0.744 (2.682)	0.734 (3.045)
	DT	0.758 (3.091)	**0.762** (2.545)	**0.762** (2.273)	**0.762** (**2.091**)

表 3.5　Hamming 编码器与最终步骤中的离散化、随机选择 k-mer 和随机生成 k-mer 的性能比较
Tab. 3.5　Performance comparison of Hamming Encoder with discretization in the final step, random selection of k-mers, and random generation of k-mers

数据集	分类器	Hamming Encoder	Direct Binary	Random k-mer (Select)	Random k-mer (Generate)
aslbu	SVM	**0.580** (0.016)	0.553 (0.053)	0.546 (0.055)	0.361 (0.062)
	NB	0.494 (0.019)	**0.498** (0.044)	0.364 (0.042)	0.246 (0.080)
	KNN	0.509 (0.022)	**0.519** (0.047)	0.455 (0.046)	0.339 (0.073)
	DT	0.478 (0.025)	**0.499** (0.051)	0.462 (0.049)	0.326 (0.069)
auslan2	SVM	0.237 (0.018)	0.203 (0.056)	**0.258** (0.056)	0.190 (0.039)
	NB	0.232 (0.011)	0.210 (0.048)	**0.273** (0.057)	0.175 (0.043)
	KNN	0.201 (0.026)	0.210 (0.034)	**0.235** (0.065)	0.185 (0.079)
	DT	0.221 (0.012)	0.218 (0.034)	**0.245** (0.046)	0.203 (0.039)
context	SVM	**0.949** (0.007)	0.948 (0.025)	0.923 (0.026)	0.446 (0.067)
	NB	**0.834** (0.015)	0.827 (0.060)	0.715 (0.059)	0.373 (0.096)
	KNN	0.927 (0.012)	**0.946** (0.028)	0.881 (0.030)	0.479 (0.062)
	DT	0.846 (0.019)	**0.875** (0.041)	0.833 (0.032)	0.529 (0.091)
epitope	SVM	**0.924** (0.004)	0.917 (0.011)	0.897 (0.017)	0.852 (0.016)
	NB	**0.718** (0.004)	0.713 (0.016)	0.713 (0.009)	0.670 (0.021)
	KNN	**0.877** (0.004)	0.867 (0.015)	0.857 (0.013)	0.779 (0.013)
	DT	**0.869** (0.006)	0.867 (0.015)	0.856 (0.017)	0.835 (0.016)
gene	SVM	**1.000** (0.000)	**1.000** (0.000)	1.000 (0.000)	**1.000** (0.000)
	NB	0.961 (0.064)	**1.000** (0.000)	0.907 (0.186)	0.752 (0.249)
	KNN	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.000)
	DT	**1.000** (0.000)	**1.000** (0.000)	**1.000** (0.001)	**1.000** (0.001)
pioneer	SVM	0.989 (0.007)	**0.997** (0.009)	0.988 (0.021)	0.631 (0.036)
	NB	0.935 (0.009)	**0.972** (0.022)	0.863 (0.076)	0.256 (0.062)
	KNN	0.989 (0.006)	**0.997** (0.009)	0.950 (0.042)	0.600 (0.064)
	DT	0.938 (0.019)	**0.978** (0.020)	0.928 (0.028)	0.566 (0.089)
question	SVM	**0.955** (0.003)	0.927 (0.013)	0.905 (0.010)	0.519 (0.027)
	NB	**0.923** (0.003)	0.915 (0.014)	0.737 (0.010)	0.513 (0.030)
	KNN	0.921 (0.005)	**0.922** (0.011)	0.859 (0.019)	0.511 (0.021)
	DT	**0.940** (0.005)	0.935 (0.009)	0.907 (0.009)	0.531 (0.024)
robot	SVM	**0.952** (0.002)	0.949 (0.006)	0.935 (0.006)	0.655 (0.016)
	NB	0.779 (0.005)	**0.811** (0.016)	0.781 (0.020)	0.624 (0.020)
	KNN	0.935 (0.002)	**0.947** (0.004)	0.942 (0.008)	0.527 (0.016)
	DT	0.892 (0.005)	**0.895** (0.009)	0.892 (0.010)	0.659 (0.016)
skating	SVM	**0.342** (0.018)	0.325 (0.036)	0.259 (0.037)	0.223 (0.026)
	NB	0.275 (0.011)	**0.288** (0.039)	0.213 (0.030)	0.180 (0.027)
	KNN	**0.308** (0.014)	0.276 (0.029)	0.236 (0.027)	0.208 (0.025)
	DT	**0.270** (0.010)	0.249 (0.038)	0.247 (0.028)	0.216 (0.038)
reuters	SVM	**0.975** (0.002)	0.944 (0.015)	0.948 (0.013)	0.510 (0.051)
	NB	**0.927** (0.003)	0.917 (0.013)	0.801 (0.030)	0.501 (0.050)
	KNN	**0.948** (0.004)	0.932 (0.015)	0.915 (0.015)	0.508 (0.052)
	DT	0.908 (0.008)	**0.909** (0.013)	0.871 (0.020)	0.510 (0.054)
unix	SVM	**0.716** (0.002)	0.708 (0.011)	0.675 (0.010)	0.493 (0.009)
	NB	**0.627** (0.015)	0.610 (0.018)	0.350 (0.007)	0.201 (0.031)
	KNN	**0.685** (0.007)	0.677 (0.023)	0.615 (0.020)	0.448 (0.034)
	DT	**0.701** (0.002)	0.699 (0.012)	0.678 (0.013)	0.452 (0.010)
平均准确度（排名）	SVM	**0.784** (**1.682**)	0.770 (1.727)	0.758 (2.705)	0.535 (3.886)
	NB	0.700 (**1.698**)	**0.706** (1.721)	0.610 (2.698)	0.408 (3.884)
	KNN	**0.755** (**1.690**)	0.754 (1.738)	0.722 (2.690)	0.508 (3.881)
	DT	0.733 (**1.683**)	**0.738** (1.756)	0.720 (2.683)	0.530 (3.878)

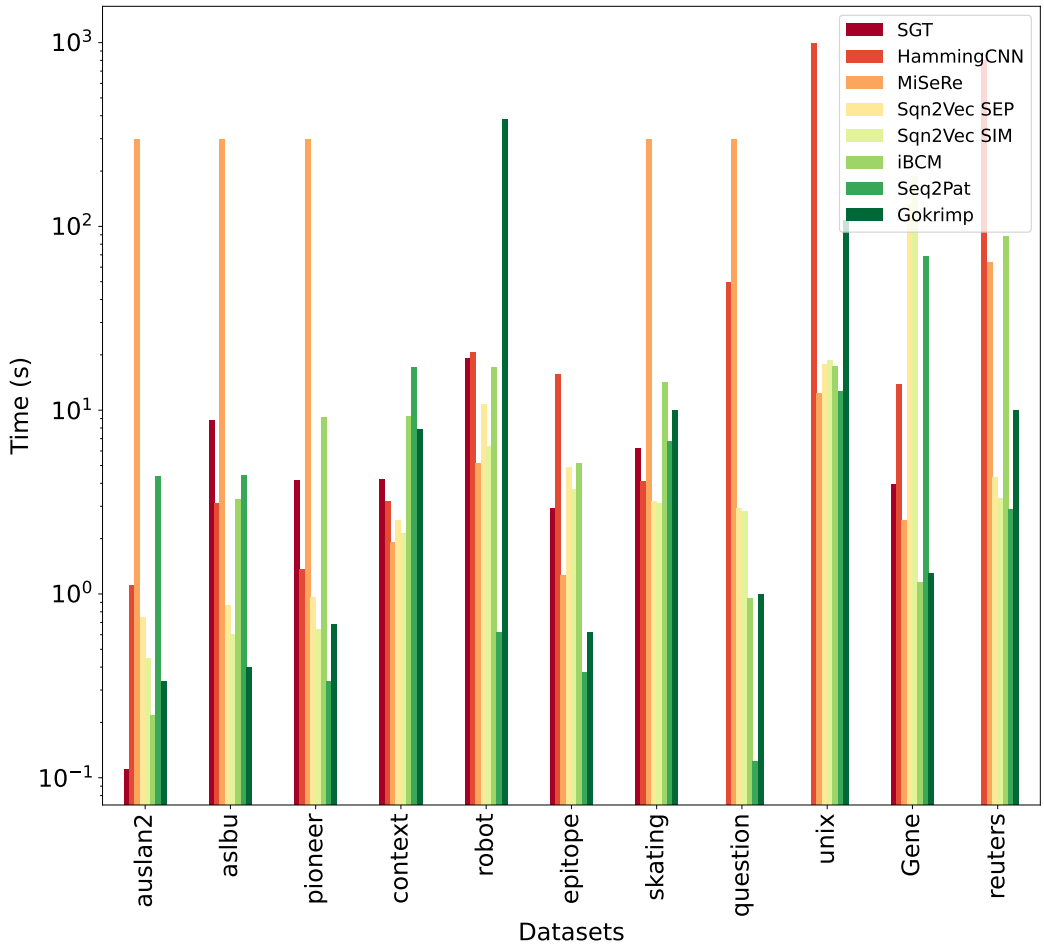


图 3.10　基于特征的方法在模式挖掘时间方面的比较

Fig. 3.10　Comparison of feature-based methods in terms of pattern mining time

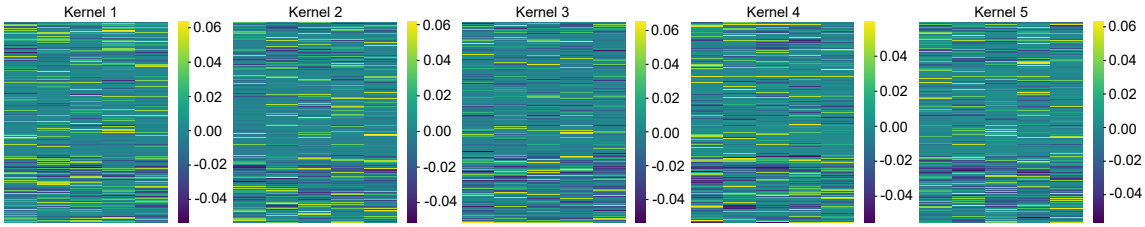
3.3.5　卷积核的可解释性

首先值得强调的是，本文提出的 Hamming 编码器直接提供了具有可解释权重的训练策略。为了展示其可解释性优势，本文选取”aslbu” 数据集进行对比分析，将标准 CNN 与 Hamming 编码器的前 5 个卷积核进行了可视化，详细结果如图3.11所示。从图中可以清晰地观察到，标准 CNN 的每个卷积核包含大量突出显示的（黄色）区域，这表明每个卷积核实际上包含了多种模式的复杂耦合。这种复杂性导致难以解释神经网络在决策过程中依赖的具体信息。相比之下，通过检查 Hamming 编码器在前向计算过程中的权重分布，能够明确理解每个卷积核的功能与作用。Hamming 编码器的设计确保了在构建分类模型时，权重可以直接转换为解释性的字符串，这一特性对某些应用领域尤为重要。例如，在大规模基因数据库中，Hamming 编码器不仅能高效构建致病基因分类器，还能同时识别如密码子等关键基因片段。

