智能系统学报
CAAI TRANSACTIONS ON INTELLIGENT SYSTEMS

一种新的稀疏表示驱动的大规模数据集成聚类算法

何玉林, 杨振宇, 肖又旗, 黄哲学

引用本文:
何玉林, 杨振宇, 肖又旗, 等. 一种新的稀疏表示驱动的大规模数据集成聚类算法[J]. 智能系统学报, 2026, 21(4): 888-907.
HE Yulin, YANG Zhenyu, XIAO Youqi, et al. A new sparse representation-driven ensemble clustering algorithm for large-scale data[J]. *CAAI Transactions on Intelligent Systems*, 2026, 21(4): 888-907.

在线阅读 View online: https://dx.doi.org/10.11992/tis.202509003

DOI: 10.11992/tis.202509003
网络出版地址: https://link.cnki.net/urlid/23.1538.tp.20260402.1725.002

一种新的稀疏表示驱动的大规模数据集成聚类算法

何玉林[1,2], 杨振宇[1,2], 肖又旗[1,2], 黄哲学[1,2]
(1. 人工智能与数字经济广东省实验室(深圳), 广东 深圳 518107; 2. 深圳大学 计算机与软件学院, 广东 深圳 518060)

摘　要: 针对大规模高维数据聚类中存在的采样代表性不足与计算开销过高的问题, 本文提出了一种稀疏表示驱动的集成聚类(sparse representation-driven ensemble clustering, SREC)算法。在基聚类生成阶段, SREC 算法引入了随机样本划分与 K-means 的混合代表点采样策略, 结合 FAISS(facebook AI similarity search)索引构建高效稀疏图, 有效克服了局部采样难以覆盖全局分布的缺陷; 在共识函数构建阶段, SREC 算法采用了一种非迭代的加权谱共识函数, 利用 Tcut(transfer cut)策略在稀疏图中实现高精度的簇划分。在分布式环境下基于 Spark 计算框架实现了 SREC 算法, 通过百万级数据集对其表现进行了系统性地验证。实验结果表明: SREC 算法在评价指标 NMI(normalized mutual information)、ARI(adjusted Rand index)和 ACC(clustering accuracy)上优于选用的 10 种主流聚类算法, 较之次优算法获得了 4.15%、3.33% 以及 0.98% 的性能提升; 同时, SREC 算法具备良好的稳定性, 对基聚类算法个数的变化不敏感, NMI 指标标准差仅为 0.21%; 另外, 在处理五百万级样本数据的聚类问题时, 相比次优算法, SREC 算法的计算效率提升了 51.8%, 证实了其在处理大规模数据聚类问题时的优势。本文研究结果可为大规模数据的高效聚类分析、分布式数据挖掘及相关智能应用系统设计提供参考。

关键词: 大规模聚类; 稀疏表示; 集成聚类; 代表点采样; 分布式计算; Spark; 大规模数据; 共识函数
中图分类号: TP391.4　　**文献标志码**: A　　**文章编号**: 1673−4785(2026)04−0888−20

中文引用格式: 何玉林, 杨振宇, 肖又旗, 等. 一种新的稀疏表示驱动的大规模数据集成聚类算法 [J]. 智能系统学报, 2026, 21(4): 888–907.
英文引用格式: HE Yulin, YANG Zhenyu, XIAO Youqi, et al. A new sparse representation-driven ensemble clustering algorithm for large-scale data[J]. CAAI transactions on intelligent systems, 2026, 21(4): 888–907.

A new sparse representation-driven ensemble clustering algorithm for large-scale data

HE Yulin[1,2], YANG Zhenyu[1,2], XIAO Youqi[1,2], HUANG Zhexue[1,2]
(1. Guangdong Laboratory of Artificial Intelligence and Digital Economy(Shenzhen), Shenzhen 518107, China; 2. College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China)

Abstract: To address the challenges of inadequate sampling representativeness and high computational cost in large-scale high-dimensional data clustering, this paper proposes a sparse representation-driven ensemble clustering(SREC) algorithm. In the base clustering generation stage, SREC algorithm incorporates a hybrid sampling strategy that combines random sample partitioning with K-means, along with facebook AI similarity search(FAISS) indexing to build an efficient sparse graph. This approach effectively mitigates the limitation of local sampling in capturing the global data distribution. For the consensus function construction, a non-iterative weighted spectral consensus function is introduced, which employs the Tcut strategy to achieve high-precision cluster partitioning within the sparse graph. The SREC algorithm is implemented in a distributed environment using the Spark computing framework, and its performance is systematically evaluated on datasets containing over one million samples. Experimental results show that SREC outperforms ten selected mainstream clustering algorithms on the NMI, ARI, and ACC evaluation metrics, with improvements of 4.15%, 3.33% and 0.98%, respectively, over the second-best algorithm. Moreover, SREC algorithm exhibits strong stability, demonstrating low sensitivity to changes in the number of base clusters, as reflected by an NMI standard deviation of only 0.21%. Furthermore, when applied to clustering tasks with five million samples, SREC algorithm achieves a 51.8% increase in computational efficiency compared to the next-best algorithm, confirming its advantages for large-scale data clustering. The findings of this study can provide a reference for efficient clustering analysis of large-scale data, distributed data mining, and the design of related intelligent application systems.

Keywords: large-scale clustering; sparse representation; ensemble clustering; representative point selection; distributed computing; Spark; large-scale data; consensus function

随着信息技术的飞速发展与行业数据获取能力的不断提升, 我们正步入一个由海量、高维和异构数据构成的“大数据”时代。科学计算、社交

收稿日期: 2025−09−01.　**网络出版日期**: 2026−04−03.
基金项目: 深圳市科技重大专项项目 (KJZD20230923114809020); 深圳市基础研究面上项目 (JCYJ20250604175602004).
通信作者: 杨振宇. E-mail: yangzhenyu@gml.ac.cn.

网络和物联网等领域生成的海量数据，蕴含着巨大的商业价值，给传统数据分析技术带来了前所未有的挑战。在众多数据分析方法中，聚类作为机器学习的核心分支，其重要性愈发凸显[1-2]。它的根本目标是在无先验标签的条件下，揭示数据内在的群组结构，在模式发现、数据归纳和异常检测等实际应用中发挥着至关重要的作用[3]。然而，随着数据规模的急剧增加，传统的单机运行的聚类算法面临着严重的计算效率瓶颈，导致其难以高效地处理大规模和高维数据集的聚类问题。

面对大规模数据集，谱聚类（spectral clustering, SC）算法因其在处理非凸簇结构方面的卓越能力而备受关注。作为一种基于图论的聚类算法，SC 将聚类问题巧妙地转化为图划分问题，通过分析数据相似性图的谱特性，将数据嵌入到一个新的低维空间，使得簇结构在该空间中线性可分。这种基于图论的思想使其能够超越传统距离度量的局限性。然而，SC 算法的强大能力是以高昂的计算代价为前提的。对于规模为 N 的数据集，其标准实现需要构建一个 $N\times N$ 的稠密相似度矩阵，时间与空间复杂度均高达 $O(N^2)$，随后的特征分解步骤更是达到了 $O(N^3)$ 的时间复杂度。当数据规模 N 达到百万级别时，这种计算和存储的双重高复杂度使得 SC 算法对硬件资源的要求非常高，严重阻碍了其在真实世界大规模聚类问题中的应用[4-5]。

近年来，为了应对这一难题，稀疏表示理论（sparse representation, SR）被提出，旨在通过揭示数据内在的低维结构来简化对高维数据的处理难度。该理论强调数据点间的自表达性，即每个数据点可以通过同一子空间内的其他数据点的稀疏线性组合来表示。基于稀疏表示的聚类算法——子空间聚类（sparse subspace clustering, SSC）[6-7] 能够通过构建稀疏相似度图，揭示数据的潜在子空间结构。然而，SSC 算法的标准实现需要求解一个大规模稀疏优化问题，计算复杂度较高，难以在大规模数据集上直接应用。这一计算瓶颈不仅存在于 SSC 算法中，也是许多先进聚类算法共同面临的挑战。

尽管前述的 SC 和 SSC 等算法在特定场景下展现出独特优势，但它们往往难以同时满足大规模数据聚类面临的处理效率和聚类精度的双重要求。在此背景下，集成聚类（ensemble clustering, EC）作为一种可行的解决方案，提供了一种突破单一算法局限性、进而提升大数据聚类问题处理能力的思路。EC 通过巧妙地融合多个基础聚类结果，不仅能够有效降低对单个算法计算复杂度的依赖，还能通过多样性集成显著提升聚类结果的鲁棒性和准确性，特别是在处理具有复杂结构的大规模数据集时展现出明显优势[8-9]。然而，现有的 EC 算法通常依赖于计算简单但假设过于简化的基础聚类方法[10]，这限制了其在复杂结构数据中的表现。此外，集成算法本身也面临着基础聚类质量不稳定而带来的共识结果偏差问题，因此如何构建能够有效捕获数据内在结构、稳定性强的基础聚类成为 EC 能否成功的关键。

上述算法在处理非凸簇结构、揭示子空间结构或提高聚类鲁棒性等特定方面虽已证明有效，但在面对百万级样本规模甚至更高规模的数据集时，特别是在分布式集群环境下，其较高的计算复杂度、较强的硬件依赖性以及较窄的平台适用性成为制约其性能表现的主要因素。

首先，许多先进的聚类算法其设计的核心逻辑仍是面向单机运行环境的。当数据规模超出单机内存容量，或单机计算能力无法在合理时间内完成处理时，为增强对大规模聚类问题的处理能力，普遍采用基于代表点的近似策略来降低计算复杂度。然而，常用的随机采样方法存在显著缺陷，随机选取的样本点往往无法充分覆盖数据的全局流形结构，尤其对于规模不均或形状复杂的簇，易导致采样偏差，从而影响最终聚类结果的准确性与稳定性。

其次，当试图将这些算法，尤其是集成聚类算法移植到分布式集群环境下时，其训练流程的高迭代性与对全局数据的依赖性构成了根本性障碍——许多经典共识函数的核心操作（如构建全局相似度矩阵）天然依赖全局数据视图，这要求集群节点间进行海量的数据交换和频繁的全局同步等待操作，导致计算并行化严重受限、通信开销随规模急剧膨胀，同时需设计复杂的分布式协调机制处理数据划分、任务调度和容错，最终使得算法在分布式环境下的计算效率低下，难以真正发挥集群的并行计算优势。

因此，如何设计一种能够克服上述瓶颈的大规模聚类算法，稳定选取能充分表征数据全局结构的高质量代表点，构建低通信开销、非迭代的分布式共识函数，充分利用分布式计算资源，是当前大规模聚类研究领域面临的核心挑战。为此，本文提出了一种新的稀疏表示驱动的大规模数据集成聚类（sparse representation-driven ensemble clustering, SREC）算法，该算法的实现过程包括两个核心步骤：

1) 设计高稳定稀疏表示聚类(sparse representation clustering, SRC)算法，该算法通过融合基于随机样本划分(random sample partition, RSP)[11-12]的 K-means 代表点选择策略，确保了所选代表点能够充分覆盖数据的整体分布与准确捕获关键的局部特征，有效克服了纯随机采样带来的不稳定性。同时，结合基于 FAISS(facebook AI similarity search)向量索引库的快速稀疏图生成技术，实现了从数据到图的高效转换，从而加速了相似度图的构建过程。

2) 实现非迭代的集成聚类算法 SREC，将 SRC 作为强大的基础聚类器，并在单机层面设计了一种非迭代的加权图谱共识函数，该函数通过生成稀疏相似矩阵并应用转移切割(transfer cut, Tcut)技术直接在图结构中识别簇，避免传统迭代共识算法中频繁的全局数据同步与迭代通信开销，通过单台主机一次性完成图构建与簇识别，显著提升分布式环境下的整体计算效率。

为系统评估所提出的 SRC 与 SREC 算法在聚类准确率和计算效率方面的性能，本研究在百万级大规模真实与仿真数据集上进行了 3 方面有针对性的实验验证：1)代表点选择方法可行性实验，旨在验证混合代表点选择策略能否有效捕捉数据全局结构并提升采样质量；2)算法有效性实验，将所提出方法与 K-means[13]、Nyström[14]、超大规模谱聚类(ultra-scalable spectral clustering, U-SPEC)[15]、快速自监督聚类(fast self-supervised clustering, FSSC)[16]、LiteWSC[17]、证据积累聚类(evidence accumulation clustering, EAC)[18]、谱集成聚类(spectral ensemble clustering, SEC)[19]、ECPCS-MC[20]、超可扩展集成聚类(ultra-scalable ensemble clustering, U-SENC)[15] 和 LiteWSEC[21] 共 10 种经典聚类与集成聚类算法进行对比，评估其综合性能；3)算法参数合理性分析实验，针对 SRC 算法的代表点个数 p 和最近代表点数量 K 以及 SREC 算法的基聚类个数 m，考察不同参数取值对聚类准确率与计算效率的影响规律。

实验结果显示，混合代表点选择策略能有效捕捉数据全局结构并提升采样质量；在算法有效性对比中，本文的方法虽未始终达到最优准确率但接近最优水平，并显著提升了计算效率，有效平衡了效率与性能之间的矛盾；参数验证实验则明确了关键参数对性能的影响趋势。综合来看，相较于当前最先进的大规模聚类与集成聚类方法，SRC 与 SREC 算法在计算效率与聚类准确率方面均展现出显著优势。

