基于自适应融合全局和局部信息的锚点多视图聚类

冉　懿，　王思为，　祝　恩

(国防科技大学 计算机学院　湖南 长沙 410073)

摘要：基于子空间的多视图聚类算法因其良好的聚类性能和数学可解释性而备受关注。其中，一些基于锚点策略的大规模多视图子空间聚类算法，能够有效降低算法的时空复杂度。然而，现有的算法往往从全局结构中学习子空间自表示矩阵，忽视了视图数据、锚点和子空间自表示矩阵之间的局部结构信息。受多视图自加权多图学习算法的启发，提出了基于自适应融合全局和局部信息的锚点多视图聚类(AMVC-AFGL)算法。所提算法旨在通过自适应分配视图权重，融合数据之间的全局信息和局部信息，为每个视图数据学习一个更有效的子空间锚图矩阵，进而拼接为较小的融合锚图矩阵然后进行谱聚类。在公开的 10 个真实基准数据集上开展了充分的实验，结果表明，与其他 12 个先进的多视图聚类算法相比，所提算法具有有效性和可扩展性。

关键词：多视图聚类；自加权；锚点；子空间聚类；谱聚类；大规模

中图分类号：TP301　　**文献标志码**：A　　**文章编号**：1671-6841(2025)04-0030-10

DOI：10. 13705/j. issn. 1671-6841. 2023205

Anchor Multi-view Clustering Based on Adaptive Fusion of Global and Local Information

RAN Zhuang, WANG Siwei, ZHU En

(*School of Computer*, *National University of Defense Technology*, *Changsha* 410073, *China*)

Abstract: Subspace-based multi-view clustering algorithms have attracted much attention due to their good clustering performance and mathematical interpretability. Among them, some large-scale multi-view subspace clustering algorithms based on anchor strategy can effectively reduce the spatiotemporal complexity. However, existing algorithms often learned the subspace self-representation matrix from the global structure, ignoring the local structure information between the view data, anchors and the subspace self-representation matrices. Inspired by the multi-view self-weighted multi-graph learning algorithm, the anchor multi-view clustering based on adaptive fusion of global and local information (AMVC-AFGL) algorithm was proposed. The proposed algorithm aimed to learn a more effective subspace anchor graph matrix for each view data by adaptively allocating view weights and fusing the global information and local information between the data, and then concatenated them into a smaller fusion anchor graph matrix for spectral clustering. Extensive experiments were carried out on 10 public real benchmark datasets, and compared with other 12 advanced multi-view clustering algorithms, the results showed the effectiveness and scalability of the proposed algorithm.

Key words: multi-view clustering; self-weighted; anchor; subspace clustering; spectral clustering; large-scale

收稿日期:2023-08-31
基金项目:国家自然科学基金项目(62276271, 62325604)
第一作者:冉懿(1994—),男,硕士研究生,主要从事多视图聚类研究,E-mail:1294386964@ qq. com。
通信作者:祝恩(1976—),男,教授,主要从事聚类算法、异常检测、计算机视觉、医学图像分析等研究,E-mail: enzhu@ nudt. edu. cn。

0　引言

聚类算法可以探索数据的自然结构和分布以及其他有用的信息。例如,聚类可以用于数据降维和矢量量化[1],可以压缩数据高维特征,常用于压缩图像、声音和视频等非结构化数据,大幅减少此类数据的存储和传输开销。

对于多视图数据,多视图聚类是无监督学习数据分析的关键算法之一。多视图聚类的目的是通过结合多视图数据的一致性和互补性,挖掘其潜在的关联交互信息来提高聚类性能,能够有效揭示多视图数据的内部结构,其关键问题是如何有效融合异构信息[2-3]。现有的多视图聚类算法包括基于非负矩阵分解的算法[4-5]、基于图的算法[6-8]和基于子空间的算法[9-10]等。基于子空间的多视图聚类算法,因其良好的聚类性能和数学可解释性,已成为多视图聚类的主流算法之一。然而,当前一些多视图聚类算法直接对单个视图分别进行自表示学习,忽略了原始数据中存在的噪声和冗余信息,且时空复杂度较高。

近年来,锚点策略在降低多视图子空间聚类算法复杂度的研究中备受瞩目。传统的子空间聚类算法通常是构建所有样本点之间的相似度矩阵,计算开销很大,而基于锚点的子空间聚类通常是学习原始数据和锚点之间的关系矩阵,在很大程度上降低了算法的时空复杂度[11-12]。但是现有的算法往往从全局结构中学习相似度矩阵,忽略了视图数据、锚点和子空间相似度图之间的局部相关性信息。另外在构建子空间相似度图的过程中,有的算法平等对待每个视图所包含的信息,忽视了视图异构性带来的聚类差异。这些问题可能限制算法学习更有效的子空间锚图表示,导致算法有较差的可扩展性和稳定性。