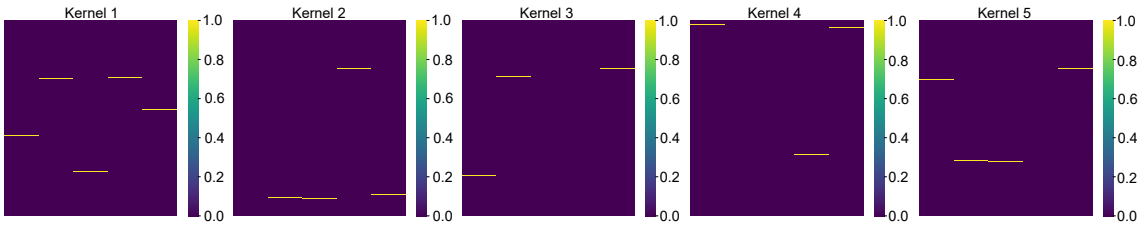
为了研究本方法在与广泛使用的 1DCNN 方法以及普遍的二值化方法：Heaviside $(0,1)$ 和 Sign $(-1,1)$ 相比，在序列分类下的比较优势，本方法在十一个基准数据集上

表 3.6　Hamming 编码器、基线 CNN、Heaviside、Sign 的分类准确性比较
Tab. 3.6　Performance comparison of Hamming Encoder, Baseline CNN, Heaviside, and Sign in terms of classification accuracy

数据集	Hamming Encoder	Baseline CNN	Heaviside	Sign
aslbu	0.642	**0.647**	0.623	0.613
auslan2	0.285	0.268	**0.305**	0.275
context	0.932	**0.954**	0.950	0.942
epitope	0.889	**0.945**	0.939	**0.945**
gene	**1.000**	**1.000**	**1.000**	**1.000**
pioneer	**0.982**	0.913	0.788	0.806
question	0.956	**0.970**	0.942	0.949
robot	0.904	**0.975**	0.970	0.967
skating	0.344	**0.366**	0.323	0.347
reuters	0.978	**0.982**	0.970	0.976
unix	0.923	0.928	0.925	**0.930**
平均准确率	0.803	**0.813**	0.794	0.796



(a) Baseline CNN



(b) Hamming Encoder

图 3.11　标准 CNN 与 Hamming 编码器的前五个卷积核的可视化
Fig. 3.11　Visualization of the first five convolution kernels of standard CNN and Hamming Encoder

进行了大量实验。参数与 Hamming 编码器的参数保持一致。

首先，表3.6展示了四种端到端 CNN 方法的分类准确率。从表中可以看出，本方法的算法实现了 0.803 的平均准确率，这是优于两种量化 CNN 的。量化是一种模型压缩技术，通常会导致模型准确性下降，从而与原始模型相比会有一些损失。与两种常见的量化方法相比，本方法表现出更小的性能损失。此外，对于某些数据集，如“pioneer”，本方法通过限制每个卷积核的学习特征，减少了过拟合，甚至优于基线 CNN。

3.4 本章小结

本章中，本方法提出了一种基于 1DCNN 的方法来挖掘用于序列分类的具有区分性的 k-mer 集合。通过将区分性模式的识别构建为可解释 CNN 编码器的优化问题，本方法有效地利用基于梯度的策略提取了更具区分性的 k-mer 集合，用于分类。本方法在各种基准数据集上的实验结果表明，本方法挖掘的 k-mer 与最先进的方法相比，实现了有竞争力的分类准确性。

此外，本方法采用基于汉明距离的相似性度量将挖掘的 k-mer 转换为特征向量，从而确保与神经网络架构中的全局最大池化操作一致。它确保了在神经网络的前向计算过程中的可解释性。此外，本方法与其他二值化 1DCNN 模型的端到端训练的比较结果表明，本方法可以获得更好的性能，展示了对过拟合的一定适应性。

然而，本方法仍然存在一些局限性。首先，它只能报告连续的子字符串，而在现实场景中，大多数序列信息包含间隔。由于本方法的二值化策略，本方法当前的方法无法报告带有间隔的子字符串。为了解决这个问题，本方法需要提出更好的二值化策略，例如通过统计方法选择某些列。其次，尽管本方法可以替代传统的 1DCNN 神经网络，但由于在二值化过程中丢失了一些信息，它仍然表现出一定的性能下降。因此，如果本方法想要使用这种二值化策略增强 1DCNN 神经网络的可解释性，以满足高准确性的场景，本方法需要更好的梯度估计方法。在未来的研究中，本方法将专注于解决挖掘带有间隔的子字符串的问题，并提出更好的梯度估计方法。

4 可解释序列聚类树

本章提出了一种可解释序列聚类树算法（ISCT），通过构建一个具有简洁结构的决策树来实现序列数据的聚类和解释，利用序列模式的存在与否作为分裂标准，在保证聚类性能的同时提供了直观易懂的模型解释，并设计了基于相对风险的模式选择方法和随机投影的聚类预处理策略，有效平衡了聚类准确性与可解释性。

4.1 问题描述

给定一个包含 m 个不同项目的项集 $\mathcal{I} = \{e_1, e_2, \ldots, e_m\}$，长度为 $|s| = l$ 的离散序列 $s = \langle \sigma_1, \sigma_2, \ldots, \sigma_l \rangle$ 是 $\mathcal{I}$ 上的有序列表，其中每个元素 $\sigma_i \in \mathcal{I}$。序列 s 的子序列 t 被定义为通过从 s 中删除某些元素并保留剩余元素的相对顺序而获得的序列。形式化地表述，若存在严格递增的索引序列 $1 \leq i_1 < i_2 < \cdots < i_k \leq l$，使得 $t = \langle \sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_k} \rangle$，则称 t 为 s 的子序列，记作 $t \subseteq s$。在本研究中，我们将这种子序列作为序列模式 p。

序列数据库 $\mathcal{D}$ 是一个序列列表，其中数据库中的序列数表示为 $|\mathcal{D}|$。给定序列模式 p 在数据库中出现的次数表示为 $Occ(p, \mathcal{D})$。模式 p 在 $\mathcal{D}$ 中的支持度定义为：

$$Supp(p, \mathcal{D}) = \frac{Occ(p, \mathcal{D})}{|\mathcal{D}|}. \tag{4.1}$$

给定阈值 α_{Supp}，如果 $Supp(p, \mathcal{D}) \geq \alpha_{Supp}$，则可以将序列模式 p 视为频繁序列模式。

然而，在实际应用中，直接指定适当的支持度阈值往往面临诸多挑战。因此，作为一种更为实用的替代方案，研究者通常会设置一个参数 k，用于获取支持度最高的前 k 个模式，从而避免阈值选择的困难。

对于给定的数据库 $\mathcal{D} = \{s_1, s_2, \ldots, s_n\}$ 和整数 k，序列聚类方法将数据库 $\mathcal{D}$ 划分为 k 个子集 $C = \{c_1, c_2, \ldots, c_k\}$。目标是构建一个可解释的序列聚类树，其特点是紧凑而简洁的结构，便于人类用户直观和全面地理解。具体来说，聚类树应具有以下特征：

- *浅层*: 决策树的可解释性主要由其大小决定。与[50] 类似，我们的目标是获得一个具有 k 个叶节点的树，每个叶节点对应一个聚类，这意味着深度和非叶节点的总数最多为 $k - 1$。
- *简洁*: 树的分裂标准应该简洁，特别是在顺序数据的情况下，我们的目标是避免在分裂过程中引入距离函数或相似性矩阵。
- *精准*: 可解释的聚类树应产生与不可解释聚类算法相当的准确结果。

4.2 算法设计

在本章中，我们针对序列聚类的可解释性挑战，提出了一种创新的可解释序列聚类树（ISCT）方法。序列数据聚类的可解释性一直是数据挖掘领域的难点，传统方法往往依赖于复杂的距离度量或相似度计算，难以为用户提供直观理解的划分依据。

ISCT 通过将序列数据的聚类问题转化为一个可解释的决策过程，实现了性能与可解释性的平衡。其核心思想是构建一个结构简洁的决策树，每个内部节点代表一个简单明了的判断：” 序列中是否包含特定模式”。这种基于模式存在与否的决策机制，使得聚类结果易于理解和验证，同时保持了聚类的有效性。

与传统的基于相似度矩阵或复杂特征表示的聚类方法不同，ISCT 直接利用序列模式作为划分标准，这些模式通常能够捕获数据中最具代表性的结构特征。该方法特别适用于要求决策过程透明化的应用场景，如医疗诊断中的症状序列分析、用户行为序列的市场细分、生物序列的功能聚类等领域。

图4.1展示了 ISCT 的整体工作流程，算法设计包含两个关键创新：一是高效的序列预处理机制，二是精确的模式选择策略。这两个创新点共同保证了 ISCT 在保持模型简洁性的同时，实现了与复杂模型相当的聚类准确度。下面各小节将详细介绍算法的具体组成部分。

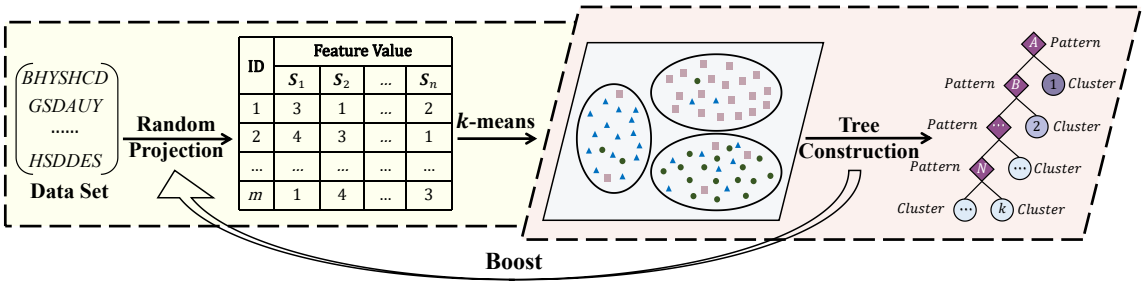


图 4.1　可解释序列聚类树的工作流程

Fig. 4.1　Workflow of the Interpretable Sequence Clustering Tree

4.2.1 算法概述

构建一个既保持聚类效果又具有可解释性的序列聚类树是一项技术挑战。一种直观思路是将已有聚类算法的结果作为伪标签，然后应用决策树等分类方法。然而，这种直接应用的方式通常产生复杂且难以解释的树结构，即使采用剪枝策略或控制分裂过程，也难以得到简洁直观的决策树。