1　大规模聚类算法与集成共识机制

1.1　大规模近似聚类算法

传统聚类算法，如 K-means 和 SC，在中小规模数据集上表现良好。但这些算法计算复杂度高、内存占用大，无法有效处理大规模数据的聚类问题。为解决此问题，研究者提出多种近似方法来提高算法处理大规模数据的能力。然而，这些方法通常在提升效率的同时会显著降低聚类质量。本节回顾几种代表性的大规模近似聚类算法，展示它们在采样策略、代表点选择和图构建等方面采用的核心技术，并分析其在稳定性、全局结构感知以及误差累积方面的不足。

Zhu 等[22] 指出了小批量方法中的陈旧性问题，提出了陈旧性规约的小批量 K-means 算法，其收敛速度比标准小批量方法快 40~130 倍，并且最终能达到的聚类损失也更低。李大瑞等[23] 提出了鲁棒近似 K-means 算法，该算法在近似搜索的基础上，通过在迭代中利用更多历史信息来更新数据分配，从而更好地保证了算法的收敛性与聚类性能，实现了损失的非递增和快速下降。

为解决谱聚类的扩展性难题，研究者们探索了多种近似策略，其中基于子矩阵近似的方法尤为突出。Fowlkes 等[14] 率先将 Nyström 方法应用于谱聚类，通过随机抽取 l 个样本点来构建一个低秩矩阵，用以近似完整的亲和度矩阵。该方法将计算复杂度从 $O(N^2d)$ 降低到以 $O(Nld)$ 或 $O((N-l)l^2)$ 为主导项的时间水平，但其聚类质量高度依赖于初始随机采样的代表性。为提升代表点的质量，Chen 等[24] 提出地标谱聚类，采用 K-means 选取代表点(即地标点)，并为每个数据点寻找 K 个最近邻地标来构建稀疏的 $N\times p$ 亲和度矩阵。尽管这些方法有效降低了复杂度，但仍面临 $O(Np)$ 的计算复杂度瓶颈，且一次性的子矩阵近似可能影响聚类结果的鲁棒性。

另一类加速谱聚类的思路则更为直接，它旨在绕过高代价的特征分解步骤。研究发现，谱聚类中的归一化切割目标函数在数学上等价于加权核 K-means 的迹最大化问题。基于此，贾洪杰等[25] 提出了一种近似加权核 K-means 算法，该算法用 K-means 的迭代优化过程替代了特征分解。为避免计算完整的核矩阵，该方法仅使用部分核矩阵进行近似计算，从而在保证聚类性能的同时，大幅降低了时间与空间复杂度。但其性能转而依赖于核矩阵近似的质量以及 K-means 优化过程的收敛情况。在此基础上，更高效的锚点策略被相继

提出。Huang 等[15] 设计的 U-SPEC 算法采用混合策略高效选取代表点, 并结合二部图划分将最耗时的特征分解复杂度降至$O(NK(K+k)+p^3)$, 使主要时间开销降至$O(Np^{1/2}d)$。然而, 其多层近似策略在提升效率的同时, 也可能引入累计误差, 导致结果存在一定的随机性波动。Wang 等[16] 提出的 FSSC 算法更为复杂, 它融合了层次 K-means、二部图构建、伪标签半监督学习以及标签传播等多个阶段。该方法虽然通过锚点和矩阵求逆引理等技术将复杂度优化至$O(nmd)$, 但其多阶段流程也引入了误差累积的风险, 最终结果的鲁棒性高度依赖于每个环节的中间产出质量。

另一条加速路径是直接在降维后的数据集上执行聚类。Yan 等[26] 提出的局部 K-means 聚类算法, 利用 K-means 将数据集压缩为p个簇中心, 再仅对这p个中心点执行标准谱聚类。这种方法虽然极致简化了计算, 但其性能完全取决于初始质心能否有效保持原始数据的流形结构。Yang 等[17] 的 LiteWSC 算法则采用了更大幅度的降维策略, 它首先从原始数据中随机抽取一部分样本$l \ll N$, 再仅对该子集运行 K-means 以生成p个代表点。这种设计极大提升了原型选择阶段的效率, 使其能应对超大规模数据。然而, 该算法的根本局限在于其性能完全依赖最初的随机采样, 一次质量不佳的采样便可能导致后续所有步骤偏离真实的数据分布, 从而忽略对全局数据结构的感知。

1.2　集成聚类共识函数设计

EC 旨在融合多个基础划分以获得比任一单一划分更为稳健和准确的结果。这一高阶学习框架在应对异构数据、挖掘潜在结构以及提升聚类稳定性方面展现出显著优势。EC 算法的设计流程主要包括两个阶段: 1)基础划分生成: 通过不同算法、参数设定或数据子集, 产生一组差异性结果; 2)共识函数设计: 将这些划分整合为最终一致解。有效的集成要求基础划分兼具高质量与高多样性。其中, 共识函数的设计是关键环节, 现有研究大体沿 3 条技术路径展开。

共关联矩阵方法最早由 Fred 等[18] 在 EAC 中系统提出。其核心思想是把每个基聚类视为一次投票, 统计样本对在同一个簇中出现的频次, 构成$N \times N$的共关联矩阵(co-association matrix, CAM), 再以层次聚类或谱聚类作用于 CAM 得到一致划分。在此基础上, Liu 等[19] 提出了 SEC 算法, 并进一步证明了对 CAM 的谱聚类等价于对加权 K-means 的求解, 将复杂度由$O(N^3)$降至$O(N)$。为了缓解大样本带来的存储压力, Yang 等[21] 提出了一种轻量级框架用于 web 规模的光谱集成聚类(lightweight framework for web-scale spectral ensemble clustering, LiteWSEC)算法, 通过随机采样将 CAM 规模压缩至$l \times l$($l \ll N$), 在仅 1 GB 内存下即可处理 800 万级数据, 为 Web-scale 场景提供了可行方案。

图分割方法则将集成信息建模为图结构, 并通过图划分实现一致性聚类。Strehl 等[8] 提出基于聚类的相似度划分算法、超图划分算法与元聚类算法 3 种超图分割算法, 奠定了该方向的研究基础。针对超大规模任务, Huang 等[15] 提出的 U-SENC 算法, 该算法先利用轻量级谱聚类 U-SPEC 生成基划分, 再构建对象–簇二分图, 并通过转移割在$O(N\sqrt{pd})$时间内完成共识, 实验表明其可稳健处理千万级样本。进一步地, Huang 等[20] 提出基于层次共识函数和基于元聚类共识函数的簇间相似性传播集成聚类(ensemble clustering by propagating cluster-wise similarities with meta-cluster based consensus function, ECPCS-MC)算法, 引入簇与簇之间的多层图, 通过随机游走传播簇间相似度, 并在对象级或簇级执行分割, 从而兼顾全局结构与局部细节。

目标优化方法通过显式目标函数形式化一致性要求, 并设计优化策略求解。Wu 等[27] 提出基于 K-means 的共识聚类算法, 以加权互信息为目标并采用 K-means 式迭代优化。Liu 等[28] 的基于熵的共识聚类算法则将熵最小化转化为谱聚类问题, 并在理论上证明了算法的单调收敛性。Li 等[29] 提出的加权共识聚类借助非负矩阵分解对目标函数解耦, 实现了基聚类的自动加权。该类方法具有良好的可解释性, 但在大规模数据场景中仍需结合二次采样或图稀疏化技术以提升可扩展性。

2　SRC 算法和 SREC 算法设计

针对大规模数据集谱聚类所面临的可扩展性与鲁棒性双重挑战, 本研究提出一种新的基于稀疏表示驱动的大规模数据聚类算法——稀疏表示聚类 SRC 算法。该算法的核心思想是通过高效的代表点选择机制、基于稀疏表示的图生成策略以及快速的图划分技术, 实现对海量数据的精准、快速聚类。此外, 本文进一步将 SRC 算法作为基础聚类器, 构建了一个强大的集成聚类算法——稀疏表示集成聚类 SREC 算法, 以显著加速对大规模数据的处理。

2.1　稀疏表示聚类 SRC 算法

SRC 算法流程如图 1 所示。可分为基于随机

样本划分 RSP 的代表点选取、基于 FAISS[30-31] 向量索引库的快速近邻搜索与稀疏图构建以及基于转移切割 Tcut 的图划分 3 个主要模块。

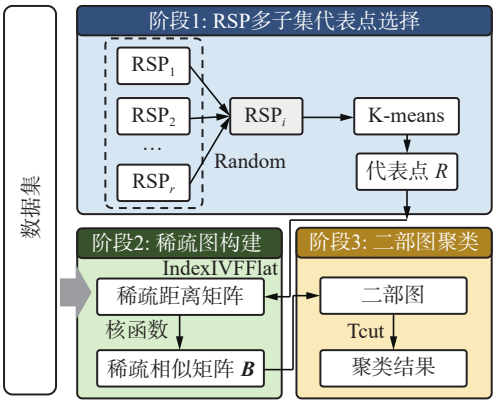


图 1　SRC 算法框架

Fig. 1　SRC algorithm framework

首先，为了从大规模数据集中高效选取既能反映全局结构又具有代表性和覆盖性的代表点，算法采用 RSP 的策略[11-12] 进行代表点的选择。具体而言，将整个数据集 $\boldsymbol{X}\in\mathbf{R}^{N\times d}$ 通过 RSP 方法随机且均匀地划分成 r 个小规模子集 $\{\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_r\}$，每个子集都尽量保留原始数据的统计分布特征。这样的数据划分方式不仅有助于并行处理，降低后续计算复杂度，而且能够在缩小问题规模的同时保持对原始数据集结构的描述。

随后，从这些子集中随机抽取一个样本量小于总数据量 N 且大于目标代表点数 p 的子集 $\boldsymbol{X}_i,\forall i\in\{1,2,\cdots,r\}$，并在该子集上运行 p 个簇的 K-means 聚类算法。将得到的 p 个簇的簇中心作为最终的代表点集合 $\boldsymbol{R}=\{\text{rep}_1,\text{rep}_2,\cdots,\text{rep}_p\}$。该混合策略结合了随机抽样的效率和 K-means 聚类的代表性，一方面避免了在全量数据上执行计算复杂度高的聚类计算，另一方面通过对足够规模的随机候选集进行精炼，使得选出的代表点在质量和代表性方面显著优于简单随机采样。在获得代表点集 $\boldsymbol{R}$ 之后，下一步需要构建一个稀疏相似度矩阵 $\boldsymbol{B}\in\mathbf{R}^{N\times p}$ 来刻画原始数据点集 $\boldsymbol{X}$ 与代表点集 $\boldsymbol{R}$ 之间的关系。直接对每个数据点与所有 p 个代表点计算距离并寻找 K 个最近邻，其计算复杂度为 $O(Npd)$，在数据量 N 极大时，该计算资源消耗过大。因此，算法借助 FAISS 库[30-31] 中的倒排文件索引（IndexIVF）技术来进行高效的近似最近邻搜索。具体地，对代表点集 $\boldsymbol{R}$ 构建 IndexIVF 索引，将其预先划分到多个簇或倒排单元中。在查询阶段，对于原始数据集中的每个点 $x_j\in\boldsymbol{X},\forall j\in\{1,2,\cdots,N\}$，FAISS 首先确定其落入的一个或几个倒排单元，然后仅在这些单元内进行局部精确搜索，以高效地找到 x_j 的 K 个最近邻代表点。接着，利用高斯核函数根据每个数据点 x_j 与其最近邻代表点的距离来构建相似矩阵 $\boldsymbol{B}$。具体而言，如果 $\text{rep}_i\in N_K(x_j)$，即 rep_i 是 x_j 的 K 近邻之一，则

$$\boldsymbol{B}_{ji}=\exp\left(-\frac{\|x_j-\text{rep}_i\|^2}{2\sigma^2}\right)$$

否则 $\boldsymbol{B}_{ji}=0$。通过这种方式，矩阵 $\boldsymbol{B}$ 仅保留每个数据点与其 K 个近邻代表点之间的非零关联，共计 $N\times K$ 个非零元素，显著降低了存储需求和后续计算开销。

最终，将稀疏相似矩阵 $\boldsymbol{B}$ 视作一个连接原始数据点集 $\boldsymbol{X}$ 和代表点集 $\boldsymbol{R}$ 的二部图 $\boldsymbol{G}=\{\boldsymbol{X},\boldsymbol{R},\boldsymbol{B}\}$。为了对该二部图进行有效划分并获得最终聚类结果，本文采用了转移切割方法[32]。转移切割的核心思想是将包含 $N+p$ 个节点的原始二部图 $\boldsymbol{G}$ 的谱划分任务等效地转移到仅包含代表点的较小图上进行。具体来说，首先构建代表点子图 $\boldsymbol{G}_{\mathcal{R}}=\boldsymbol{R},\boldsymbol{E}_{\mathcal{R}}$，其中相似矩阵 $\boldsymbol{E}_{\mathcal{R}}=\boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}_X^{-1}\boldsymbol{B}$，$\boldsymbol{D}_X$ 是由矩阵 $\boldsymbol{B}$ 的行和构成的对角矩阵；然后在该子图上求解广义特征值问题 $\boldsymbol{L}_{\mathcal{R}}\boldsymbol{v}=\lambda\boldsymbol{D}_{\mathcal{R}}\boldsymbol{v}$，这里 $\boldsymbol{L}_{\mathcal{R}}$ 是 $\boldsymbol{G}_{\mathcal{R}}$ 的拉普拉斯矩阵，以获得代表点上的特征向量 $\boldsymbol{v}$；接下来，根据特征向量 $\boldsymbol{v}$ 和特征值 λ，通过特定的变换关系将特征扩展到原始二部图 $\boldsymbol{G}$，从而得到原始数据点的特征向量 $\boldsymbol{u}$，将这些特征向量对应于前 k_{final} 个特征排序后堆叠在一起，构成新的低维表示；最后，对该表示应用 K-means 聚类，即可得到最终的聚类标签。该转移切割策略避免了对尺寸为 $(N+p)\times(N+p)$ 的大规模矩阵进行谱分解，而只需对规模为 $p\times p$ 的矩阵求解特征值问题，其计算复杂度主要受 $O(p^3)$ 支配，与原始数据规模 N 无关，从而大幅提升了谱聚类在大规模数据上的可扩展性。