针对上述问题,本文提出基于自适应融合全局和局部信息的锚点多视图聚类算法(AMVC-AFGL),在从原多视图数据中学习锚图的基础上,考虑不同视图的重要性并自适应地为每个视图赋予权重,进一步解决现有算法没有考虑锚图与原数据的局部相关性信息的问题,进而提高算法的聚类性能。通过理论分析和实验验证解决算法优化问题,表明了本算法的有效性与可扩展性。

本文的主要贡献包括 3 个方面:

1) 探索在大规模多视图数据中构造和学习锚图矩阵的优缺点,考虑为子空间锚图矩阵学习自适应分配视图权重系数,为每个视图学习一个更有效的子空间锚图矩阵;

2) 探索子空间锚图矩阵与原视图数据的局部结构信息的内在关系,将其局部结构信息与全局信息进行融合,建立一个统一的优化框架;

3) 在 10 个公开的多视图基准数据集上,对本文算法与其他 12 个先进的多视图聚类算法开展对比实验,分析算法的聚类精度和效率,为后续的研究工作提供新的视角和参考价值。

1　相关工作

1.1　多视图子空间聚类

大数据时代背景下,许多应用都涉及大规模多视图数据,因此多视图子空间聚类(multi-view subspace clustering, MVSC)算法[13-16]的研究备受关注。对于多视图数据 $\{\boldsymbol{X}^v\}_{v=1}^c$,$\boldsymbol{X}^v \in \mathbf{R}^{d_v \times n}$ 是第 v 个视图上特征维度为 d_v 的数据,MVSC 通常求解优化问题,

$$\min_{\boldsymbol{S}^v} \sum_{v=1}^{c} \| \boldsymbol{X}^v - \boldsymbol{X}^v \boldsymbol{S}^v \|_F^2 + \lambda f(\boldsymbol{S}^v),$$
$$\text{s.t. } \boldsymbol{S}^v \geqslant 0, \boldsymbol{S}^{v\mathrm{T}} \mathbf{1} = 1, \tag{1}$$

其中:$f(\cdot)$ 是某一个正则化函数,可根据算法的不同需求进行选择;$\lambda > 0$ 是平衡参数;$\boldsymbol{S} \in \mathbf{R}^{n \times n}$ 是非负的相似度图矩阵;$\mathbf{1}$ 是元素全为 1 的向量。$\boldsymbol{S}^{v\mathrm{T}}\mathbf{1} = 1$ 约束了 $\boldsymbol{S}^v$ 的每列元素之和为 1。

基于式(1)的框架,选择不同的正则化函数 $f(\cdot)$,将会得到不同性质的结果。例如,Gao 等[13]提出的 MVSC 算法学习每个视图的图表示,并假设所有图共享一个唯一的聚类矩阵;Luo 等[14]探索了不同视图的一致性和特异性;Wang 等[3]和 Zhang 等[15]在潜在表示中进行多视图子空间聚类;Brbić 等[10]对图施加了低秩约束和稀疏约束;Kang 等[16]为每个视图学习一个分区,并在分区空间中执行融合。在聚类精度方面,这些算法是有吸引力的。然而,它们的计算复杂度至少为 $O(n^2k)$。因此,较高的时空复杂度限制了它们在大规模多视图数据上的应用。此外,由于多视图数据的异构性,现有的单视图子空间聚类加速技术并不适用于多视图场景。

1.2　基于锚点的多视图子空间聚类

从多视图数据中直接构建维度为 $n \times n$ 相似度图矩阵 $\boldsymbol{S} \in \mathbf{R}^{n \times n}$,会耗费大量的计算时间和存储空间,继而还将应用 $O(n^2k)$ 复杂度的谱聚类算法才能得到聚类结果。近年来,研究者们利用锚点策略构造大尺度图矩阵[11-12],在降低计算复杂度的同时

还能拥有较好的聚类精度。锚点策略的原理是从原始数据中通过启发式采样来选取一部分代表点作为锚点,或者通过优化学习策略来得到锚点,进而构造或者学习一个稀疏的亲和锚图矩阵 $\boldsymbol{Z} \in \mathbf{R}^{n\times m}(m \ll n)$,从而代替直接求解系数矩阵 $\boldsymbol{S} \in \mathbf{R}^{n\times n}$,然后 $\boldsymbol{S}$ 可以通过公式(2)得到,

$$\boldsymbol{S} = \hat{\boldsymbol{Z}}\hat{\boldsymbol{Z}}^{\mathrm{T}}, \tag{2}$$