ISCT 算法克服了这一挑战，通过以下技术路线实现了聚类树的构建：

（1）序列随机投影

首先，通过随机生成的序列模式将原始序列数据映射到向量空间。具体而言，我们计算每个随机模式与数据库中序列的相似度（基于最长公共子序列），形成特征矩

阵。随后应用 PCA 降维并执行 K-Means 聚类，获取初始聚类标签。

（2）区分性模式挖掘

针对每个初始聚类，挖掘其特征性的序列模式。我们采用相对风险指标评估模式的区分能力，选择最具辨别力的模式作为树节点的分裂标准。

（3）递归树构建

基于选定的区分性模式，递归构建二叉决策树。每个内部节点表示一个“是否包含特定模式”的判断，左子树对应不包含该模式的序列集合，右子树对应包含该模式的序列集合。

（4）自适应聚类优化

为应对多聚类情况下的复杂性，设计了基于 Boost 策略的自适应聚类优化机制，在树构建过程中动态调整聚类标签，提高最终树结构的区分性能。

该算法的数学表示如下：给定序列数据库 $\mathcal{D}$ 和目标聚类数 k，ISCT 生成一个具有 k 个叶节点和最多 $k-1$ 个内部节点的决策树，每个内部节点代表一个序列模式 p，决策准则为序列 s 是否包含模式 p（即 $p \subseteq s$）。

在下文中，我们将详细阐述基于随机投影的预聚类策略（第4.2.2节）和基于模式选择的聚类树构建方法（第4.2.3节）。

4.2.2 基于随机投影的预聚类

在本章中，我们将详细介绍如何通过随机生成的序列模式将序列转换为矢量表示。也就是说，通过计算每个随机模式与目标序列之间的距离，将数据库中的每个序列投影到相应的子空间。

$$LCS[i][j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0, \\ LCS[i-1][j-1] + 1 & \text{if } P[i] = S[j], \\ \max(LCS[i-1][j], LCS[i][j-1]) & \text{if } P[i] \neq S[j] \end{cases} \tag{4.2}$$

Pattern	Sequence	Match	Value
$< bd >$	$< abddc >$	<*bd**>	2
$< cec >$	$< acaeadcd >$	$<* c * e ** c *>$	3
$< abbaa >$	$< acbbccac >$	$< a * bb ** a *>$	4

图 4.2　计算最长公共子序列长度的示例

Fig. 4.2　Example of calculating the length of the Longest Common Subsequence

本文采用最长公共子序列（LCS）来计算两个序列之间的相似性（随机序列模式

和数据库中的每个序列)。图4.2提供了计算模式和序列之间 LCS 长度的几个示例。为了减轻冗余模式的影响并增强稳定性，本文进一步采用主成分分析（PCA）进行降维。这将特征向量的维数降低到一个较低维度空间。随后，本文应用 K-Means 算法来对转换后的矢量数据进行聚类。

整体流程如算法4.1所示。主要步骤如下：首先，从序列数据库 $\mathcal{D}$ 中随机生成一些子序列（行 3 到 6)。然后，本文使用 LCS 相似性将序列转换为特征向量（行 8)。随后，使用 PCA 进行降维（行 9)。最后，使用 K-Means 对特征向量进行聚类以获得聚类结果（行 10)。

Algorithm 4.1 随机投影聚类

输入: 序列数据库 $\mathcal{D} = \{s_1, s_2, \ldots, s_n\}$, 聚类数 k, 模式数量 $numP$, 最大随机模式长度 $maxS$.
输出: 聚类 $C = \{c_1, c_2, \ldots c_k\}$

1: $\mathcal{I} \leftarrow readItem(\mathcal{D})$
2: $Pset \leftarrow \emptyset$
3: **for** $i \in [1, numP]$ **do**
4: $\quad l = random(1, maxS)$
5: $\quad p = randomSubsequence(l, \mathcal{I})$
6: $\quad Pset.add(p)$
7: **end for**
8: $feature \leftarrow lcsTransform(\mathcal{D}, Pset)$
9: $feature \leftarrow PCA(feature, k)$
10: $C \leftarrow kmeans(feature, k)$
11: **return** C

通过动态规划计算序列 s_n 和随机序列模式 p 之间的距离的时间复杂度为 $O(|s_n| \cdot |p|)$。因此，将所有 $|\mathcal{D}|$ 序列投影到 $numP$ 个随机模式的计算开销为 $O(|\mathcal{D}| \cdot numP \cdot maxl \cdot maxS)$，其中 $maxl$ 表示数据库中序列的最大长度。此外，执行后续 PCA 算法跨 $numP$ 个特征的时间复杂度为 $O(|\mathcal{D}|^2 \cdot numP + numP^3)$ [78]。在最终的 K-Means 算法中，复杂度为 $O(|\mathcal{D}| \cdot t \cdot k \cdot numP)$，其中 k 是指定的聚类数，t 是其收敛后的迭代次数。

4.2.3 聚类树构建

我们可以利用特定模式的存在作为构建树的分裂标准。具体来说，给定初始聚类 $C = \{c_1, c_2, \ldots, c_k\}$，我们需要为 $k-1$ 个聚类中的每一个找到代表性模式以构建树。在构建聚类树时，首先根据序列模式 p 的存在将数据库 $\mathcal{D}$ 划分为两个集合 $\mathcal{D}_p$ 和 $\overline{\mathcal{D}_p}$。然后将问题转化为找到用于树分裂的 Top-1 区分性模式 p。

然而，当处理具有大量聚类的数据集时，直接使用传统的分裂标准，如基尼不纯度，可能表现不佳。在本文中，我们提出了一种基于相对风险的新分裂度量来识别区分性模式。

（1）分裂标准

相对风险（RR）是用于评估序列模式区分能力的常用度量。模式 p 的相对风险计

Algorithm 4.2 ISCT 构建

输入: 序列数据库 $\mathcal{D} = \{s_1, s_2, \ldots, s_n\}$, 聚类数 k, 提升指示器 $boost$, 最大模式数量 $maxN$, 最小序列数 $minS$.
输出: 模式树根节点 N

$C = \{c_1, c_2, \ldots, c_k\} \leftarrow Algorithm\ 4.1(\mathcal{D}, k)$
$N \leftarrow creatNode(\mathcal{D})$
$Built\_Tree(N, \mathcal{D}, C)$
return N

function BUILT_TREE($N, \mathcal{D}, C$)
    if $k < 2$ **or** $|\mathcal{D}| \leq min(K, minS)$ **then**
        return
    end if
    if $boost = True$ **then**
        $C \leftarrow Algorithm\ 4.1(\mathcal{D}, k)$
    end if
    $candidatePatterns \leftarrow \emptyset$
    for each cluster $c_i \in C$ **do**
        $freqPattern \leftarrow mineFreqPattern(c_i, maxN)$
        $candidatePatterns.add(freqPattern)$
    end for
    $p \leftarrow getTop1Pattern(candidatePatterns)$
    if $p = \emptyset$ **then**
        return
    end if
    $\mathcal{D}_p \leftarrow \{s \in \mathcal{D} | p \subseteq s\}$
    $\overline{\mathcal{D}_p} \leftarrow \{s \in \mathcal{D} | p \not\subseteq s\}$
    $k \leftarrow k - 1$
    $N.left \leftarrow createNode(\overline{\mathcal{D}_p})$
    $N.right \leftarrow createNode(\mathcal{D}_p)$
    $Built\_Tree(N.left, \overline{\mathcal{D}_p}, C)$
    $Built\_Tree(N.right, \mathcal{D}_p, C)$
end function

算为正类 $\mathcal{D}^+$ 中的支持度与负类 $\mathcal{D}^-$ 中的支持度之比:

$$RR = \frac{Supp(p, \mathcal{D}^+)}{Supp(p, \mathcal{D}^-)}, \tag{4.3}$$

其中，正类 $\mathcal{D}^+$ 由目标类中的序列组成，$\mathcal{D}^-$ 由其他类中的所有剩余序列组成。除了相对风险，还使用以下内部相似度度量来评估模式 p 的好坏:

$$SIM = \frac{|p| \times |\mathcal{D}^+|}{\sum_{j=1}^{|\mathcal{D}^+|} |s_i|}, \tag{4.4}$$

其中，$|p|$ 是模式 p 的长度，$|\mathcal{D}^+|$ 是正类 $\mathcal{D}^+$ 中的序列数。也就是说，如果具有相同区分能力，我们会倾向于选择更长的序列模式。

然而，相对风险的概念只能作为一种度量来应用，当涉及两个类时。对于输出 k 个分区，需要扩展相对风险的概念以适应多个类。为了满足这一要求，定义 $\mathcal{D}^+$ 为:

$$\mathcal{D}^+ = \{c_i \mid \arg\max_i Supp\,(p, c_i)\}. \tag{4.5}$$

本文选择具有模式 p 最高支持度的聚类作为正类 $\mathcal{D}^+$，并使用一对多策略获得负类 $\mathcal{D}^-$，其中 $\mathcal{D}^- = \mathcal{D} \setminus \mathcal{D}^+$。本文尝试根据公式4.3识别具有最高区分能力的模式。如果多个模式具有相同的相对风险，我们根据公式4.4中的最大 SIM 值选择其中的模式。并且使用该模式的存在作为划分数据集的基础。

（2）聚类树构建

在树构建过程中，使用 Top-1 区分性序列模式进行分裂，将包含该模式的序列分配到同一聚类中。基于这一思想，我们提出了一种用于构建可解释序列聚类树的算法。

构建过程如算法4.2所示。第一步是根据算法4.1获得 k 个初始聚类。在初始化根节点 N 后，我们调用 $Built\_Tree$ 来构建模式树。

树构建遵循自顶向下的二叉分裂方法。在第一阶段，算法检查给定节点的停止标准（行 2 到 4）。第一个标准是 $k < 2$，用于确定是否已达到所需的叶节点数。第二个标准是 $|\mathcal{D}| \leq \min(k, minS)$，用于确定是否继续基于包含的最小序列数进行分裂。

在检查停止标准后，应用频繁序列模式挖掘算法来提取每个 $c_i \in C$ 的频繁序列模式（行 9 到 12）。使用不同的度量选择 Top-1 区分性模式 p（行 13）。如果 $\mathcal{D}$ 中的序列 s 包含模式 p，则将其分配给右子节点。另一方面，如果序列不包含模式 p，则将其分配给左子节点，并通过递归调用 $Built\_Tree$ 函数继续构建树（行 17 到 23）。一旦所有树节点满足停止标准，过程就完成了。

（3）提升构建

实验中观察到，当处理具有更多聚类的数据集时，这种直接构建方法通常会导致报告的聚类 $\hat{C}$ 与给定分区 C 之间的不一致。这种不一致性源于使用分裂模式构建的聚类 $\hat{c}_i$ 与算法 1 给出的预期聚类 c_i 之间的差异。

为了解决这个问题，本文在树构建过程中提出了一种提升策略。换句话说，根据当前剩余的聚类数 k，使用算法4.1重新生成每个 C。图4.3说明了这个问题和提升结果。

由于随机投影聚类的局限性，得到的序列聚类可能并不总是准确的。例如，在图4.3中，簇 2 并不纯净，因为它包含来自簇 1 和簇 3 的序列。然而，当 $\mathcal{D}_p$ 从数据库中被移除时，一些仅出现在簇 1 序列中的项目也被丢弃。为了利用这些信息，我们提