综上所述，通过上述代表点选择、稀疏相似构建以及基于转移切割的图划分 3 个主要模块，本研究期待 SRC 算法能够在确保聚类质量的同时，实现对大规模数据集的高效谱聚类，后续实验将对其性能进行详细验证，该算法的详细设计流程参见算法 1。

算法 1　SRC 算法

输入　$\boldsymbol{X},k,p,K,r$

输出　Y_{pre}

1) $N,d=\text{size}(\boldsymbol{X})$;

// 阶段一：RSP 多子集代表点选择

2) $\boldsymbol{X}_i=\text{random}[\text{RandomSamplePartition}(\boldsymbol{X},r)]$; // 从 r 个子集中随机选择一个

3) $\boldsymbol{R}=\text{K-means}(\boldsymbol{X}_i,p)$; //对子集进行 K-means 聚类获得代表点

// 阶段二: 稀疏图构建

4) IVFIndex = IndexIVF($\boldsymbol{R}$); //利用 FAISS 库函数计算距离矩阵

5) distances,indices=IVFIndex.search($\boldsymbol{X},K$); //查找所有样本最近的 K 个样本

6) $\boldsymbol{B}$ = GaussianKernel(distances);// 计算稀疏相似矩阵

// 阶段三: 二部图聚类

7) Y_{pre} = Tcut($\boldsymbol{B},k$); //划分相似矩阵获得聚类结果

8) 返回: Y_{pre}

2.2　稀疏表示集成聚类 SREC 算法

SRC 算法的设计初衷旨在兼顾大规模数据聚类中的可扩展性与聚类质量。然而, 单一聚类模型可能因初始化的随机性导致结果不稳定。为提升算法在处理大规模数据聚类问题时的鲁棒性与准确性, 本文在 SRC 算法基础上提出了一种分布式加权集成聚类算法——稀疏表示驱动的集成聚类 SREC 算法。

SREC 算法的设计目标是同时突破传统集成聚类的两大瓶颈: 1)基础聚类器能力有限: 现有方法普遍依赖 K-means 等结构简单的聚类器, 难以有效捕捉复杂数据的潜在子空间结构, 导致生成的基础划分质量不高; 2)分布式计算与通信瓶颈: 传统共识函数通常依赖循环迭代优化(如基于 K-means 的共识), 这在大规模分布式环境中会导致频繁的数据交换与高昂的通信开销, 难以支撑百万级乃至更大规模的数据处理。为此, SREC 算法引入 SRC 作为高质量基础聚类器, 并在 Spark 框架下采用了一种非迭代的加权共识函数来解决上述问题。

SREC 算法遵循分而治之再融合的思想, 整体分为两个阶段, 如算法 2 所示。

算法 2　SREC 算法

输入　$\boldsymbol{X},k,p,K,m,r,k_{\min},k_{\max}$

输出　Y_{pre}

// 阶段一: 分布式基础划分生成(Map 阶段)

1) for i = 1 to m in parallel do

2) 随机生成聚类数量$k_i \in [k_{\min},k_{\max}]$;

3) 生成一个基聚类$\pi_i \leftarrow \text{SRC}(\boldsymbol{X},k_i,p,K,r)$;

4) $\boldsymbol{\Pi} \leftarrow \pi_i$;

5) end for

6) 从工作节点中汇总基聚类$\boldsymbol{\Pi} = \{\pi_1,\pi_2,\cdots,\pi_m\}$;

//阶段二: 加权共识函数与最终聚类(Reduce 阶段)

7) 初始化基聚类质量向量$\boldsymbol{Q} \in \mathbf{R}^m$;

8) for j = 1 to m do

9) for $l = j+1$ to m do

10) nmi $\leftarrow$ NormalizedMutualInfo(π_j,π_l);// 计算基聚类之间的归一化互信息

11) $\boldsymbol{Q}[j] \leftarrow \boldsymbol{Q}[j]$ + nmi;

12) $\boldsymbol{Q}[l] \leftarrow \boldsymbol{Q}[l]$ + nmi;

13) end for

14) end for

//将质量得分转化为权重, 增强高分聚类的影响力

15) $w \leftarrow \text{Softmax}(\boldsymbol{Q})$;

//为$\boldsymbol{\Pi}$中的所有基簇创建唯一的标签。

16) 构建加权对象簇二部图$\tilde{\boldsymbol{B}} \in \mathbf{R}^{N\times k_{\text{cluster}}}$;

17) if $x_j \in C_l$ and $C_l \in \pi_i$ do

18) $\tilde{\boldsymbol{B}}_{jl} = w_i$;

19) else

20) $\tilde{\boldsymbol{B}}_{jl} = 0$;

21) end if

22) Y_{pre} = Tcut($\tilde{\boldsymbol{B}},k$);

23) 返回: Y_{pre}

阶段 1: 分布式基础划分生成(Map 阶段)。此阶段在分布式计算集群上并行执行。算法执行m个独立的 SRC 任务, 每个任务在s个工作节点的 Spark 集群上并行, 以生成一个包含m个基础划分的集成$\boldsymbol{\Pi} = \{\pi_1,\pi_2,\cdots,\pi_m\}$。为了确保集成所必需的多样性, 本文通过以下两种方式主动注入多样性:

1)代表点集多样性: 在每次 SRC 算法执行中, 都通过混合策略独立地重新选择一组代表点$\boldsymbol{R}$。由于该选择过程包含随机预采样, 每次生成的代表点集都会有所不同, 从而使每个基础聚类器从不同的视角观察数据。

2)聚类数量多样性: 每次运行 SRC 算法时, 算法从一个预设的范围$[k_{\min},k_{\max}]$内随机选择一个聚类簇数k_i, 使基础划分能覆盖不同粒度的数据结构。

最终, 所有工作节点将生成的m个基础聚类结果将被汇总至主节点, 进行下一阶段处理。

阶段 2: 加权共识函数与转移切割(Reduce 阶段)。为了突破传统集成方法在计算效率与鲁棒性上的瓶颈, 本文提出了一种基于二部图的非迭代加权图谱共识策略。具体而言, 算法构建了连接数据样本与基簇的二部图, 并将基聚类质量评价分数直接映射为图的边权重, 从而显式保留了基划分的可靠性信息。SREC 算法利用 Tcut 技术对图分割进行建模, 将原本复杂的优化目标转化为一个标准的广义特征值问题。

这一数学转化赋予了算法显著的非迭代特性与计算优势。不同于传统共识算法依赖反复的参数迭代来逼近局部最优，SREC 算法能够通过单次线性代数运算直接获得全局最优的闭式解。这种确定性的求解方式不仅从根本上消除了因初始化敏感导致的收敛不稳定性，更在分布式计算环境中彻底规避了因频繁状态同步而带来的高昂通信开销，实现了在大规模数据上的高效、鲁棒共识。具体步骤如下：

1）评估基聚类质量并计算权重：本研究认为，并非所有基础划分都同等重要。通过计算每个基聚类$\pi_i, \forall i \in \{1,2,\cdots,m\}$与所有其他基聚类的一致性来评估其质量。一致性越高的基聚类，被认为越可靠，从而被赋予更高的权重w_i。

2）构建加权对象簇二部图：将所有m个基础划分中的全部基簇汇集起来，然后构建一个连接N个数据对象和所有基簇的二部图，并用稀疏矩阵$\tilde{\boldsymbol{B}} \in \mathbf{R}^{N\times k_c}$表示其相似度，其中

$$\tilde{\boldsymbol{B}}_{jl} = \begin{cases} w_i, & \text{若对象} x_j \in C_l \wedge C_l \in \pi_i \\ 0, & \text{否则} \end{cases}$$

式中$k_{\text{cluster}} = \sum_{i=1}^{m} k_i$表示集成中所有基簇的总数。

3）转移切割：在二部图$\tilde{\boldsymbol{B}}$上直接应用 Tcut 方法，对样本节点进行谱分解，并最终将N个对象划分为固定簇数k。

其中，式设计核心在于将基聚类的质量信息w_i显式编码进二部图的邻接矩阵中。不同于传统的二值连接方式，将w_i映射为对象与基簇之间的加权连接强度，从数学上为高可靠性样本分配了更大的图切割代价。该约束机制使谱聚类在求解最优划分时能够优先遵循高质量基划分的结构信息，从而在抑制噪声干扰的同时提升共识结果的鲁棒性。

通过这种分布式与加权设计，SREC 算法不仅利用 SRC 算法提升了基础划分的质量和多样性，更通过非迭代机制显著降低了计算与通信成本。结合基于 Spark 的并行 MapReduce 架构，该算法具备了优异的可扩展性和鲁棒性，能够高效完成从海量数据输入到高质量聚类结果输出的完整流程。

2.3　算法时间复杂度分析

在本节将对 SRC 算法和 SREC 算法的时间复杂度进行分析。设数据集大小为$N\times d$，代表点个数为p，数据集分区数为r，簇个数为k，最近邻数为K，K-means 算法迭代次数为t，基聚类个数为m，Hadoop 工作集群的个数为s。SRC 算法主要包含 3 个阶段：

1）通过 RSP 方法将数据集划分为r个子集，在这个过程中首先将原始数据集切分为多个大数据块，对每个初始数据大块进行随机重排并切分成若干小块；其次，根据各大块中索引将小块合并，形成最终的 RSP 数据块集合。该过程本质上是对数据集进行遍历和随机分配，因此时间复杂度为$O(Nd)$；之后，从r个子集中随机选择一个子集执行 K-means 算法，以生成p个聚类中心作为最终代表点，此步骤的时间复杂度为$O(N/rpdt)$。由于随机样本分区是主导项，总时间复杂度约为$O(N/rpdt)$。

2）首先，使用p个代表点构建一个 FAISS 的 IndexIVF 索引，对代表点本身进行一次内部聚类。由于此操作仅在数量较少的代表点上执行，与数据集总规模N无关，因此其开销在整个算法中通常可以忽略不计；之后，对于N个数据点，利用 FAISS 索引查询其最近的K个代表点。这个过程通过索引筛选候选点，再在候选集中计算精确距离，从而避免了对全部$N\times p$个距离的暴力计算，该过程时间复杂度由$O(Np^{1/2}d)$主导；最后，根据搜索到的K近邻结果，计算$N\times K$个高斯核函数值并填充稀疏矩阵$\boldsymbol{B}$，此步骤需要计算$N\times K$个距离和函数值，复杂度为$O(NKd)$，由于通常$K \ll p^{1/2}$，该复杂度低于近似最近邻搜索的复杂度。此阶段的复杂度主要由近似最近邻搜索主导，总时间复杂度约为$O(Np^{1/2}d)$。

3）利用生成的稀疏二分图$\boldsymbol{B}$来获得最终的聚类结果。首先，利用 Tcut 方法将对$N\times p$二分图的划分问题，等效转化为对一个$p\times p$的小图进行处理。由于$\boldsymbol{B}$是一个包含NK个非零元素的稀疏矩阵，此矩阵乘法操作的时间复杂度为$O(NK^2)$；接着，对构建的$p\times p$子图的拉普拉斯矩阵进行特征分解，求出其前k个特征向量，对一个$p\times p$的稠密矩阵进行特征分解的复杂度为$O(p^3)$；之后，根据小图的特征向量恢复出原始N个数据点的k个特征向量；最后，对$N\times k$的嵌入矩阵运行 K-means 算法，得到最终的聚类标签。特征向量恢复的复杂度为$O(NKk)$，最终的 K-means 步骤复杂度为$O(Nk^2t)$。此阶段的总时间复杂度为$O(NK^2+p^3+NKk+Nk^2t)$。

综合上述分析，SRC 算法的总时间复杂度为$O(N/rpdt)+O(Np^{1/2}d)+O(NK^2+p^3+NKk+Nk^2t)$，在大规模应用的场景下，参数满足$k,K,r,t \ll p \ll N$。在此前提下，复杂度表达式可被显著简化。算法的计算瓶颈由与N相关的最高阶项以及对p敏感

的$O(p^3)$项共同决定。与N呈线性关系的项中，$O(Np^{1/2}d)$和$O(NK^2)$通常占据主导地位。因此，算法的总体时间复杂度可简化为$O(N\cdot\max(p^{1/2}d,K^2)+p^3)$。这一结果表明，SRC 算法成功地将传统谱聚类$O(N^2d+N^3)$的巨大开销降低到一个对大规模数据仍能高效处理的可扩展水平。

SREC 算法包含两个重要阶段。在阶段 1 中，首先，将计算任务被分配到s个工作节点上并行执行，其总耗时取决于单个 SRC 算法的复杂度以及并行化的效率。单次 SRC 算法运行的时间复杂度如上所示，在处理大规模数据集时，N通常是最大的变量，因此$O(Np^{1/2}d)$这一项是决定 SRC 算法性能的关键。在不失一般性的前提下，可将 SRC 复杂度的主导部分视为$O(Np^{1/2}d)$；其次，$m(m\ll N)$个独立的 SRC 算法被并行地在s个工作节点上执行。理想情况下，每个节点承担约m/s个任务。因此，整个阶段的计算时间由完成任务最慢的节点决定，其复杂度为$O(m/sNp^{1/2}d)$，分布式架构将m个基础划分的计算成本有效降低了s倍。

阶段 2 主要在单个主节点上集中执行，包含权重计算和 Tcut 划分两个核心步骤。权重计算需要计$O(m^2)$对基础划分之间的 NMI。每次 NMI 计算需要$O(N)$的时间。因此，该步骤总复杂度为$O(m^2N)$，构建包含$N\times m$个非零元素的稀疏矩阵$\tilde{\boldsymbol{B}}$，其复杂度为$O(Nm)$。切割转移 Tcut 划分的复杂度基本与 SRC 算法中一致，但二部图的大小为$N\times m$，因此，时间复杂度为$O(Nm^2+m^3+NKk+Nk^2t)$。综上所述，SREC 算法的总时间复杂度可简化为$O(m/sNp^{1/2}d)$。

3　实验验证与结果分析

为了全面评估聚类算法的性能，本文在多个真实和仿真数据集上进行了实验验证，将本文所提输出的算法与先进的聚类和集成聚类算法进行比较。所有的算法均在 Intel(R) Xeon(R) Platinum 8168, 48 CPU 核心的 5 节点 Hadoop 集群中运行，使用了 Python 3.6.5 版本的编译环境。

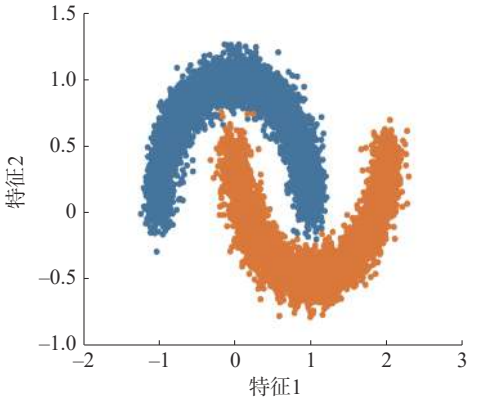
3.1　实验设计

本文实验在 8 个不同规模的数据集上进行，其中包含 4 个真实数据集和 4 个仿真数据集，这些数据集的规模从 7 291 到 5 000 000、维度在 2~784 变化。具体来说，4 个真实数据集为 USPS、Letters[33]、Mnist[34] 和 Covertype[35]；4 个仿真数据集分别为 Moons、Circles、Streaks 和 Flower。实验数据集的详细情况见表 1 和图 2。

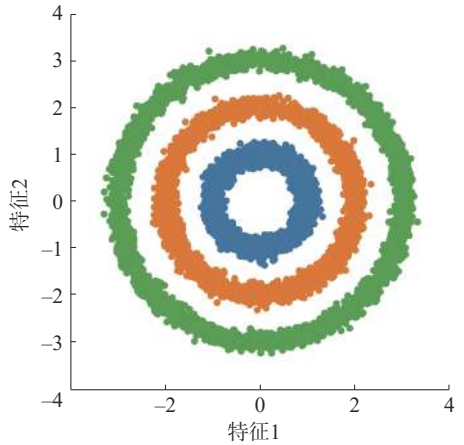
表 1　数据集信息

Table 1　Information of data sets

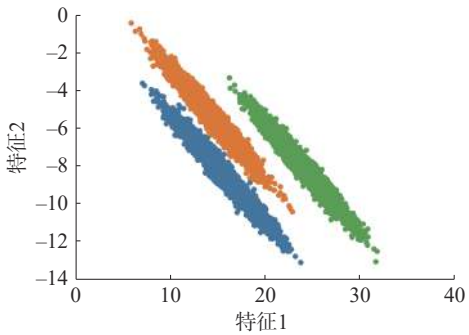
数据类型	数据集	样本量	维度	类
真实数据	USPS	7 291	256	10
	Letters	20 000	16	26
	Mnist	70 000	784	10
	Covertype	581 012	54	7
仿真数据	Moons	800 000	2	2
	Circles	1 000 000	2	3
	Streaks	2 000 000	2	3
	Flower	5 000 000	2	6



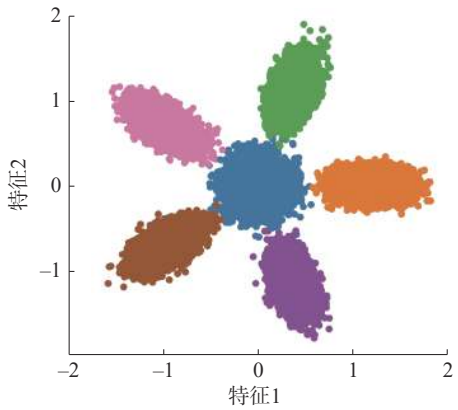
(a) Moons 数据集



(b) Circles 数据集



(c) Streaks 数据集



(d) Flower 数据集

图 2　4 个 1% 数据量的仿真数据集

Fig. 2　Four simulation data sets with 1% data size

为了兼顾算法的大规模扩展能力与实验结果的可复现性，SREC 算法在 Apache Spark 分布式计算框架下进行了标准化部署，并明确了算法的核心步骤。在计算架构上，本文采用分布式基聚类生成和中心化图谱集成的混合策略。在数据划分阶段，利用弹性分布式数据集将原始数据水平切分并负载均衡至集群节点。计算过程严格遵循 Map-Reduce 范式：首先，Map 阶段在各工作节点并行执行基聚类算法，将高维原始数据压缩为紧凑的基聚类标签信息；随后，通过聚合操作（aggregate/collect）将分散的基聚类结果汇聚至主节点，最终的共识阶段由主节点独立完成，其负责构建全局加权二部图并求解基于 Tcut 的广义特征值问题。该架构仅需在网络上传输轻量级的基聚类结果，完全避免了在分布式环境下直接求解大规模图割所需的迭代式通信与全网 Shuffle 操作，从而显著降低了系统开销。

为了评价不同算法之间的聚类结果，实验采用了归一化互信息 NMI、调整兰德指数（adjusted Rand index, ARI）和聚类准确率（accuracy, ACC）3 种广泛使用的聚类评价指标。为了削弱算法训练过程的随机性，在每个实验中，每种聚类算法运行 10 次，最终评估结果取平均值。

NMI 是互信息的归一化结果，用于衡量两个数据分布之间的相关性[8]。它的取值范围通常在 [0, 1]，值越大表示聚类结果 $Y_{\text{pre}}=\{y'_1,y'_2,\cdots,y'_N\}$ 与真实标签 $Y=\{y_1,y_2,\cdots,y_N\}$ 越一致，具体计算公式为

$$L_{\text{MI}}(Y,Y_{\text{pre}})=\sum_{i=1}^{C}\sum_{j=1}^{C'}P(i,j)\log\left(\frac{P(i,j)}{P(i)P(j)}\right)$$

$$L_{\text{NMI}}(Y,Y_{\text{pre}})=\frac{2L_{\text{MI}}(Y,Y_{\text{pre}})}{H(Y)+H(Y_{\text{pre}})}$$

式中：C 和 C' 分别为真实标签和预测标签的集合，共 k 个类别；$P(i)$ 为样本簇为 i 的概率；$P(i,j)$ 为样本同时属于簇 i 和 j 的联合概率；$H(\cdot)$ 为计算信息熵。

ARI 是一种衡量两组聚类结果之间一致性的统计指标，其优势在于对随机分组的可能性进行了调整[36]。ARI 的取值范围通常在 [−1, 1]，其中 1 表示两组划分完全相同，0 表示其相似度等同于随机猜测，而负值则意味着一致性甚至低于随机水平。该算法的核心基于混淆矩阵实现，该矩阵记录了两个聚类划分中各个簇的重叠情况。基于此，ARI 的计算公式为

$$L_{\text{ARI}}=\frac{\sum_{i=1}^{C}\sum_{j=1}^{C'}\binom{n_{ij}}{2}-\left[\sum_{i=1}^{C}\binom{a_i}{2}\sum_{j=1}^{C'}\binom{b_j}{2}\right]\Big/\binom{N}{2}}{\frac{1}{2}\left[\sum_{i=1}^{C}\binom{a_i}{2}+\sum_{j=1}^{C'}\binom{b_j}{2}\right]-\left[\sum_{i=1}^{C}\binom{a_i}{2}\sum_{j=1}^{C'}\binom{b_j}{2}\right]\Big/\binom{N}{2}}$$

式中：a_i 为真实类别 C_i 的样本总数，b_j 为预测簇 C'_j 的样本总数，n_{ij} 为同时属于真实类别 C_i 和预测簇 C'_j 的样本数量。

$$n_{ij}=|\{c|y_c=C_i\wedge y'_c=C'_j\}|$$

ACC 是一个衡量预测聚类标签与真实类别标签一致性的指标，其值在 [0, 1]，越高表示结果越好[37]。由于聚类算法产生的簇标签是任意的，与真实类别标签没有直接对应关系，因此不能直接比较。本文在计算 ACC 值之前使用匈牙利算法进行最佳簇匹配[38]。

$$L_{\text{ACC}}(Y,Y_{\text{pre}})=\frac{1}{N}\max_{\pi}\sum_{j=1}^{c}n_{\pi(j),j}$$

式中：$\max_{\pi}$ 为通过匈牙利算法找到的最优映射所对应的最大匹配数，$\sum_{j=1}^{c}n_{\pi(j),j}$ 为在最优映射下被正确分类的样本总数。

为系统性地验证本文所提出算法的有效性，算法的实验评估分为两个核心阶段，旨在分别对基础算法 SRC 及其集成聚类算法 SREC 进行全面的性能检验。第 1 阶段：评估基础算法 SRC 的聚类性能。在此阶段，本文目标是检验 SRC 算法作为一个独立聚类引擎的竞争力。为此，将其与经典的 K-means 算法以及 4 种先进的谱聚类算法（Nyström[14]、U-SPEC[15]、FSSC[16]、LiteWSC[17]）进行对比。这组对比旨在证明 SRC 算法在处理原始数据时，相较于其他主流的聚类算法，具备同等甚至更优的性能。第 2 阶段：评估 SREC 算法的优越性。本阶段旨在量化所提集成策略带来的性能增益。为此，将 SREC 与 5 种先进的集成聚类算法（EAC[18]、SEC[19]、U-SENC[15]、ECPCS-MC[20]、LiteWSEC[21]）进行对比。此举的目的在于证明，本文的集成算法在融合多个聚类结果、提

升聚类鲁棒性和准确性方面具有强于其他前沿集成技术的优势。

对于上述算法中存在多个共同参数，在实验中按照以下标准设置：

1）所有算法在各个实验上均重复运行了 10 次，以消除算法偶然性所带来的影响。

2）Nyström、U-SPEC、LiteWSC、FSSC 和 SRC 算法均使用代表点的方式来进行矩阵的稀疏表示，在实验中这些方法均设置代表点 $p = 1\,000$，随着代表点 p 大小的变化，算法性能的差异将在 3.4 章节继续讨论。

3）U-SPEC、LiteWSC、FSSC 和 SRC 算法均使用最近代表点大小 K 来寻找各个样本点周围最近的 K 个代表点，在实验中统一设置最近代表点大小 $K = 5$，K 变化时的性能将在 3.4 章节进一步评估。

在集成聚类算法中，每次运行时基础聚类算法随机初始化初始聚类中心，并在 $[k, \min(\sqrt{N}, 50)]$ 范围内随机选择每个基聚类中的聚类数量，其中 k 为类的数量，构建 $m = 20$ 个基聚类的集合。

3.2　合理性分析

SRC 算法的成功在很大程度上依赖于其高效且高质量的代表点选择机制。一个优秀的代表点集应能以较低的计算成本准确地捕捉原始数据集的内在结构和分布特性。为阐明所提出的基于随机样本划分 RSP 与 K-means 结合的混合策略的合理性与优越性，本小节设计了一项消融实验。在该实验中，本文固定 SRC 算法的后续流程，仅替换其核心的代表点选择模块，并比较 3 种不同策略的性能。

1）随机选择（Random）：作为最简单的基线，该方法从整个数据集中完全随机地选择 p 个点作为代表点。

2）全局 K-means：该方法在整个数据集上运行 K-means 算法，将其生成的 p 个簇中心作为代表点。这通常被认为是能获得高质量代表点的标准方法，但计算成本高昂。

3）SRC：基于随机样本划分的多子集代表点选择，即本文提出的混合策略。

本文在真实数据集 USPS 和 Mnist 上对这 3 种代表点选择策略进行了评估，旨在深入分析其在计算效率和聚类精度之间的权衡关系。为了便于可视化展示，本文利用了 t-SNE 方法将高维的 USPS 数据集降维至两维。

图 3 给出了代表点选择策略在 USPS 和 Mnist 数据集上的可视化结果及性能指标。可视化结果表明，Random 方法产生的簇结构混乱且重叠严重。相比之下，SRC 表现出与 K-means 相近的优异聚类结构，各簇之间边界清晰、簇内紧致度高。