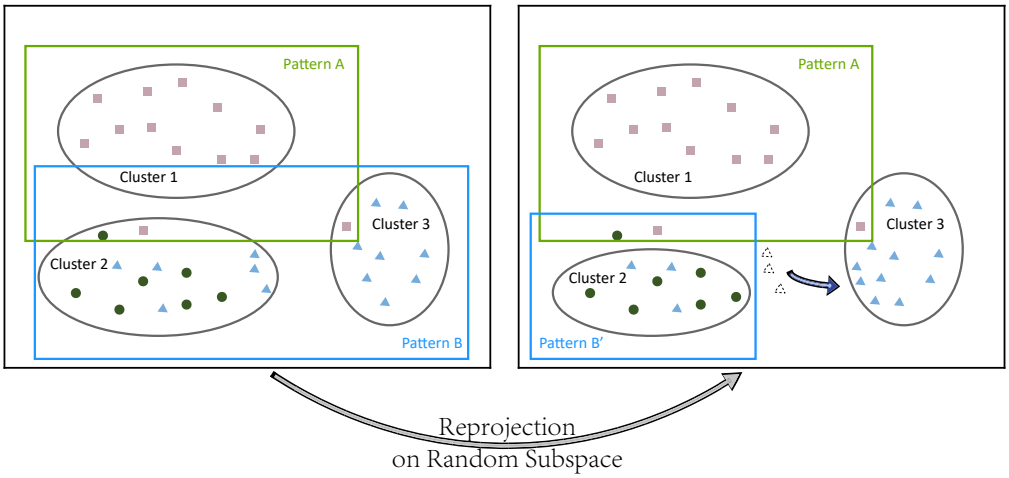


图 4.3　报告的聚类与给定分区之间的不一致性及 Boost 策略

Fig. 4.3　Inconsistency between reported clusters and given partitions, and the boosting strategy

出了一种提升策略，通过将序列随机重新投影到不同的子空间来重新划分数据库 $\mathcal{D}$。这种方法可以部分校准我们的结果，并提高其与基本序列聚类方法的一致性。

示例 *1.* 考虑表4.1中给出的示例序列数据库，其中包含六个序列（$|\mathcal{D}| = 6$）和一个由 $\mathcal{I} = \{a, b, c, d, e\}$ 表示的项目集。数据库被划分为三个聚类。图4.4说明了示例数据的模式树的构建。

表 4.1　示例序列数据库和聚类结果

Tab. 4.1　Toy sequence database and clustering results

ID	序列	聚类	ID	序列	聚类
s_1	$\langle abddc \rangle$	1	s_2	$\langle adbbded \rangle$	1
s_3	$\langle acdeeaa \rangle$	2	s_4	$\langle acaeadcd \rangle$	2
s_5	$\langle abbcaac \rangle$	3	s_6	$\langle acbbccaa \rangle$	3

第一个选择的 Top-1 区分性模式是 $p_1 = \langle bd \rangle$，它将数据分为两个聚类。得到第一个聚类 $\hat{c}_1$，其中包含序列 s_1 和 s_2。然后将剩余序列分配给左节点。由于停止标准表明左节点应进一步划分，选择了 Top-1 区分性模式 $p_2 = \langle ab \rangle$。最终获得了一个高度简洁且可解释的聚类树，其中有三个叶节点，如图中的矩形所示。

（4）时间复杂度

与其他基于特征的聚类算法类似，ISCT 需要提取频繁序列模式。因此，ISCT 的时间复杂度至少等同于用于发现频繁序列模式的算法。如[79] 所述，从事务数据中挖掘最大频繁模式是一个 NP 难题。需要注意的是，序列模式的搜索空间更加广阔。这种复杂性延伸到从序列数据库中发现频繁序列模式，使得这个问题同样是 NP 难的。然而，选择更大的最小支持度阈值可以大大剪枝搜索空间，尽管其分析形式难以获得，

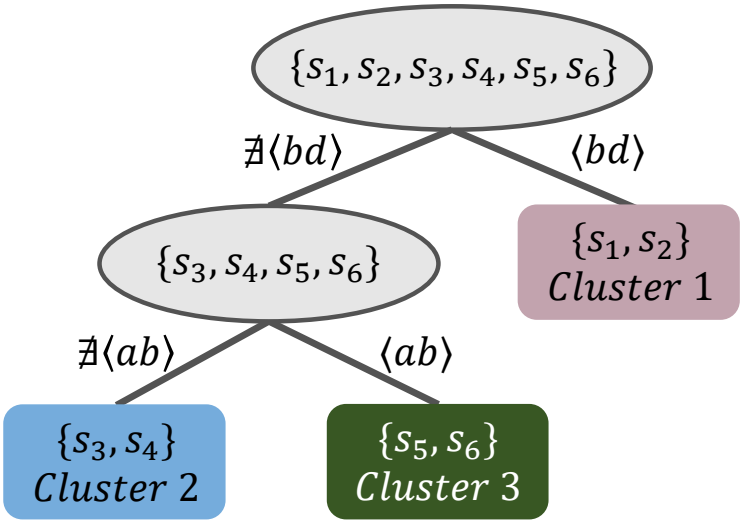


图 4.4　模式树的结构
Fig. 4.4　Structure of the pattern tree

但实际的计算复杂度是可以接受的。因此，本文中使用符号 δ 来表示实际中与挖掘频繁序列模式相关的操作数。

树构建过程的分析如下：首先需要在随机投影的结果下对每个簇挖掘频繁模式，这里的时间复杂度为 $O(k \cdot \delta)$。对每个得到的模式来说，计算判别度需要 $O(n)$。对树进行分裂的操作则需要检验节点序列是否包含该判别模式，时间复杂度为 $O(n \cdot maxl \cdot maxS)$。总的时间复杂度为 $O(k \cdot \delta + k \cdot \beta + k \cdot n \cdot maxl \cdot maxS)$，其中 β 为在第4.1节中介绍的重新投影的时间复杂度，如果不进行 boost 操作则为 1。

4.3　实验

本章详细介绍了 ISCT 的实验设计，包括数据集、实验细节、消融实验、聚类性能和可解释性对比。我们在 14 个真实数据集上进行了大量实验，包括 Activity [80]、News [73] 和 Webkb [73] 等 3 个新增数据集，以及在上一章节中使用的 11 个基准数据集，以评估 ISCT 的性能。

4.3.1　实验细节

表4.2总结了这些新增数据集的基本特征，包括样本数、项集大小、序列长度范围及类别数。

（1）评估指标

本研究中使用的评估指标包括纯度（Purity）、NMI（归一化互信息）和 F1 分数。

表 4.2　新增的三个序列数据集统计特征
Tab. 4.2　Statistical characteristics of three additional sequence datasets

数据集	$\|\mathcal{D}\|$	$\|\mathcal{I}\|$	min$l$	max$l$	类别数
Activity	35	10	12	43	2
News	4976	27884	1	6779	5
Webkb	3667	7736	1	20628	3

纯度的定义如下：

$$Purity(C,T)=\frac{1}{n}\sum_{i=1}^{k}\max_{j=1}^{m}|c_i\cap t_j|, \tag{4.6}$$

其中 n 是数据集的数量，T 和 C 分别是实际标签集和输出聚类集。

NMI 的定义如下：

$$NMI=\frac{2\times MI(C,T)}{H(C)+H(T)}, \tag{4.7}$$

其中

$$MI=H(C)+H(T)-H(C,T), \tag{4.8}$$

是聚类结果 (C) 和实际标签 (T) 之间的互信息，$H(C)$ 和 $H(T)$ 分别是聚类结果和实际标签分布的熵。

F1 分数的定义如下：

$$F1=2\times\frac{Precision\times Recall}{Precision+Recall}, \tag{4.9}$$

其中

$$Precision=\frac{TP}{TP+FP}, \tag{4.10}$$

$$Recall=\frac{TP}{TP+FN}, \tag{4.11}$$

TP、TN、FP、FN 分别是真阳性、真阴性、假阳性和假阴性。

（2）对比方法

本章将使用以下基线方法进行性能比较：

- 层次聚类方法：HC [32] 采用标准编辑距离来获得成对的不相似性，平均链接用于计算聚类之间的距离。对于 MCSC [21]，我们使用 B_MCSC 变体。

- 基于特征的方法：对于 FB-LL [81]，最小支持度固定为 0.05，模式的长度限制在 3 到 5 之间。对于 SGT [28]，我们设置 $kappa=1$ 和 $lengthSensitive=False$。

表 4.3　在不同树构建策略下，平均减少率的性能比较。

方法	**ICST(Boost)**	ICST(Direct)
Purity	**-0.099**	-0.158
NMI	**-0.202**	-0.264
F1	**0.040**	0.039

- 分区方法：kkmeans [82] 使用 3-谱字符串核。对于 k-Median [83]，编辑距离被用作成对的不相似性度量。对于 MinDL [84]，惩罚参数 α 和 γ 分别指定为 0.1 和 0。

- 基于树的方法：NTC [85] 首先使用默认窗口长度将序列划分为子序列，以获得分段矩阵表示，然后将其用作单个决策树或随机森林的输入。然后，计算每个分段子序列在决策树叶节点中的重复次数（nTree=10），以创建序列的数值表示。对于 RFSC [86]，该方法首先生成一组假序列作为负样本，然后应用监督随机森林算法生成多个聚类结果。然后，基于二分图划分的聚类集成过程用于生成最终的一致聚类。在这里，树的数量 (q) 和随机模式的数量 ($numP$) 分别设置为 30 和 10。

（3）实验平台及算法设置

在实验中，最终聚类数 k 被设置为真实聚类数，除了 MinDL。实验在一台配备 AMD Ryzen 9 5900X CPU 和 32GB 内存的 PC 上进行，结果是通过十次试验的平均值获得的。ISCT 的最大模式数设置为 2048，如果 min$l \leq 10$，则默认最大模式长度为 5，最大模式长度为 min$l$。前 k 个频繁模式的数量被设置为 512。

4.3.2　消融实验

在本节中，本文通过消融实验验证了 ISCT 的有效性。首先，通过随机投影获得的聚类结果与直接构建树的结果不同。本文提出的直接构建树方法适用于初始聚类，同时限制了树的复杂性。在树的构建过程中，本文引入了一种 Boost 方法来辅助构建。为了验证策略的有效性，本文在包含两个以上类别的数据集上进行了实验。

表4.3展示了不同树构建方法和随机投影聚类算法之间的平均性能减少率的比较。结果表明，Boost 策略的性能减少率优于直接构建方法。

为了展示构建策略的优势，本文使用 t-SNE 对“Reuters”数据集进行可视化。可视化结果如图4.5所示。从图中可以看出，Reuters 数据集在随机投影子空间中的四个簇的可区分性并不是很明显。因此，这导致 K-Means 算法错误地将蓝色和绿色簇合并为一个簇。在此基础上构建树将导致识别不合适的模式。然而，通过利用 Boost 策略，我们成功地提取出能够准确区分四个不同类别的模式。