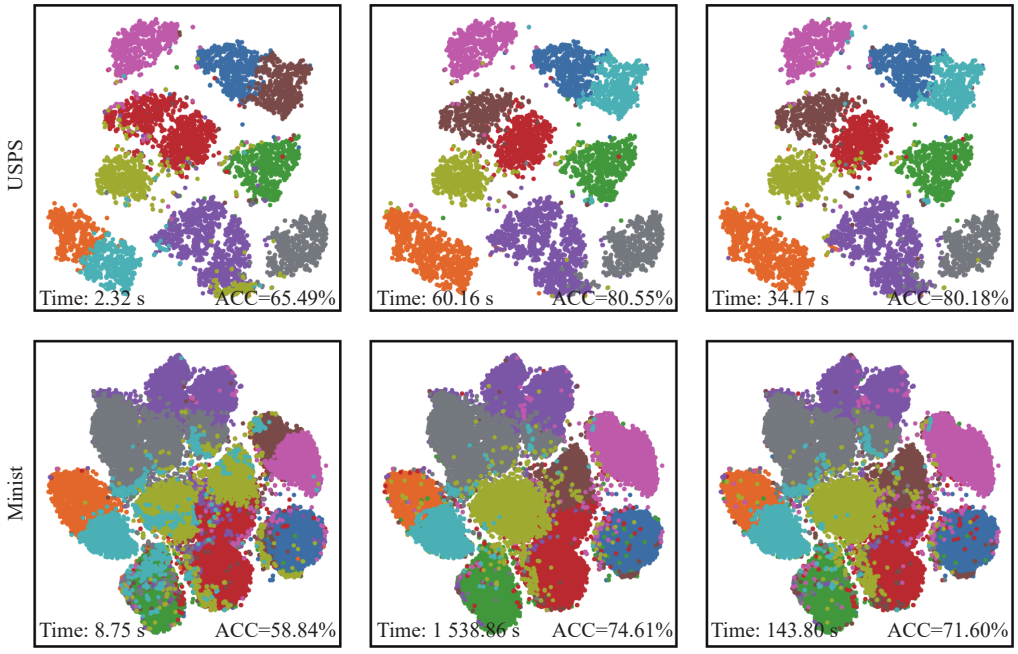


图 3　不同代表点选择方法在 USPS 和 Mnist 数据集上的聚类效果对比

Fig. 3　Comparison of clustering results corresponding to different representative point selection methods on USPS and Mnist datasets

实验的量化结果清晰地展示了不同策略间的性能差异。在 USPS 数据集上，实验结果揭示了计算效率与聚类精度之间典型的权衡现象。随机选择策略虽然速度最快，但其聚类精度 ACC 仅

为 65.49%, 这表明随机选取的点无法充分代表数据中复杂的数字笔划结构。与之相对, 全局 K-means 策略通过迭代优化, 获得了高质量的代表点, 使最终的 ACC 达到了 80.55% 的卓越水平, 但其代价是高达 60.16 s 的计算时间。在此背景下, SRC 策略有效缓解了上述矛盾, 在保证计算效率的同时提升了聚类精度。它取得了 80.18% 的 ACC, 与计算成本高昂的全局 K-means 策略的性能几乎无差, 但其运行时间仅为 34.17 s, 相比全局 K-means 效率提升了约 43%。这有力地证明了本文的混合策略成功地在大幅降低计算成本的同时, 保持了近乎最优的代表点质量。

在数据规模更大的 Mnist 数据集上, SRC 策略在平衡效率与精度方面的优势愈发显著, 充分验证了其在处理大规模数据时优越的可扩展性。全局 K-means 策略虽然获得了最高的 ACC, 但其运行时间激增至 1538.86 s, 对于更大规模的数据集, 该方法将变得不切实际。而 SRC 策略不仅获得了几乎同等的聚类精度, 其运行时间仅为 143.80 s, 相比全局 K-means 实现了超过 10 倍的惊人加速。这一结果证明, 本文的基于 RSP 的代表点选择策略是专为大规模数据设计的, 它成功规避了数据高维特性引发的维度灾难, 同时有效克服了海量样本在直接运行 K-means 时面临的计算瓶颈。

3.3　算法有效性验证

为了全面评估本文提出的 SRC 和 SREC 算法的性能, 本小节将从聚类有效性、计算效率和可扩展性等多个维度进行综合验证。将 SRC 和 SREC 算法与当前主流的、先进的聚类和集成聚类算法在多个真实及大规模仿真数据集上进行细致的比较。由于部分对比算法无法处理大规模数据集, 因此使用 N/A 来表示无法计算的结果。

表 2~4 详细给出了 SRC 算法与其他 5 种聚类算法在所有数据集上的 NMI、ARI 和 ACC 结果。在 USPS、Letters 和 Mnist 这 3 个高维真实数据集上, 本文提出的 SRC 算法在所有 3 个指标上均取得了与当前最先进的 U-SPEC 算法相当甚至略优的性能。例如, 在 Mnist 数据集上, SRC 算法的 NMI 达到了 71.19%, 超越了 U-SPEC 算法的 69.70%。更值得注意的是, SRC 算法在这些数据集上的标准差普遍小于 U-SPEC 算法, 这表明本文算法具有更高的稳定性。相比之下, K-means 和 Nyström 算法等传统方法在处理这些复杂数据时表现不佳, 而 LiteWSC 和 FSSC 算法的性能也与 SRC 算法有明显差距。

表 2　各类聚类算法的 NMI 性能评估

Table 2　NMI performance evaluation of different clustering algorithms　　%

类型	算法	USPS	Letters	Mnist	Covertype	Moons	Circles	Streaks	Flower
聚类	K-means	62.90±0.07	35.60±0.47	49.69±0.56	7.40±5.1×10^{-4}	19.22±0.03	1.36×10^{-4}±8.74×10^{-5}	55.66±0.02	99.81±3.5×10^{-15}
	Nyström	10.28±8.04	34.21±11.49	0.03±1.1×10^{-4}	0.00±0.00	17.04±0.06	1.19×10^{-3}±1.55×10^{-3}	33.66±19.85	96.19±5.23
	U-SPEC	84.66±0.44	45.74±0.78	69.70±0.53	8.50±0.12	98.66±0.16	99.998±9.41×10^{-4}	**89.26±13.03**	99.84±0.02
	LiteWSC	74.58±3.65	42.42±1.10	61.54±3.33	8.13±0.46	6.67±8.58	63.11±12.10	51.91±9.66	89.12±5.86
	FSSC	69.13±4.50	40.38±1.38	59.24±4.07	**9.56±1.29**	54.42±27.90	22.85±4.70	65.51±20.33	N/A
	SRC	**84.92±0.21**	**45.83±1.04**	**71.19±0.32**	8.50±0.20	**98.81±0.14**	**99.999±5.2×10^{-4}**	86.67±13.3	**99.86±0.01**
集成聚类	EAC	72.27±1.37	37.19±0.29	62.04±1.81	N/A	N/A	N/A	N/A	N/A
	SEC	53.53±5.54	31.75±1.13	46.57±2.5	5.87±1.62	26.75±7.21	10.12±7.10	47.57±12.71	85.17±7.34
	ECPCS-MC	74.91±1.48	38.27±0.32	61.24±1.50	7.85±0.17	87.16±15.41	29.49±22.56	64.53±1.07	99.87±1.3×10^{-3}
	U-SENC	85.52±0.88	**47.17±0.65**	74.85±1.57	9.20±0.62	75.29±24.56	99.998±3.8×10^{-4}	99.79±0.26	99.876±2.3×10^{-3}
	LiteWSEC	80.3±2.06	42.7±1.91	67.77±2.55	7.97±1.98	59.52±41.52	94.21±11.65	80.47±15.37	92.72±5.57
	SREC	**86.11±0.23**	47.07±0.59	**76.32±1.85**	**9.92±2.06**	**99.17±0.01**	**99.999±5.7×10^{-4}**	**99.94±0.04**	**99.879±1.2×10^{-3}**

注: 加粗表示各类型算法本列最优结果。

表 3　各类聚类算法在标准数据集上的 ARI 性能评估

Table 3　ARI performance evaluation of different clustering algorithms on standard datasets　　%

类型	算法	USPS	Letters	Mnist	Covertype	Moons	Circles	Streaks	Flower
聚类	K-means	55.29±0.09	12.97±0.36	36.58±0.08	−0.45±2.76×10^{-3}	25.43±0.03	-5.0×10^{-5}±9.6×10^{-5}	53.66±0.03	99.91±1.1×10^{-14}
	Nyström	3.86±4.49	6.48±2.34	-4.3×10^{-6}±5.8×10^{-5}	1.5×10^{-14}±0.00	22.67±0.08	1.1×10^{-3}±1.7×10^{-3}	29.48±17.94	92.9±10.44
	U-SPEC	77.3±0.41	**18.93±1.17**	59.43±1.97	0.31±0.11	99.52±0.06	99.999 4±2.3×10^{-4}	**82.83±20.99**	99.93±0.01
	LiteWSC	61.43±5.81	18.10±1.09	48.98±4.93	1.78±0.47	4.7±5.41	54.09±15.61	44.68±8.79	78.69±12.80
	FSSC	54.79±8.38	17.50±1.88	44.79±6.32	**5.62±2.72**	54.61±30.16	8.61±2.25	57.04±26.62	N/A
	SRC	**77.49±0.23**	17.99±0.87	**59.76±0.56**	0.35±0.16	**99.58±0.06**	**99.999 8±1.4×10^{-4}**	78.57±21.42	**99.94±3.6×10^{-3}**
集成聚类	EAC	62.31±2.08	13.36±0.77	50.66±3.43	N/A	N/A	N/A	N/A	N/A
	SEC	27.65±9.76	8.26±0.99	22.95±5.48	−0.41±2.13	24.14±5.66	5.58±4.29	32.69±10.26	69.91±14.29
	ECPCS-MC	66.25±2.49	13.1±0.43	48.42±1.62	0.31±0.25	89.08±14.43	26.07±19.63	60.78±1.64	99.94±7.4×10^{-4}
	U-SENC	77.59±0.85	**19.30±0.86**	65.63±2.77	1.54±0.46	74.66±26.37	99.999 7±9×10^{-5}	99.91±0.13	99.946±1.2×10^{-3}
	LiteWSEC	70.98±5.36	17.77±0.95	56.19±4.08	**2.03±0.91**	59.19±42.49	91.53±17.18	77.54±17.83	84.68±11.47
	SREC	**78.08±0.29**	18.78±0.74	**66.80±3.08**	1.87±1.28	**99.72±3.3×10^{-3}**	**99.999 9±1.4×10^{-4}**	**99.98±0.01**	**99.947±5.7×10^{-4}**

注: 加粗表示各类型算法本列最优结果。

表 4　各类聚类算法在标准数据集上的 ACC 性能评估

Table 4　ACC performance evaluation of different clustering algorithms on standard datasets　　%

类型	算法	USPS	Letters	Mnist	Covertype	Moons	Circles	Streaks	Flower
聚类	K-means	68.25±0.12	25.96±0.47	53.51±0.39	25.06±6.3×10^{-3}	75.22±0.02	33.40±0.03	78.90±0.02	99.96±0.00
	Nyström	21.79±5.13	22.77±6.43	11.26±2.0×10^{-3}	**48.76±0.00**	73.81±0.04	33.46±0.07	58.06±14.13	93.19±10.25
	U-SPEC	79.85±0.28	**33.94±1.11**	**71.47±1.97**	26.81±1.56	99.88±0.02	99.999 8±7.8×10^{-5}	**86.66±16.33**	99.970±4.3×10^{-3}
	LiteWSC	65.92±6.29	30.8±1.37	62.01±5.36	24.06±1.05	59.25±5.65	73.04±14.68	69.59±6.4	82.1±12.65
	FSSC	63.11±7.01	29.21±1.3	56.85±7.19	31.07±4.50	85.52±10.18	44.06±2.49	72.96±20.43	N/A
	SRC	**80.07±0.24**	33.11±0.88	71.09±0.46	25.99±1.87	**99.90±0.01**	**99.999 9±4.6×10^{-5}**	83.33±16.67	**99.973±1.5×10^{-3}**
集成聚类	EAC	70.29±2.81	25.98±0.84	63.94±4.08	N/A	N/A	N/A	N/A	N/A
	SEC	50.88±6.85	22.78±0.76	44.67±4.20	**28.86±3.58**	74.36±3.22	43.77±5.86	57.25±7.97	76.22±11.03
	ECPCS-MC	73.28±2.88	26.26±0.58	62.5±2.03	27.54±1.39	97.02±4.04	54.02±13.05	82.13±1.13	99.977±3.1×10^{-4}
	U-SENC	**80.72±1.53**	**33.52±1.1**	74.87±1.56	23.6±0.72	92.45±8.03	99.999 91±3×10^{-5}	99.97±0.04	99.977±5×10^{-4}
	LiteWSEC	76.4±5.79	31.79±2.05	68.85±4.22	23.64±1.32	84.50±17.01	94.54±11.89	89.37±9.90	86.91±9.85
	SREC	80.41±0.25	33.28±1.03	**75.25±1.68**	24.07±1.07	**99.93±8.2×10^{-4}**	**99.999 99±4.6×10^{-5}**	**99.99±4.6×10^{-3}**	**99.978±2.4×10^{-4}**

注: 加粗表示各类型算法本列最优结果。

Moons、Circles、Streaks 和 Flower 这 4 个仿真数据集被设计用来测试算法处理非线性、非凸结构数据聚类问题的能力。实验结果清晰地表明，SRC 算法在处理此类任务时表现出显著优越性。在 Circles 数据集上，SRC 算法的 NMI 和 ARI 几乎达到了完美的 100%，而传统的 K-means 算法在此任务上完全失效。这充分证明了算法所采用的谱聚类算法在发现复杂数据结构方面的有效性。从表 2~4 中的标准差数据可以看出，SRC 算法在绝大多数数据集上的结果波动都非常小，显示了其优秀的鲁棒性。相反，Nyström 和 FSSC 等算法在 Streaks 数据集上的标准差非常大，表明它们的聚类结果对随机初始化或数据采样高度敏感，稳定性较差。

表 5 为各类聚类算法在标准数据集上的平均运行时间。在表 5 中，尽管 SRC 算法的运行时间高于 K-means 和 LiteWSC 算法等速度极快但精度较低的算法，但与同样追求高精度的 U-SPEC 和 FSSC 算法相比，SRC 算法在多个数据集上展现出更优或相当的计算效率。在 Mnist 和 Covertype 数据集上，SRC 算法的运行时间均显著少于 U-SPEC 算法。这得益于算法中基于 FAISS 的快速稀疏图生成机制，在保证精度的前提下有效地降低了计算开销。

表 5　各类聚类算法在标准数据集上的平均运行时间

Table 5　Average runtime of different clustering algorithms on standard datasets　　s

类型	数据集	USPS	Letters	Mnist	Covertype	Moons	Circles	Streaks	Flower
聚类	K-means	1.66	4.83	37.87	24.33	3.8	4.94	10.78	19.51
	Nyström	4.26	5.84	10.65	89.36	87.2	113.63	272.58	745.94
	U-SPEC	46.64	27.22	177.91	47.98	34.25	39.09	69.37	170.87
	LiteWSC	0.76	1.31	5.89	26.32	38.06	53.20	149.60	498.28
	FSSC	3.91	5.65	102.23	209.19	215.97	274.82	676.50	N/A
	SRC	26.41	27.18	114.62	78.93	59.62	65.03	112.26	275.33
集成聚类	EAC	73.28	274.84	2925.50	N/A	N/A	N/A	N/A	N/A
	SEC	72.10	182.39	1696.19	3508.70	2596.69	4766.14	13156.36	25784.42
	ECPCS-MC	57.29	151.43	1453.01	3131.95	1351.17	1490.37	2899.87	4932.63
	U-SENC	932.20	554.45	3323.27	903.28	909.21	1022.75	1821.32	3943.56
	LiteWSEC	18.68	21.18	43.71	212.89	279.33	345.42	697.64	2611.13
	SREC	52.53	62.88	279.72	302.78	357.02	422.04	841.91	1900.95

在验证了 SRC 算法作为单个聚类器的有效性后，进一步评估 SREC 算法的性能。对比两组实验结果可以发现，SREC 算法的性能在绝大多数情况下都显著优于其基础聚类器 SRC 以及所有其他的单一聚类算法。在 Mnist 数据集上，SREC 算法的 NMI 为 76.32%，相比 SRC 算法的 71.19% 有了超过 5 个百分点的明显提升。这一结果有力地证明了本文集成算法的有效性，它成功地通过融合多个高质量的基础划分获得了更鲁棒、更精确的共识结果。

在与 EAC、SEC、ECPCS-MC、U-SENC 和 LiteWSEC 等先进集成算法的比较中，SREC 算法在选用的数据集上取得了最佳或接近最佳的性能指标。特别是在大规模仿真数据集上，SREC 算法的优势尤为突出。在 Moons 数据集上，SREC 算法的 NMI 高达 99.17%，显著优于次优算法 U-SENC 的 74.66%。这充分展示了本文提出的基于强基学习器的集成策略相对于传统依赖 K-means 算法的集成算法的优越性。

为了深入评估 SREC 算法的计算效率，本文不仅在数值实验上进行了比较，还从理论层面对算法的时间复杂度进行了详细的对比分析。

首先，传统的 EAC 算法需要计算并存储大小为 $N\times N$ 的全量共关联矩阵，其时间与空间复杂度高达 $O(N^2)$。面对百万级的大规模数据，二次方复杂度导致了不可接受的内存开销。如表 5 所示，EAC 算法在多数数据集上因内存溢出而无法运行。其次，SEC 算法虽然通过加权 K-means 避免了构建全量矩阵，将复杂度降低至 $O(Nmkdt)$，然而，其核心的迭代优化机制导致收敛速度对初始值高度敏感，且在高维空间中迭代次数 t 值往往较大，导致其在 Covertype 和 Streaks 等数据集上耗时较长。相比之下，U-SENC 算法和本文提出的 SREC 算法均成功实现了线性的时间复杂度。U-

SENC 算法利用二部图划分将复杂度优化为 $O(Nmp^{1/2}d+nk^2t)$，但在实验中，仍需在单个节点上串行完成。

相比之下，SREC 算法利用 Spark 框架实现了计算任务的分布式并行处理。在最耗时的基聚类生成阶段将数据划分到 s 个节点并行计算，将理论时间复杂度从单机的 $O(N)$ 降低为分布式的 $O(N/s)$。

此外，SREC 算法在共识阶段引入非迭代图谱划分策略，有效规避了分布式环境下因频繁同步引发的通信延迟，从而克服了单机算法面临的计算与存储资源限制。实验结果显示，在 Flower 数据集上，SREC 的运行时间为 1 900.95 s，显著优于单机最优算法 U-SENC 的 3 943.56 s，验证了分布式并行机制在大规模数据聚类中的高效性与可扩展性。

为了进一步增强对比结果的可信性，采用了在机器学习领域广泛使用的非参数统计检验对表 2~4 显示的实验结果进行了统计学分析。具体而言，本研究首先在 8 个数据集上，基于 10 次独立重复实验的平均 NMI、ARI 和 ACC 结果，计算出各个算法的平均排名。随后，本文采用 Wilcoxon 符号秩检验对算法进行成对比较，并利用 Holm 法对 p 值进行校正，以控制在多重比较下的族系误差率。将统计检验的结果通过临界差异（critical difference，CD）图进行可视化，如图 4~5 所示。

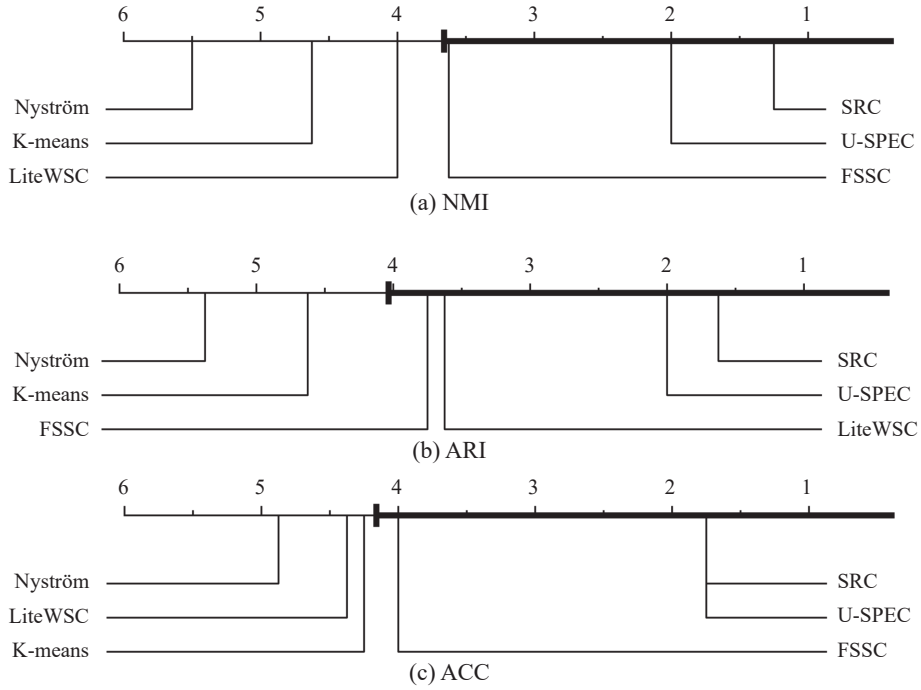


图 4　不同聚类算法 NMI、ARI、ACC 指标的临界差异图

Fig. 4　Critical difference diagrams of NMI, ARI, and ACC metrics corresponding to different clustering algorithms

所有参与比较的算法根据其平均排名被绘制在一条水平轴上，排名越靠前意味着该算法的聚类表现越出色。在本次分析中，设定显著性水平 $\alpha=0.05$，聚类和集成聚类算法参与对比的算法数量各 6 个，数据集数量为 8 个。根据 Bonferroni-Dunn 检验，临界差值的计算公式为

$$L_{\mathrm{CD}}=q_{0.05}\times\sqrt{\frac{k\times(k+1)}{k\times N}}\approx 2.410$$

式中：$k=6$，$N=8$，在 $\alpha=0.05$ 时对应的临界值 $q_{0.05}$ 为 2.576。

图 4 统计分析结果明确证实了 SRC 算法作为一种高性能基础聚类算法的先进性。在 NMI、ARI 和 ACC 这 3 项关键评估指标上，SRC 算法均取得了最优或并列最优的平均排名。根据 CD 图所示，SRC 算法与 U-SPEC 算法共同位于平均排名第一的性能梯队。统计检验结果证实，SRC 算法的性能不仅可以媲美当前最优的 U-SPEC 算法，更显著超越了其余所有对比方法。

在 SRC 算法的稳健基础之上，SREC 算法展现出更为显著的性能优势，进一步体现了整体算法设计的优越性。如图 5 所示，SREC 算法在所有评估指标上的平均排名均居首位，并且其性能在统计学意义上显著优于当前所有参与对比的先进集成聚类算法。实验结果表明，SREC 算法通过融合 SRC 算法所提供的高质量基划分与高效的加权共识机制，有效克服了传统集成聚类方法的性能局限，从而为复杂数据聚类问题提供了更为鲁棒且精确的解决方案。

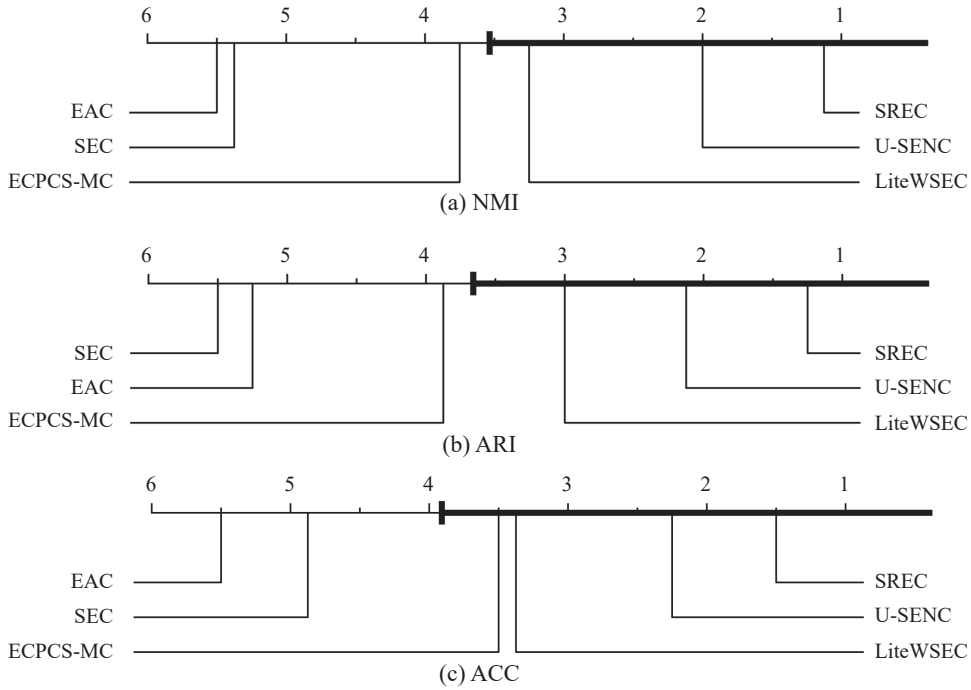


图 5　不同集成聚类算法 NMI、ARI、ACC 指标的临界差异图

Fig. 5　Critical difference diagrams of NMI, ARI, and ACC metrics corresponding to different ensemble clustering algorithms

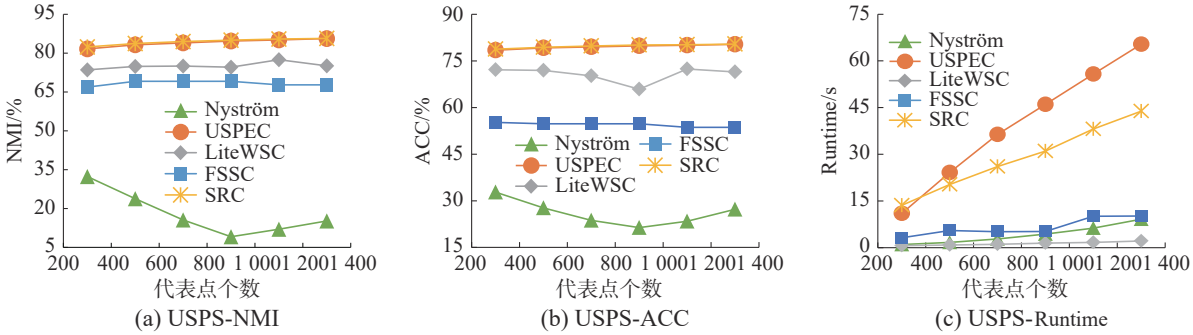
3.4　参数分析

为了验证 SRC 算法和 SREC 算法的稳定性，并为其实际应用提供有价值的调参指导，本节将对算法的 3 个核心参数（即代表点个数p、最近邻代表点个数K以及基聚类个数m）进行系统性的敏感性分析。在理想的情况下，鲁棒的聚类算法应当在合理的取值范围内对参数的变动是不敏感的。

本文在涵盖不同样本规模与空间分布特征的 3 个代表性数据集（USPS、MNIST 和 Moons）上进行实验，当p和K变化时，SRC 算法在 NMI、ACC 和运行时间 Time 上的表现。代表点个数p直接决定了算法对原始数据空间进行近似的精度和后续图划分的计算复杂度。理论上，一个较小的p值可能无法充分捕捉数据的复杂结构，而一个过大的p值则会显著增加计算和存储开销。

图 6 给出了当p从 400 变化至 1 400 时，SRC 及其他对比算法的性能和运行时间变化曲线。在所有 3 个数据集上，SRC 算法的 NMI 和 ACC 指标随着p的增加呈现出先稳步上升后趋于平稳的态势。在 USPS 数据集上，当p从 400 增加到 1 000 时，SRC 的 NMI 从 82.39% 提升至 85.02%。然而，当p继续增大至 1 400 时，性能增益已趋于微弱，表明模型性能在此参数下已趋于收敛。