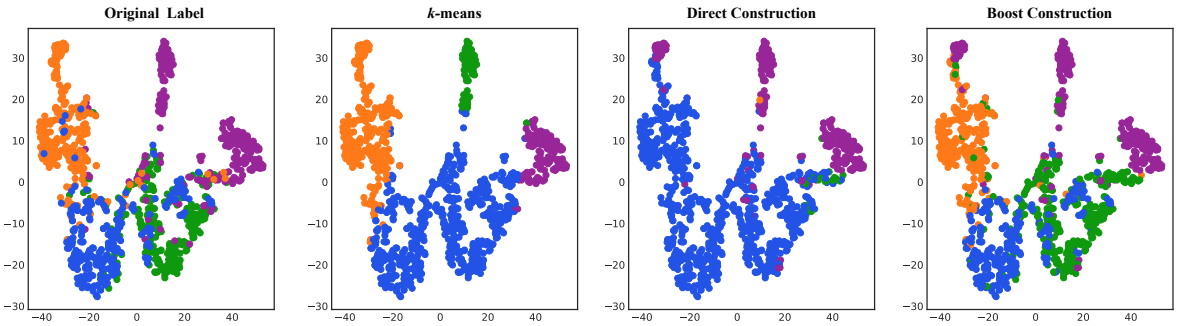
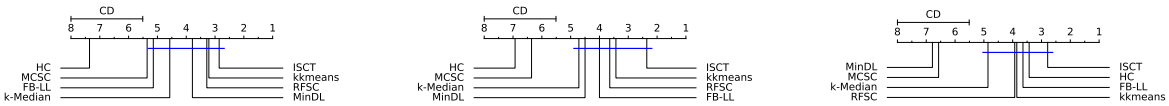


图 4.5　具有原始标签、K-Means、直接构建和 Boost 构建的特征空间的 t-SNE 可视化
Fig. 4.5　T-SNE visualization of feature space with original labels, K-Means, direct construction, and Boost construction

4.3.3　聚类性能

（1）性能

我们的方法为序列聚类过程提供了高度可解释的结构。然而，与现有算法进行比较也是很重要的。表4.4给出了 ISCT、HC、FB-LL、MCSC、k-Median、MinDL、kkmeans、RFSC、SGT 和 NTC 的性能比较结果，包括 Purity、NMI 和 F1-score。由于实验中存在内存限制，部分 SGT 的结果无法获得。至于 NTC，由于在某些数据集上报告了操作错误，因此部分结果缺失。图4.7显示了 14 个数据集上不同方法的运行时间。



(a) Purity　(b) NMI　(c) F1-score

图 4.6　在显著性水平为 0.05 时的 Bonferroni-Dunn 临界差异图
Fig. 4.6　Bonferroni-Dunn critical difference diagrams at a significance level of 0.05

本章提出的 ISCT 方法在所有评估指标上均取得了最佳的平均性能，表明在识别性能方面优于其他方法。为了进一步评估 ISCT 与其他方法的性能，进行了 Bonferroni-Dunn 检验，以验证 ISCT 是否与其他方法表现相似的零假设。显著性检验结果 ($CD = 2.49$) 如图4.6所示。如果我们的方法与现有算法之间没有联系，则表明我们的方法显著更好。从结果中可以看出，ISCT 并不显著优于所有其他方法，但在性能方面优于 HC。值得注意的是，MinDL 在某些数据集上获得了最高的纯度。然而，它通常会生成大量的簇，每个簇中只有很少的序列。例如，在“Pioneer”数据集上，MinDL 生成了 23 个簇，而实际簇的数量只有 3 个。至于 SGT，在许多数据集上表现良好，但在包含大型项集的数据集上由于其高存储空间需求而表现不佳。与两种层次聚类算法 (HC 和 MCSC) 相比，我们的方法在大多数数据集上始终优于它们。

表 4.4　ISCT 与对比方法性能比较

Tab. 4.4　Performance comparison of ISCT with baseline methods

Datasets	Evaluation	**ISCT**	HC	FB-LL	MCSC	k-Median	MinDL	kkmeans	RFSC	SGT	NTC
Activity	Purity	0.994	0.600	0.643	0.686	0.695	0.600	0.720	0.700	**1**	0.886
	NMI	0.973	0.038	0.129	0.118	0.231	0.078	0.225	0.248	**1**	0.565
	F1-score	0.989	0.652	0.613	0.537	0.620	0.515	0.612	0.664	**1**	0.808
Aslbu	Purity	0.491	0.380	0.501	0.373	0.441	**0.507**	0.469	**0.507**	0.432	N.A
	NMI	0.218	0.028	**0.220**	0.046	0.139	0.177	0.157	0.230	0.164	N.A
	F1-score	0.317	**0.366**	0.286	0.183	0.268	0.126	0.257	0.360	0.306	N.A
Auslan2	Purity	0.348	0.195	0.309	0.275	0.316	0.295	0.321	0.347	**0.356**	N.A
	NMI	0.331	0.159	0.336	0.221	0.315	0.245	0.316	0.312	**0.355**	N.A
	F1-score	**0.265**	0.173	0.246	0.153	0.251	0.193	0.256	0.209	**0.265**	N.A
Context	Purity	0.569	0.425	0.518	0.354	0.572	0.55	0.610	0.377	**0.792**	0.600
	NMI	0.557	0.466	0.407	0.071	0.515	0.300	0.593	0.201	**0.655**	0.538
	F1-score	0.551	0.487	0.444	0.235	0.523	0.211	0.573	0.352	**0.699**	0.535
Epitope	Purity	0.627	0.559	0.559	**0.683**	0.594	0.670	0.656	0.674	0.559	0.559
	NMI	0.114	0.044	0.096	0.093	0.054	0.060	0.126	0.105	**0.142**	**0.142**
	F1-score	0.587	**0.671**	0.635	0.585	0.550	0.277	0.582	0.568	0.620	0.620
Gene	Purity	**1**	0.511	**1**	0.519	0.935	0.965	0.999	0.977	**1**	1
	NMI	**1**	0.060	0.994	0.012	0.782	0.158	0.989	0.875	**1**	1
	F1-score	**1**	0.665	0.999	0.500	0.914	0.059	0.998	0.956	**1**	1
News	Purity	0.254	0.210	0.269	0.241	0.284	**0.535**	0.462	0.266	N.A	N.A
	NMI	0.020	0.002	0.106	0.007	0.036	**0.249**	0.226	0.023	N.A	N.A
	F1-score	0.262	0.253	0.329	0.218	0.263	0.256	**0.344**	0.282	N.A	N.A
Pioneer	Purity	0.798	0.656	0.652	0.644	0.662	0.713	0.807	0.790	0.806	**0.881**
	NMI	0.549	0.064	0.175	0.090	0.097	0.181	0.459	0.465	0.474	**0.616**
	F1-score	0.688	0.657	0.476	0.406	0.515	0.338	0.652	0.686	0.684	**0.727**
Question	Purity	0.656	0.518	0.518	**0.745**	0.604	0.570	0.560	0.656	N.A	0.637
	NMI	0.122	0.022	0.019	**0.295**	0.081	0.047	0.015	0.086	N.A	0.205
	F1-score	0.572	**0.666**	0.619	0.665	0.591	0.546	0.514	0.561	N.A	0.626
Reuters	Purity	0.579	0.255	0.321	0.347	0.448	0.287	**0.699**	0.528	N.A	0.450
	NMI	0.392	0.097	0.101	0.048	0.154	0.062	**0.482**	0.293	N.A	0.319
	F1-score	0.501	0.298	0.350	0.292	0.388	0.391	**0.582**	0.479	N.A	0.431
Robot	Purity	0.639	0.515	0.544	0.637	0.557	0.535	0.618	0.595	**0.683**	0.551
	NMI	0.071	0.046	0.035	0.056	0.017	0.019	0.068	0.032	**0.107**	0.032
	F1-score	0.551	**0.655**	0.592	0.538	0.521	0.522	0.564	0.522	0.585	0.643
Skating	Purity	0.208	0.183	0.222	0.221	0.220	**0.232**	0.228	0.221	0.227	0.215
	NMI	0.041	0.029	0.041	0.023	0.041	**0.112**	0.032	0.039	0.053	0.047
	F1-score	0.224	**0.252**	0.193	0.150	0.181	0.246	0.164	0.169	0.159	0.181
Unix	Purity	0.506	0.444	0.491	0.464	0.479	**0.756**	0.451	0.610	N.A	N.A
	NMI	0.110	0.004	0.094	0.031	0.055	**0.224**	0.018	0.235	N.A	N.A
	F1-score	0.385	**0.484**	0.422	0.308	0.371	0.227	0.438	0.430	N.A	N.A
Webkb	Purity	0.502	0.444	0.443	0.448	0.479	**0.723**	0.473	0.491	N.A	N.A
	NMI	0.094	0.020	0.055	0.019	0.072	**0.199**	0.126	0.078	N.A	N.A
	F1-score	0.452	0.421	0.455	0.378	0.426	0.341	0.403	**0.473**	N.A	N.A
平均值 (排名)	Purity	**0.584 (2.86)**	0.421 (7.36)	0.499 (5.14)	0.474 (5.36)	0.520 (4.57)	0.567 (3.79)	0.577 (3.21)	0.553 (3.29)	/	/
	NMI	**0.328 (2.35)**	0.077 (6.93)	0.201 (4.00)	0.081 (6.36)	0.185 (4.71)	0.151 (4.50)	0.274 (3.42)	0.230 (3.64)	/	/
	F1-score	**0.525 (2.79)**	0.479 (3.43)	0.476 (3.64)	0.368 (6.57)	0.456 (4.86)	0.303 (6.79)	0.496 (3.86)	0.479 (3.93)	/	/