这一现象表明，当代表点数量达到一定阈值，例如p = 1 000后，其对数据结构的描述能力已趋于饱和，进一步增加代表点数量对精度的提升效果甚微。与性能曲线的平稳不同，SRC 算法的运行时间随着p的增加而稳定增长。在 Mnist 数据集上，当p从 400 增长到 1400 时，运行时间从 61.7 s 增加到 187.5 s。这完全符合算法时间复杂度分析，与其他算法相比，SRC 算法展现了显著优越的性能；Nyström 算法的性能在p增加时甚至出现了下降或剧烈波动，表现出极大的不稳定性；而 U-SPEC 算法虽然性能也随p增加而提升，但在 Mnist 等更复杂的数据集上，其性能始终被 SRC 算法超越。



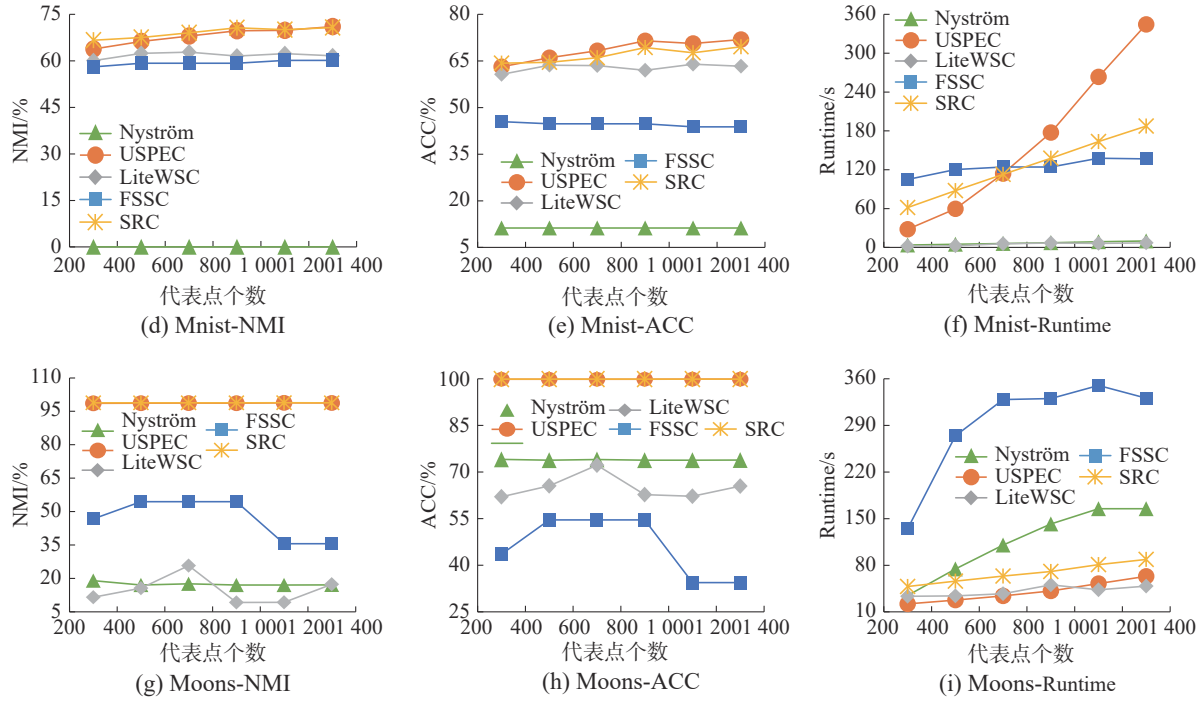


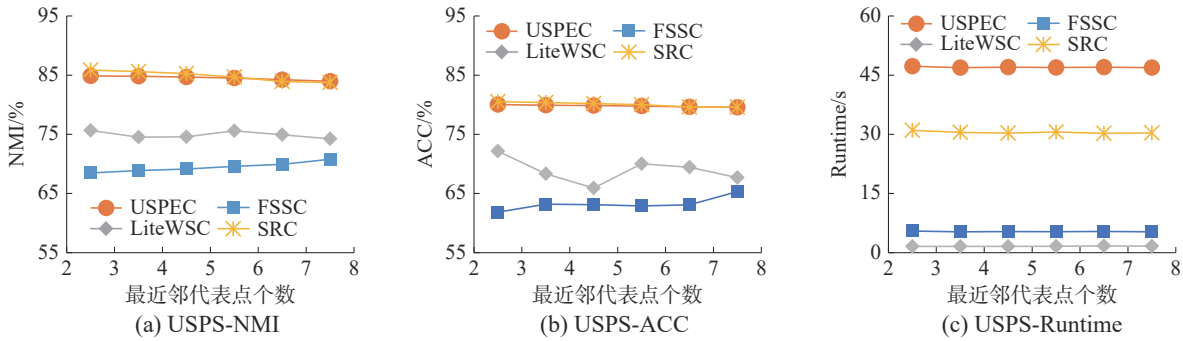
图 6　代表点个数 p 对聚类性能与运行时间的影响分析

Fig. 6　Impact of representative point number p on clustering performance and runtime

通过观察聚类精度与运行时间上的实验结果，可以发现 SRC 算法对参数 p 的选择具有良好的鲁棒性。对于不同的数据集，存在一个最佳的参数 p 取值区间。在实验中，选择代表点数量 $p = 1\,000$，可以在保证接近最高聚类精度的同时，将计算成本控制在合理水平，这一特性增强了 SRC 在实际应用中的易用性。

最近邻代表点个数 K 控制着稀疏二部图的构建，直接影响着每个数据点的局部邻域结构。一个过小的 K 可能导致图的连通性不足，无法捕捉完整的簇结构；而一个过大的 K 则可能引入噪声连接到不相关的代表点，从而模糊簇与簇之间的边界。图 7 给出了当 K 从 3 变化至 8 时，SRC 算法的性能与时间变化。当 K 继续增加时，所有数据集上的性能均出现轻微的下降趋势。这表明一个较小的 K 值足以捕捉关键的局部邻域信息，而更大的 K 值反而会因引入非结构性连接而对聚类产生负面影响。尽管存在最优值，但 SRC 算法的性能在 K 的变化范围内并未出现明显振荡。在 USPS 上，当 K 从 3 增加到 8，ACC 仅从 80.53% 下降到 79.60%。这说明 SRC 算法对参数 K 的变化并不敏感，在最优值附近的一个小邻域内均能取得优异且稳定的结果。从时间曲线可以看出，运行时间在 K 的变化范围内基本保持稳定，没有出现显著增长。由于 K 的取值范围通常不大，其对整体运行时间的影响是温和且可控的。

对参数 K 的分析进一步验证了 SRC 算法的鲁棒性。实验表明，对于不同的数据集，选择一个较小的 K 值通常是一个安全且高效的选择。SRC 算法能够在较宽的 K 值范围内保持高性能，这再次证明了本文算法设计的合理性，使其在面对新任务时，无需进行代价高昂的、精细的参数搜索。



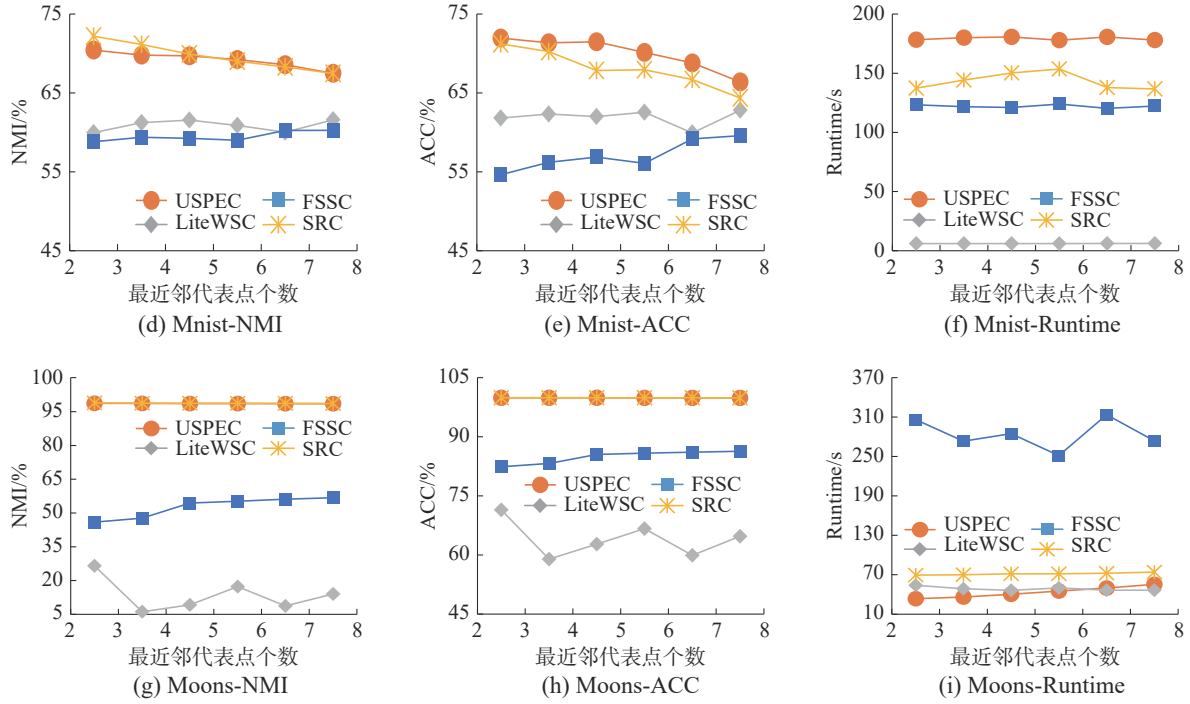


图 7　最近邻代表点个数 *K* 对聚类性能与运行时间的影响

Fig. 7　Impact of nearest neighbor number *K* on clustering performance and runtime

在集成聚类算法中，基聚类个数m是一个关键参数，其直接影响着最终共识结果的质量以及算法的计算开销。理想的集成算法应当在m取较小值时即可快速收敛到高性能，并对m的变化不敏感，同时计算成本应随m的增加而呈现可控的增长。

在 USPS 和 Mnist 两个具有代表性的数据集上，测试了当m在 [5, 10, 15, 20, 25, 30] 范围内变化时，SREC 及其他 5 种先进的集成聚类算法在 NMI、ACC 和 Time 上的表现，如图 8 所示。

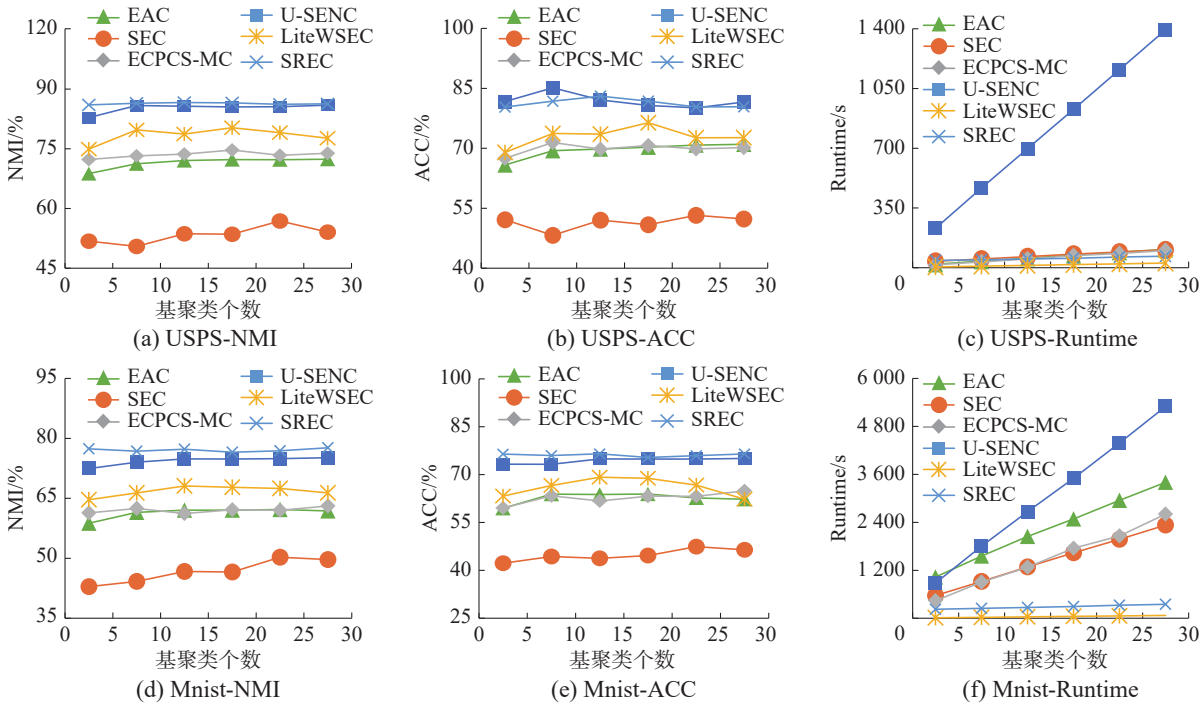


图 8　基聚类个数 *m* 对集成聚类性能与运行时间的影响

Fig. 8　Impact of base cluster number *m* on clustering performance and runtime

在聚类质量方面, SREC 展现了卓越的收敛速度与性能优势。如图 8 所示, 即使在$m=5$的起始点, SREC 的 NMI 和 ACC 指标已全面超越所有对比算法在任意m值下的表现。具体而言, 在 MNIST 数据集上, $m=5$时 SREC 的 NMI 已达 77.38%, 显著优于次优算法 U-SENC。这一优势归因于 SREC 独特的稀疏表示生成策略, 其产生的高质量基划分使得算法仅需少量集成即可迅速收敛至稳健的共识结果。此外, 当m从 10 增加至 30 时, SREC 的性能曲线表现出极高的稳定性,在 USPS 上 NMI 始终稳定在 86% 以上, 未出现明显的波动或退化。这种对参数m的低敏感性意味着 SREC 在实际部署中无需依赖复杂的参数微调, 显著提升了算法的易用性。相比之下, 其他算法表现出明显的局限性, EAC 和 ECPCS-MC 依赖更大的m值来弥补基学习器的质量不足; LiteWSEC 表现出较强的波动性; 而 U-SENC 虽然性能较好, 但快速趋于平缓, 且受限于高昂的计算成本, 无法通过增加m获得进一步的显著提升。