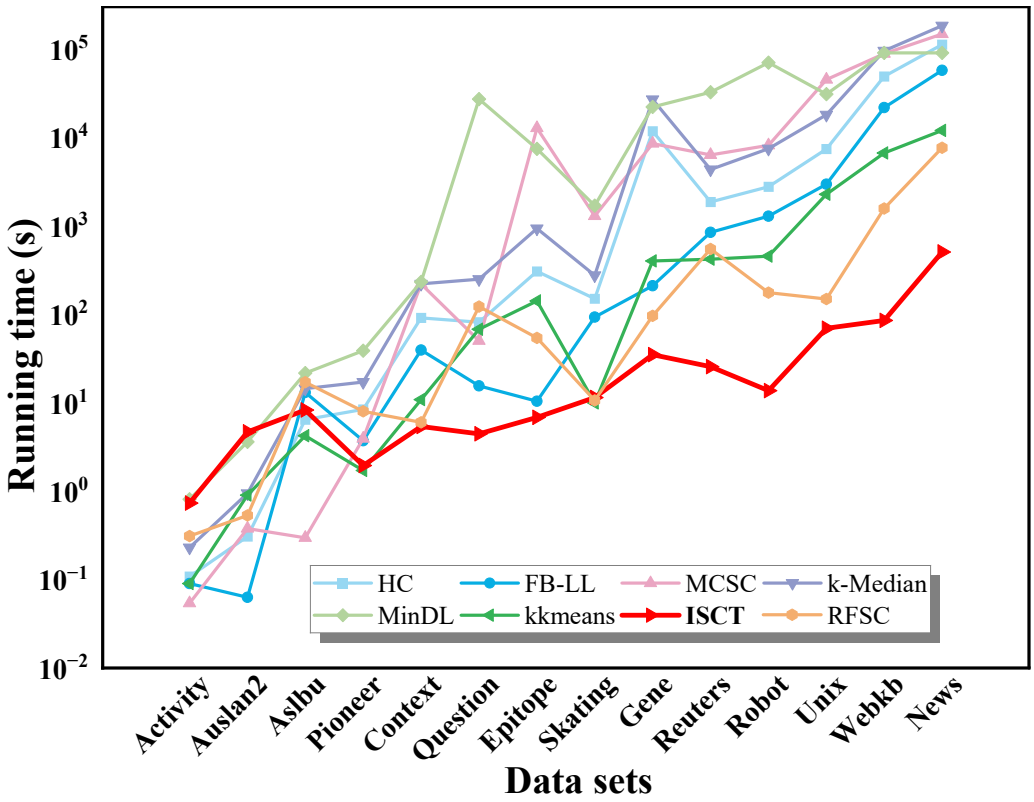


图 4.7　各聚类算法的运行时间比较

Fig. 4.7　Running time comparison of different methods

与基于特征的方法 FB-LL 相比，ISCT 在所有数据集上的纯度和 NMI 方面优于 FB-LL，而在 9 个数据集上的 F1-score 方面也优于 FB-LL。这种改进主要归因于 ISCT 采用的模式挖掘策略。在 FB-LL 中，特征向量是通过挖掘频繁模式生成的。然而，频繁模式通常在整个数据集中频繁出现，这意味着不同序列的转换特征向量可能是相似的。因此，挖掘的模式很少为聚类提供独特的信息。另一方面，通过采用随机投影和基于 LCS 的相似度度量，转换后的特征对于聚类更具信息性。这种方法增强了特征向量的区分能力，使其在聚类过程中更好地区分不同的序列。

与 kkmeans 和 k-Median 相比，ISCT 在大多数数据集上表现出更好的性能。此外，如图4.7所示，我们的方法表现出显著提高的效率。总之，ISCT 在保证可解释性的同时，实现了与最先进方法相当的性能，并且计算时间大大缩短。

(2) 可解释性

在聚类领域，仍然没有明确的定量标准来评估算法的可解释性[51]。通常，对于基于规则的方法，可解释性指标与规则的数量和长度有关[49]。同样，对于树结构，我们的目标是最小化分裂节点或模式的数量。我们努力使聚类使用更小的树，其中分裂标准尽可能简洁。

因此，本文的目标是在保持性能的同时，使用更少和更短的序列模式创建树。其

他序列聚类算法通常不将可解释性纳入设计中，使得它们不太适合直接比较。例如，在标准基于特征的聚类算法中，模式数量通常是预先确定的，通常在几百到几千之间变化。这些方法侧重于全面的数据表示，不优先考虑可解释性，而这是本方法的一个关键考虑因素。本文将本方法和基于随机森林的序列聚类算法 RFSC 进行比较[86]。这种比较侧重于每种方法中使用的模式的数量和长度。如图4.8所示的比较结果显示，ISCT 明显比 RFSC 使用更少的模式，特别是在较大的数据集（如“news”、“webkb”、“reuters”和“question”）上，同时保持竞争性能。此外，ISCT 的模式长度通常比大多数数据集上的 RFSC 短。这表明 ISCT 构建了明显更小的树，同时与 RFSC 相比，在模式长度使用上具有竞争力。

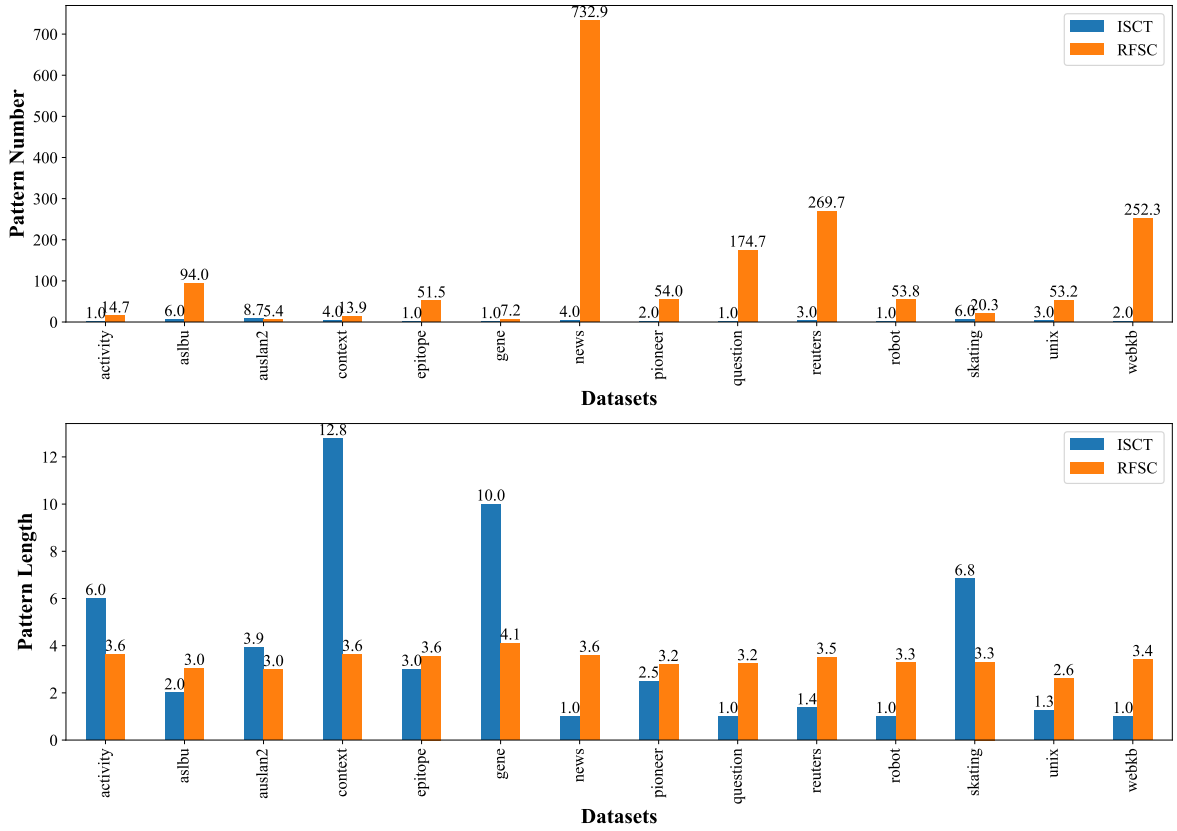


图 4.8　ISCT 和 RFSC 在平均每棵树上使用的模式数量和模式长度方面的比较

Fig. 4.8　Comparison of ISCT and RFSC in terms of utilized pattern number and pattern length per tree on average

为了具体说明本方法的可解释性，本文以“pioneer”数据集为例，绘制其聚类树。所有数据集的聚类树的详细可视化结果可在补充说明 1 中找到。此外，本文将 ISCT 得到的结果与从层次聚类方法中得到的结果进行比较。比较结果如图4.9所示。

本文的方法从随机投影的特征上的 K-Means 聚类结果（$k = 3$）作为伪标签开始。接下来，我们识别最具区分性的序列模式（‘11’、‘12’），将被模式命中的序列分组到一个簇中，如图4.9中的簇 3 所示。随后，我们再次对特征应用 K-Means（这次 $k = 2$），

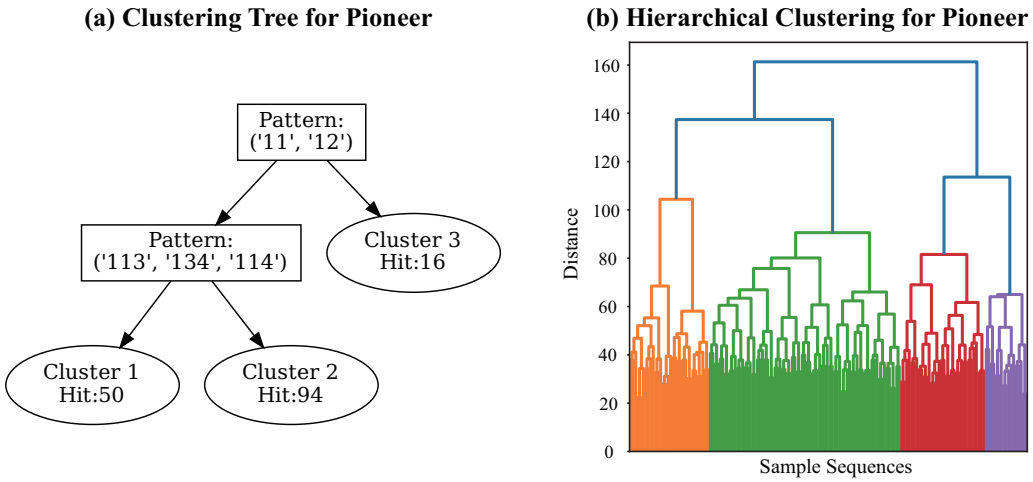


图 4.9　“Pioneer”数据集上序列聚类树和层次聚类的决策过程比较
Fig. 4.9　Comparison of the decision process of sequence clustering tree and hierarchical clustering on ”Pioneer”

发现另一个具有区分性的序列模式（‘113’、‘134’、‘114’），并形成另外两个簇，如图4.9中的簇 1 和簇 2 所示。

值得注意的是，我们的方法取得了出色的聚类性能，其纯度、NMI 和 F1-score 分别为 0.798、0.549 和 0.688。同时，我们的聚类方法保持了很高的可解释性，因为它仅使用了两个模式，且每个模式的长度非常短。这种识别性能和可解释性的平衡突显了我们方法在数据挖掘中的有效性。相比之下，由于层次聚类树依赖于距离度量，特别是在解释距离阈值大于 130 的情况下，对于“pioneer”数据集，理解决策过程面临着重大挑战。然后，从这种方法中获得可解释的结果是复杂的。此外，当将新样本分配给现有簇时，ISCT 也可以使用仅两个模式来确定其簇成员资格。相比之下，层次聚类方法需要对所有数据点进行距离计算，与 ISCT 相比，存在着相当大的效率和可解释性差距。

4.4　本章小结

本章介绍了可解释的序列聚类树（ISCT），旨在构建一个简洁且可解释的序列聚类决策树。ISCT 由 k 个叶节点组成，对应于 k 个簇，为序列聚类提供了透明且易于理解的决策过程。本文提出的方法是第一个专门为序列数据设计的可解释聚类算法。在使用 K-Means 获得序列聚类结果后，本文集成了一个序列模式发现算法来识别簇的特征。与现有的基于阈值的树不同，本方法提出了相对风险作为离散序列的分裂标准，并采用了一种增强方法来更新候选模式列表。对 14 个真实世界基准数据集的实验结果表明，本文提出的 ISCT 方法在保证可解释性的同时，实现了最先进的性能。

本文的目标是确定一组 $k-1$ 个模式并构建一棵简洁的树。然而，可能的候选模

式数量庞大，构建树的方法有很多。当前的方法是一个两阶段启发式方法，其中初始阶段影响后续阶段。在未来，本文的目标是开发一个联合优化算法，作为一个统一的过程运行，这可能将提供一个更连贯和高效的解决方案。

5 结论与展望

5.1 结论

本文围绕提高序列分类与聚类算法的可解释性和性能展开研究，提出了 Hamming 编码器神经网络和可解释序列聚类树两种新方法。在保证可解释性的同时均展现出优异的性能。实验结果表明，这两种方法在保证可解释性的同时均具有优异的性能。具体而言，Hamming 编码器神经网络在 11 个公共序列分类数据集上取得了与最新算法相当或更优的准确率，并有效揭示了模型的判别依据。此外，ISCT 算法在 14 个真实数据集的聚类任务中达到了当前最先进的性能，同时提供了清晰直观的聚类解释。上述研究成果表明，本文提出的方法在理论分析和实验验证的基础上，不仅提升了算法性能，还通过可解释性增强了模型的实用价值，展现了与传统方法相比的优势。

5.2 创新点

本研究的创新点主要体现在以下两个方面：

(1) 提出了 Hamming 编码器神经网络，该方法将卷积神经网络与序列判别模式挖掘相融合，通过卷积核权重二值化和基于 Hamming 距离的相似度度量，将判别模式的发现转化为神经网络参数优化问题，在显著提高分类准确率的同时保持了模型决策过程的透明。

(2) 设计了可解释序列聚类树，这一首个面向序列数据的可解释聚类算法。ISCT 以序列模式的存在与否作为树节点的分裂依据，并引入基于相对风险的分裂准则和增强的模式更新策略，在提供直观聚类解释的同时，取得了优于传统聚类方法的性能。

5.3 展望

尽管本研究取得了显著成果，仍存在进一步改进和探索的空间。在序列分类方面，可以探索支持含间隔的序列模式挖掘新方法，并改进梯度估计策略以降低卷积核权重二值化对分类精度的影响。在序列聚类方面，可以尝试将当前两阶段的聚类树构建过程改进为联合优化的单阶段算法，以获得更加连贯高效的聚类方案。此外，未来还可将上述方法推广到更多类型的序列数据或与其他模型相结合，不断拓展本研究成果的应用范围和影响力。本研究成果具有广泛的应用前景，例如在生物信息学领域帮助理解基因序列分类依据，在文本分析和金融科技领域提升模型决策可信度。然而，上述不足和改进方向为后续研究提供了新的期待和可能性。

参 考 文 献

[1] 保志康, 陈继璇, 刘印晓, et al. 基于机器学习的 DNA 序列分类研究 [J]. 生物化工, 2024, 10 (03): 20–27.

[2] 柏苛. 基于深度学习的文本序列分类方法及应用研究 [D]. [S. l.]: 山东建筑大学, 2024.

[3] Gupta A, Dengre V, Kheruwala H A, et al. Comprehensive review of text-mining applications in finance [J]. Financial Innovation, 2020, 6: 1–25.

[4] Xing Z, Pei J, Keogh E. A Brief Survey on Sequence Classification [J]. SIGKDD Explorations Newsletter, 2010, 12 (1): 40–48.

[5] 郭育洲, 周小安, 林洋. 基于神经网络的生物序列分类探析 [J]. 数字技术与应用, 2024, 42 (11): 152–156.

[6] 杜启蒙. 基于无监督深度聚类技术的生物信息序列分类研究 [D]. [S. l.]: 大理大学, 2024.

[7] Lesh N, Zaki M J, Ogihara M. Mining features for sequence classification [C]. In Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999: 342–346.

[8] 江冰, 谷飞洋, 何增有. 去冗余 Top-k 对比序列模式挖掘 [J]. 智能系统学报, 2018, 13 (5): 680–686.

[9] Zhou C, Cule B, Goethals B. Pattern Based Sequence Classification [J]. IEEE Transactions on Knowledge and Data Engineering, 2016, 28 (5): 1285–1298.

[10] He Z, Zhang S, Wu J. Significance-Based Discriminative Sequential Pattern Mining [J]. Expert Systems with Applications, 2019, 122: 54–64.

[11] Lam H T, Mörchen F, Fradkin D, et al. Mining compressing sequential patterns [J]. Statistical Analysis and Data Mining: The ASA Data Science Journal, 2014, 7 (1): 34–52.

[12] Egho E, Gay D, Boullé M, et al. A User Parameter-Free Approach for Mining Robust Sequential Classification Rules [J]. Knowledge and Information Systems, 2017, 52 (1): 53–81.

[13] De Smedt J, Deeva G, De Weerdt J. Mining Behavioral Sequence Constraints for Classification [J]. IEEE Transactions on Knowledge and Data Engineering, 2020, 32 (6): 1130–1142.

[14] Ghosh S, Yadav S, Wang X, et al. Dichotomic Pattern Mining Integrated With Constraint Reasoning for Digital Behavior Analysis [J]. Frontiers in Artificial Intelligence, 2022, 5.

[15] Ntagiou A N, Tsipouras M G, Giannakeas N, et al. Protein structure recognition by means of sequential pattern mining [C]. In Proceedings of the 17th International Conference on Bioinformatics and Bioengineering (BIBE), 2017: 334–339.

[16] Nguyen D, Luo W, Nguyen T D, et al. Sqn2vec: Learning sequence representation via sequential patterns with a gap constraint [C]. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases., 2018: 569–584.

[17] He Z, Wu Z, Xu G, et al. Decision Tree for Sequences [J]. IEEE Transactions on Knowledge and Data Engineering, 2023, 35 (1): 251–263.

[18] Ifrim G, Wiuf C. Bounded coordinate-descent for biological sequence classification in high dimensional predictor space [C]. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2011: 708–716.

[19] He Z, Xu G, Sheng C, et al. Reference-Based Sequence Classification [J]. IEEE Access, 2020, 8: 218199–218214.

[20] Bicego M, Murino V, Figueiredo M A. Similarity-based clustering of sequences using hidden Markov models [C]. In Proceedings of the Third International Conference on Machine Learning and Data Mining in Pattern Recognition, 2003: 86–95.

[21] Society T X, Wang S, Jiang Q, et al. A novel variable-order Markov model for clustering categorical sequences [J]. IEEE Transactions on Knowledge and Data Engineering, 2013, 26 (10): 2339–2353.

[22] Wang Y, Tian F. Recurrent residual learning for sequence classification [C]. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language processing, 2016: 938–943.

[23] Cheng H, Rao B, Liu L, et al. PepFormer: End-to-End transformer-based siamese network to predict and enhance peptide detectability based on sequence only [J]. Analytical Chemistry, 2021, 93 (16): 6481–6490.

[24] Yuan L, Wang W, Chen L. Two-stage pruning method for gram-based categorical sequence clustering [J]. International Journal of Machine Learning and Cybernetics, 2019, 10: 631–640.

[25] Li C, Dong X, Liu W, et al. Ssrdvis: Interactive visualization for event sequences summarization and rare detection [J]. Journal of Visualization, 2020, 23: 171–184.

[26] Li C, Yang Q, Wang J, et al. Efficient mining of gap-constrained subsequences and its various applications [J]. ACM Transactions on Knowledge Discovery from Data, 2012, 6 (1): 1–39.

[27] Steinegger M, Söding J. Clustering huge protein sequence sets in linear time [J]. Nature Communications, 2018, 9 (1): 2542.

[28] Ranjan C, Ebrahimi S, Paynabar K. Sequence graph transform (SGT): a feature embedding function for sequence data mining [J]. Data Mining and Knowledge Discovery, 2022, 36 (2): 668–708.

[29] Min W, Liang W, Yin H, et al. Explainable deep behavioral sequence clustering for transaction fraud detection [J]. arXiv preprint arXiv:2101.04285, 2021.

[30] Nadeem A, Verwer S. SECLEDS: sequence clustering in evolving data streams via multiple medoids and medoid voting [C]. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2022: 157–173.

[31] Ramoni M, Sebastiani P, Cohen P. Bayesian clustering by dynamics [J]. Machine Learning, 2002, 47: 91–121.

[32] Bose R J C, Van der Aalst W M. Context aware trace clustering: Towards improving process mining results [C]. In Proceedings of the 2009 SIAM International Conference on Data Mining, 2009: 401–

412.

[33] Oh S-J, Kim J-Y. A hierarchical clustering algorithm for categorical sequence data [J]. Information Processing Letters, 2004, 91 (3): 135–140.

[34] Hofmann D, Schleif F-M, Paaßen B, et al. Learning interpretable kernelized prototype-based models [J]. Neurocomputing, 2014, 141: 84–96.

[35] Biehl M, Hammer B, Villmann T. Prototype-based models in machine learning [J]. Wiley Interdisciplinary Reviews: Cognitive Science, 2016, 7 (2): 92–111.

[36] Hu Q, Yu D, Xie Z. Neighborhood classifiers [J]. Expert Systems with Applications, 2008, 34 (2): 866–876.

[37] Nova D, Estévez P A. A review of learning vector quantization classifiers [J]. Neural Computing and Applications, 2014, 25: 511–524.

[38] Angelov P, Soares E. Towards explainable deep neural networks (xDNN) [J]. Neural Networks, 2020, 130: 185–194.

[39] Zhang S, Chen X, Ran X, et al. Prioritizing causation in decision trees: A framework for interpretable modeling [J]. Engineering Applications of Artificial Intelligence, 2024, 133: 108224.

[40] Good J, Kovach T, Miller K, et al. Feature learning for interpretable, performant decision trees [J]. Advances in Neural Information Processing Systems, 2023, 36: 66571–66582.

[41] Speith T. A review of taxonomies of explainable artificial intelligence (XAI) methods [C]. In Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, 2022: 2239–2250.

[42] Lakkaraju H, Bach S H, Leskovec J. Interpretable decision sets: A joint framework for description and prediction [C]. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016: 1675–1684.

[43] Qiao L, Wang W, Lin B. Learning accurate and interpretable decision rule sets from neural networks [C]. In Proceedings of the AAAI Conference on Artificial Intelligence, 2021: 4303–4311.

[44] Yang H, Jiao L, Pan Q. A survey on interpretable clustering [C]. In Proceedings of the 40th Chinese Control Conference (CCC), 2021: 7384–7388.

[45] Wang X, Liu X, Zhang L. A rapid fuzzy rule clustering method based on granular computing [J]. Applied Soft Computing, 2014, 24: 534–542.

[46] Hsieh C-H, Yan C-H, Mao C-H, et al. GMiner: Rule-based fuzzy clustering for Google Drive behavioral type mining [C]. In Proceedings of the 2016 International Computer Symposium (ICS), 2016: 98–103.

[47] Pelleg D, Moore A W. Mixtures of Rectangles: Interpretable Soft Clustering [C]. In Proceedings of the International Conference on Machine Learning, 2001: 401–408.

[48] Chen J, Chang Y, Hobbs B, et al. Interpretable clustering via discriminative rectangle mixture model [C]. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining, 2016:

823–828.