在计算效率方面, 所有集成算法的运行时间均随m增加而增长, 但增长模式差异显著。SREC 的时间成本呈现出线性增长趋势, 这得益于本文设计的基于分布式计算的并行基聚类机制, 确立了其计算复杂度与m的线性关系。与之形成鲜明对比的是, U-SENC 的运行时间随m的增大呈现显著上升趋势。在 MNIST 数据集上, 当m从 5 增至 30 时, U-SENC 的耗时从 883 s 飙升至 5 299 s, 增长近 6 倍, 暴露了其在集成规模扩大时的效率瓶颈。虽然 LiteWSEC 运行速度最快, 但其代价是牺牲了聚类精度与稳定性。

综上所述, SREC 算法在性能增益与计算效率之间实现了最佳平衡。它不仅在聚类精度上显著优于当前集成聚类方法, 更在小样本集成下展现了近乎最优的性能与极佳的鲁棒性。同时, 其计算成本增长曲线赋予了用户在追求高性能与控制预算之间灵活选择的权利。这些特性验证了 SREC 作为一种高效、鲁棒的大规模集成聚类方案的实用价值。

4　结束语

针对现有大规模数据聚类算法面临的结果稳定性不足与分布式部署困难等挑战, 本文提出了基于稀疏表示的 SRC 与 SREC 两种新型聚类算法。其中, SRC 算法通过融合代表点选择、近似邻域搜索与转移切割策略, 有效突破了传统谱聚类在大规模场景下的计算复杂度与存储瓶颈; 在此基础上, SREC 算法进一步将 SRC 作为高鲁棒性的基聚类生成器, 并结合分布式并行架构与加权共识函数, 成功克服了传统集成方法中基学习器质量受限与单机计算扩展性不足的问题。

实验结果表明, SREC 算法在 3 个关键维度上展现了显著优势: 1)参数鲁棒性: 算法对关键超参数基聚类个数m与代表点数量p的变化表现出高度的适应性, 无需依赖精细调优; 2)算法稳定性: 通过集成机制, 有效降低了随机采样与初始化条件带来的结果方差, 保证了聚类输出的一致性; 3)可扩展性: 凭借分布式架构, 具备处理百万级超大规模数据集的能力。广泛的对比实验证实, SRC 算法在聚类精度与运行时间上均优于基础对比算法, 而 SREC 算法在所有测试场景中均确立了显著优势, 充分验证了其在大规模任务中的高效性与结果可靠性。

尽管本文提出的 SRC 和 SREC 算法在处理大规模聚类任务上验证了其有效性, 但在其理论与应用层面仍存在进一步探索和优化的空间。未来的研究可以从以下几个方向展开: 1)可扩展性探索: 可以研究更先进的、具有近线性时间复杂度的随机化算法或分布式算法来优化代表点选择过程。2)面向动态数据的在线与流式聚类: 本文的算法框架主要针对静态数据集设计, 然而, 在许多现实应用中, 数据是以流的形式动态生成的。研究如何高效地增量更新代表点集、稀疏相似度图以及共识聚类结果, 以适应数据分布随时间的动态变化。3)结合先验约束的半监督聚类扩展: 通过引导代表点选择使其偏向已知类别, 或在图划分阶段加入约束项, 以进一步提升聚类的准确性和可解释性。

参考文献:

[1] SASAKI Y. A survey on IoT big data analytic systems: current and future[J]. IEEE internet of things journal, 2022, 9(2): 1024−1036.

[2] 孙吉贵, 刘杰, 赵连宇. 聚类算法研究[J]. 软件学报, 2008, 19(1): 48−61.
SUN Jigui, LIU Jie, ZHAO Lianyu. Clustering algorithms research[J]. Journal of software, 2008, 19(1): 48−61.

[3] SINGH J, SINGH D. A comprehensive review of clustering techniques in artificial intelligence for knowledge discovery: taxonomy, challenges, applications and future prospects[J]. Advanced engineering informatics, 2024, 62: 102799.

[4] 段意强. 多视图子空间聚类算法研究[D]. 广州: 广东工

业大学, 2022: 6−7.
DUAN Yiqiang. Research on multi-view subspace clustering algorithms[D]. Guangzhou: Guangdong University of Technology, 2022: 6−7.

[5] HE Wei, ZHANG Shangzhi, LI Chunguang, et al. Neural normalized cut: a differential and generalizable approach for spectral clustering[J]. Pattern recognition, 2025, 164: 111545.

[6] 林毓秀, 刘慧, 于晓, 等. 面向子空间聚类的多视图统一表示学习网络[J]. 计算机研究与发展, 2025, 62(5): 1248−1261.
LIN Yuxiu, LIU Hui, YU Xiao, et al. A multi-view unified representation learning network for subspace clustering[J]. Journal of computer research and development, 2025, 62(5): 1248−1261.

[7] YANG Yiyang, DENG Sucheng, LU Juan, et al. GraphL-SHC: towards large scale spectral hypergraph clustering [J]. Information sciences, 2021, 544: 117−134.

[8] STREJL A, GHOSH J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions[J]. Journal of machine learning research, 2002, 3(12): 583−617.

[9] ZHOU Peng, LI Rongwen, LING Zhaolong, et al. Fair clustering ensemble with equal cluster capacity[J]. IEEE transactions on pattern analysis and machine intelligence, 2025, 47(3): 1729−1746.

[10] 闫晨, 杨有龙, 刘原园. 基于聚类质量的两阶段集成算法[J]. 吉林大学学报 (理学版), 2023, 61(4): 899−908.
YAN Chen, YANG Youlong, LIU Yuanyuan. Two stage ensemble algorithm based on clustering quality[J]. Journal of Jilin University(science edition), 2023, 61(4): 899−908.

[11] SALLOUM S, HUANG J Z, HE Yulin. Random sample partition: a distributed data model for big data analysis[J]. IEEE transactions on industrial informatics, 2019, 15(11): 5846−5854.

[12] WEI Chenghao, SALLOUM S, EMARA T Z, et al. A two-stage data processing algorithm to generate random sample partitions for big data analysis[C]//Cloud Computing. Seattle: IEEE, 2018.

[13] IKOTUN A M, EZUGWU A E, ABUALIGAH L, et al. K-means clustering algorithms: a comprehensive review, variants analysis, and advances in the era of big data[J]. Information sciences, 2023, 622: 178−210.

[14] FOWLKES C, BELONGIE S, CHUNG F, et al. Spectral grouping using the nystrom method[J]. IEEE transactions on pattern analysis and machine intelligence, 2004, 26(2): 214−225.

[15] HUANG Dong, WANG Changdong, WU Jiansheng, et al. Ultra-scalable spectral clustering and ensemble clustering[J]. IEEE transactions on knowledge and data engineering, 2020, 32(6): 1212−1226.

[16] WANG Jingyu, MA Zhenyu, NIE Feiping, et al. Fast self-supervised clustering with anchor graph[J]. IEEE transactions on neural networks and learning systems, 2022, 33(9): 4199−4212.

[17] YANG Geping, DENG Sucheng, YANG Yiyang, et al. LiteWSC: a lightweight framework for web-scale spectral clustering[C]//Database Systems for Advanced Applications. Online: Springer, 2022.

[18] FRED A L N, JAIN A K. Combining multiple clusterings using evidence accumulation[J]. IEEE transactions on pattern analysis and machine intelligence, 2005, 27(6): 835−850.

[19] LIU Hongfu, WU Junjie, LIU Tongliang, et al. Spectral ensemble clustering via weighted K-means: theoretical and practical evidence[J]. IEEE transactions on knowledge and data engineering, 2017, 29(5): 1129−1143.

[20] HUANG Dong, WANG Changdong, PENG Hongxing, et al. Enhanced ensemble clustering via fast propagation of cluster-wise similarities[J]. IEEE transactions on systems, man, and cybernetics: systems, 2021, 51(1): 508−520.

[21] YANG Geping, DENG Sucheng, CHEN Can, et al. LiteWSEC: a lightweight framework for web-scale spectral ensemble clustering[J]. IEEE transactions on knowledge and data engineering, 2023, 35(10): 10035−10047.

[22] ZHU Xueying, SUN Jie, HE Zhenhao, et al. Stale-ness-reduction mini-batch K-means[J]. IEEE transactions on neural networks and learning systems, 2024, 35(10): 14424−14436.

[23] 李大瑞, 杨林军, 华先胜, 等. 视觉特征空间中大规模聚类问题的一种鲁棒近似算法[J]. 中国科学技术大学学报, 2014, 44(10): 844−852.
LI Darui, YANG Linjun, HUA Xiansheng, et al. A robust approximate algorithm for large-scale clustering of visual features[J]. Journal of University of Science and Technology of China, 2014, 44(10): 844−852.

[24] CHEN Xinlei, CAI Deng. Large scale spectral clustering with landmark-based representation[J]. Proceedings of the AAAI conference on artificial intelligence, 2011, 25(1): 313−318.

[25] 贾洪杰, 丁世飞, 史忠植. 求解大规模谱聚类的近似加权核 K-means 算法[J]. 软件学报, 2015, 26(11): 2836−2846.
JIA Hongjie, DING Shifei, SHI Zhongzhi. Approximate weighted kernel K-means for large-scale spectral clustering[J]. Journal of software, 2015, 26(11): 2836−2846.

[26] YAN Donghui, HUANG Ling, JORDAN M I. Fast ap-

proximate spectral clustering[C]//Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Paris: ACM, 2009.

[27] WU Junjie, LIU Hongfu, XIONG Hui, et al. K-means-based consensus clustering: a unified view[J]. IEEE transactions on knowledge and data engineering, 2015, 27(1): 155−169.

[28] LIU Hongfu, ZHAO Rui, FANG Hongsheng, et al. Entropy-based consensus clustering for patient stratification [J]. Bioinformatics, 2017, 33(17): 2691−2698.

[29] LI Tao, DING C. Weighted consensus clustering[C]//Proceedings of the 2008 SIAM International Conference on Data Mining. Atlanta: SIAM, 2008.

[30] DOUZE M, GUZHVA A, DENG C, et al. The faiss library[EB/OL]. (2024−01−16)[2025−08−20]. https://arxiv.org/abs/2401.08281.

[31] 郭嘉铭. 基于大语言模型的领域问答技术研发[D]. 成都: 电子科技大学, 2025.
GUO Jiaming. Research and development of domain question answering technology based on large language models[D]. Chengdu: School of Computer Science and Engineering, 2025.

[32] LI Zhenguo, WU Xiaoming, CHANG S F. Segmentation using superpixels: a bipartite graph partitioning approach [C]//2012 IEEE Conference on Computer Vision and Pattern Recognition. Providence: IEEE, 2012.

[33] SLATE D. Letter recognition[DB/OL]. UCI machine learning repository. (1990−12−30) [2025−06−25]. https://doi.org/10.24432/C5ZP40.

[34] LECUN Y, CORTES C, BURGES C J C. The MNIST database of handwritten digits[DB/OL]. (2021−10−16) [2025−06−25]. https://doi.org/10.24432/C53K8Q.

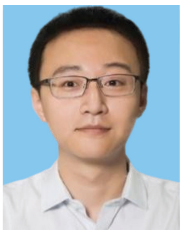
[35] BLACKARD J. Covertype[DB/OL]. UCI Machine Learning Repository. (1998−07−31) [2025−06−25]. https://doi.org/10.24432/C50K5N.

[36] HUBERT L, ARABIE P. Comparing partitions[J]. Journal of classification, 1985, 2(1): 193−218.

[37] NGUYEN N, CARUANA R. Consensus clusterings[C]// Seventh IEEE International Conference on Data Mining. Omaha: IEEE, 2007.

[38] 李婕, 张智雄, 王字飞. 增加类簇级对比的 SCCL 文本深度聚类方法研究[J]. 数据分析与知识发现, 2024, 8(3): 98−109.
LI Jie, ZHANG Zhixiong, WANG Yufei. SCCL text deep clustering with increased cluster-level comparison[J]. Data analysis and knowledge discovery, 2024, 8(3): 98−109.

作者简介:



何玉林, 研究员, 博士, 深圳市特支人才, 主要研究方向为新型大数据理论模型、大数据高性能/并行计算平台、面向大数据的机器学习算法、复杂型大数据应用系统。主持国家级、省部级科研项目 10 余项, 发表学术论文 100 余篇。E-mail: yulinhe@gml.ac.cn。

杨振宇, 硕士研究生, 主要研究方向为大规模数据聚类、集成聚类、机器学习。E-mail: yangzhenyu@gml.ac.cn。

肖又旗, 硕士研究生, 主要研究方向为大数据分布式计算、Spark 能耗优化、高性能数据挖掘。E-mail: xiaoyouqi@gml.ac.cn。