[49] Lawless C, Kalagnanam J, Nguyen L M, et al. Interpretable clustering via multi-polytope machines [C]. In Proceedings of the AAAI Conference on Artificial Intelligence, 2022: 7309–7316.

[50] Moshkovitz M, Dasgupta S, Rashtchian C, et al. Explainable k-means and k-medians clustering [C]. In Proceedings of International Conference on Machine Learning, 2020: 7055–7065.

[51] Bertsimas D, Orfanoudaki A, Wiberg H. Interpretable clustering: an optimization approach [J]. Machine Learning, 2021, 110: 89–138.

[52] Basak J, Krishnapuram R. Interpretable hierarchical clustering by constructing an unsupervised decision tree [J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17 (1): 121–132.

[53] Fraiman R, Ghattas B, Svarc M. Interpretable clustering using unsupervised binary trees [J]. Advances in Data Analysis and Classification, 2013, 7: 125–145.

[54] Jiao L, Yang H, ga Liu Z, et al. Interpretable fuzzy clustering using unsupervised fuzzy decision trees [J]. Information Sciences, 2022, 611: 540–563.

[55] Bandyapadhyay S, Fomin F V, Golovach P A, et al. How to find a good explanation for clustering? [J]. Artificial Intelligence, 2023, 322: 103948.

[56] Makarychev K, Shan L. Explainable k-means: don’t be greedy, plant bigger trees! [C]. In Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing, 2022: 1629–1642.

[57] Nagel M, Fournarakis M, Bondarenko Y, et al. Overcoming Oscillations in Quantization-Aware Training [C]. In Proceedings of the 39th International Conference on Machine Learning, 2022: 16318–16330.

[58] Lee J, Kim D, Ham B. Network quantization with element-wise gradient scaling [C]. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021: 6448–6457.

[59] Bengio Y, Léonard N, Courville A. Estimating or propagating gradients through stochastic neurons for conditional computation [J]. arXiv preprint arXiv:1308.3432, 2013.

[60] Wang Z, Zhang W, Ning L, et al. Transparent classification with multilayer logical perceptrons and random binarization [C]. In Proceedings of the AAAI Conference on Artificial Intelligence, 2020: 6331–6339.

[61] Qiao L, Wang W, Lin B. Learning accurate and interpretable decision rule sets from neural networks [C]. In Proceedings of the AAAI Conference on Artificial Intelligence, 2021: 4303–4311.

[62] Wang Z, Zhang W, Liu N, et al. Scalable rule-based representation learning for interpretable classification [J]. Advances in Neural Information Processing Systems, 2021, 34: 30479–30491.

[63] Wang W, Qiao L, Lin B. Tabular machine learning using conjunctive threshold neural networks [J]. Machine Learning with Applications, 2022, 10: 100429.

[64] Jamshed A, Mallick B, Kumar P. Deep learning-based sequential pattern mining for progressive database [J]. Soft Computing, 2020, 24: 17233–17246.

[65] Nowak J, Korytkowski M, Scherer R. Discovering Sequential Patterns by Neural Networks [C]. In

Proceedings of the 2020 International Joint Conference on Neural Networks, 2020: 1–6.

[66] Jiang L, Bosch N. Predictive sequential pattern mining via interpretable convolutional neural networks [C]. In Proceedings of the 14th International Conference on Educational Data Mining, 2021.

[67] Collery M, Bonnard P, Fages F, et al. Neural-based classification rule learning for sequential data [C]. In Proceedings of the Eleventh International Conference on Learning Representations, 2023.

[68] Feremans L, Vercruyssen V, Cule B, et al. Pattern-Based Anomaly Detection in Mixed-Type Time Series [C]. In Proceedings of the Machine Learning and Knowledge Discovery in Databases, 2020: 240–256.

[69] Di Ciccio C, Mecella M. A two-step fast algorithm for the automated discovery of declarative workflows [C]. In Proceedings of the 2013 IEEE Symposium on Computational Intelligence and Data Mining, 2013: 135–142.

[70] Maggi F M, Bose R J C, van der Aalst W M. Efficient discovery of understandable declarative process models from event logs [C]. In Proceedings of 24th International Conference on Advanced Information Systems Engineering, 2012: 270–285.

[71] Maggi F M, Montali M, Di Ciccio C, et al. Semantical vacuity detection in declarative process mining [C]. In Proceedings of the 14th International Conference on Business Process Management, 2016: 158–175.

[72] Gama J, Zliobaite I, Bifet A, et al. A Survey on Concept Drift Adaptation [J]. ACM Computing Surveys, 2014, 46 (4): 44.

[73] Zhang J, Wang Y, Yang D. CCSpan: Mining closed contiguous sequential patterns [J]. Knowledge-Based Systems, 2015, 89: 1–13.

[74] Gong Y, Liu L, Yang M, et al. Compressing deep convolutional networks using vector quantization [J]. arXiv preprint arXiv:1412.6115, 2014.

[75] Shvo M, Li A C, Icarte R T, et al. Interpretable sequence classification via discrete optimization [C]. In Proceedings of the AAAI Conference on Artificial Intelligence, 2021: 9647–9656.

[76] Demšar J. Statistical comparisons of classifiers over multiple data sets [J]. Journal of Machine Learning Research, 2006, 7: 1–30.

[77] Liu H, Setiono R. Chi2: Feature selection and discretization of numeric attributes [C]. In Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence, 1995: 388–391.

[78] Kadappa V, Negi A. Computational and space complexity analysis of SubXPCA [J]. Pattern Recognition, 2013, 46 (8): 2169–2174.

[79] Yang G. Computational aspects of mining maximal frequent patterns [J]. Theoretical Computer Science, 2006, 362 (1-3): 63–85.

[80] Asuncion A, Newman D. UCI machine learning repository. 2007.

[81] Guralnik V, Karypis G. A scalable algorithm for clustering sequential data [C]. In Proceedings of the 2001 IEEE International Conference on Data Mining, 2001: 179–186.

[82] Xu K, Chen L, Wang S. A Multi-view Kernel Clustering framework for Categorical sequences [J]. Expert Systems with Applications, 2022, 197: 116637.

[83] Dinu L P, Ionescu R T. Clustering based on median and closest string via rank distance with applications on DNA [J]. Neural Computing and Applications, 2014, 24: 77–84.

[84] Chen Y, Xu P, Ren L. Sequence synopsis: Optimize visual summary of temporal event data [J]. IEEE Transactions on Visualization and Computer Graphics, 2017, 24 (1): 45–55.

[85] Jahanshahi H, Baydogan M G. nTreeClus: A tree-based sequence encoder for clustering categorical series [J]. Neurocomputing, 2022, 494: 224–241.

[86] Jiang M, Wang J, Hu L, et al. Random forest clustering for discrete sequences [J]. Pattern Recognition Letters, 2023, 174: 145–151.

攻读硕士学位期间科研项目及科研成果

已发表论文

[1] **Dong J**, Yang X, Jiang M, Hu L, He Z. Interpretable sequence clustering[J]. ***Information Sciences***, 2025, 689: 121453. (CCF B 类期刊, 本学位论文第四章)

[2] Hu L, **Dong J**, Jiang M, Liu Y, He Z. Clusterability test for categorical data[J]. ***Knowledge and Information Systems***, 2025: 1–26. (CCF B 类期刊)

[3] Hu L, Jiang M, **Dong J**, Liu X, He Z. Interpretable categorical data clustering via hypothesis testing[J]. ***Pattern Recognition***, 2025: 111364. (CCF B 类期刊)

[4] He Z, Li Z, **Dong J**, Liu X, Jiang M, Hu L. Conjunction subspaces test for conformal and selective classification[J]. ***Information Sciences***, 2025, 707: 122037. (CCF B 类期刊)

[5] He Z, Hu L, He J, **Dong J**, Jiang M, Liu X. Significance-based interpretable sequence clustering[J]. ***Information Sciences***, 2025: 121972. (CCF B 类期刊)

[6] Yang Y, Cai S, Mo C, **Dong J**, Chen S, Wen Z. Profiles of antibiotic resistome risk in diverse water environments[J]. ***Communications Earth & Environment***, 2025, 6(1): 158. (中科院一区 Top 期刊, Nature 旗下期刊)

待发表论文

[1] **Dong J**, Jiang M, Hu L, He Z. Hamming Encoder: Mining Discriminative k-mers for Discrete Sequence Classification[J]. ***Data Mining and Knowledge Discovery***, Accept. (CCF B 类期刊, 本学位论文第三章)

[2] **Dong J**, Lyu Z, Ke Q. Towards Understanding Evolution of Science through Language Model Series[J]. ***ACM Transactions on Intelligent Systems and Technology***, Under Review. (CCF B 类期刊)

[3] Ke Q, **Dong J**. Chemical substructure sequences meet language models for explainable property prediction[J]. ***Nature Machine Intelligence***, Waiting for Revision. (Nature 子刊)

[4] Li P, **Dong J**, Jiang M, Liu X, Hu L, He Z. Interpretable Sequence Classification via Decision Set[J]. ***IEEE Transactions on Knowledge and Data Engineering***, Major Revision. (CCF A 类期刊)

致　谢

感谢导师在我人生与学术道路上的无私引领。遥记得第一次遇到老师是在旁听他课的路上，应该是老师第一次去盘锦校区吧，穿着冲锋衣，戴着鸭舌帽，大风吹得我俩脸色发白，但是我还是一眼认出，这就是我在互联网上网络聊天许久的何教授！从初识相交到成为门下学子，老师如明灯般照亮我人生的每一步旅程。在我分享成功喜悦的璀璨时刻，是老师的身影让喜悦倍增；在我跌入困境迷茫的黑暗时刻，依然是老师的智慧之光指引我前行。老师的教诲如春风化雨，润物无声，让我在人生旅途上不断成长，永志难忘。

课题组的好伙伴们，又是一年分别的季节，旅程有你们中留下的美好记忆。大师姐、胡博士、蒋博士、小刘博士，我其实不爱吃鱼的，但是我看那天大家一人一个炸鱼我不啃显得我很另类就跟着吃了 · · · · · 大师姐，我自诩已经快成为小彭于晏了，没有辜负大师姐给的健身卡喔！胡博士，有空一起看海，大连海之韵我还想去第二次！蒋博士，祝你多发 Paper！小刘博士要坚持运动，别给自己太多压力。小奇，凯哥，和你们一起我才知道原来除了学校食堂还有别的洞天福地（后悔去少了）！依涵、泽润、孝雷（这是能排序的么?），祝我们都各奔未来，行有所至！小萱，鹏举，祝你俩多赚钱！希望师弟师妹们都能发好 Paper，找到好工作！

感谢柯庆老师对我的指导，还有子优，周姐，吕总，港漂一年，有你们的陪伴，我们谈天说地，共同成长。

感谢老爸老妈，说实话我自己感觉自己傻傻的，你们也纵容我傻傻的，这种支持也是一种幸福吧。

一晃悠又是三年在大工，此时的我不是“start my new journey in another major”，也不再“wish to be the best person in whatever the area that I am focused on”。但与之前的我一致的是“I want myself to be useful and meaningful to this world”。