其中:$\hat{\boldsymbol{Z}} = \boldsymbol{Z}\boldsymbol{\Sigma}^{-\frac{1}{2}}$,$\Sigma_{ij} = \sum_j z_{ji}$。

基于学习策略的锚点获取算法,通常是将锚点的学习和锚图矩阵的构建统一到一个优化框架中,并通过交替优化的方式更新锚点。例如,Sun 等[17]提出的基于统一锚点的可扩展多视图子空间聚类算法,通过优化学习策略得到锚点;Wang 等[18]提出的基于一致锚点的快速无参数多视图子空间聚类算法,将锚点学习和锚图构造结合成一个统一的框架来共同优化。在文献[18]的基础上,Liu 等[19]提出了具有一致锚点的高效一步到位的多视图子空间聚类算法,直接从构建的具有连通性约束锚图中生成聚类标签。

虽然通过优化学习获得锚点的算法有效提升了聚类性能,但在优化过程中引入锚点作为要学习的变量会带来额外的计算开销。近来,基于启发式采样策略获取锚点的方式,因其强大的可扩展性和稳定性而得到广泛应用。例如,Li 等[20]提出的一种可扩展无参数二部图融合的多视图聚类算法,根据样本打分对其采样锚点。文献[12]还对比了 k-means 算法选择锚点和随机采样锚点的不同聚类效果,结果显示 k-means 算法采样策略更优。Kang 等[11]提出一种新的大规模多视图子空间聚类算法,首先利用 k-means 算法得到每个视图的锚点矩阵,然后为每个视图学习一个较小的子空间图矩阵,大大降低了算法的时空复杂度。但是该算法平等对待每个视图所包含的信息,没有考虑各个视图之间的差异,同时忽略了视图数据和所学锚图之间的局部结构信息,可能制约获得更有效的子空间锚图矩阵。

2 AMVC-AFGL 算法设计

本节将详细描述基于自适应融合全局和局部信息的锚点多视图聚类(AMC-AFGL)算法,包括问题建模、解决方案以及算法复杂度和收敛性分析。

2.1 问题建模

为了从数据中学习一个更有效的锚图表示,在一个统一的框架中融合原始数据和锚图之间全局和局部相关性的信息,同时考虑对各个视图的重要程度自适应地分配权重。

第一阶段是进行锚图构造。给定多视图数据 $\{\boldsymbol{X}^v\}_{v=1}^c$,$\boldsymbol{X}^v \in \mathbf{R}^{d_v\times n}$ 是第 v 个视图上特征维度为 d_v 的数据,从原始数据中利用 k-means 算法选取锚点矩阵 $\boldsymbol{A}^v = \{\boldsymbol{a}_j\}_{j\in[1,m]}$,进而从数据中学习锚图矩阵 $\boldsymbol{Z}^v \in \mathbf{R}^{n\times m}(m \ll n)$。这在降低数据维度和计算复杂度的同时,使得它们的乘积尽可能逼近原视图数据 $\boldsymbol{X}^v$,即 $\boldsymbol{X}^v \approx \boldsymbol{A}^v\boldsymbol{Z}^{v\mathrm{T}}$。$\boldsymbol{A}^v$ 中的每一列可以被视作是一个基向量,它捕获了数据 $\boldsymbol{X}^v$ 中更高层次的特征,$\boldsymbol{Z}^{v\mathrm{T}}$ 的每一列是原始输入相对于新基 $\boldsymbol{A}^v$ 的 m 维表示。如式(3)所示,

$$\boldsymbol{x}_i^v \approx \sum_{j=1}^m z_{ij}^v \boldsymbol{a}_j^v, \tag{3}$$

其中:$\boldsymbol{x}_i^v$ 是数据 $\boldsymbol{X}^v$ 中的第 i 个数据点;$\boldsymbol{a}_j^v$ 是锚点矩阵 $\boldsymbol{A}^v$ 的第 j 个列向量;z_{ij}^v 是锚图 $\boldsymbol{Z}^v$ 中的第 ij 个元素。

由式(3)可知,一个很自然的假设是,当数据点 $\boldsymbol{x}_i^v$ 越靠近 $\boldsymbol{a}_j^v$,z_{ij}^v 应该越大,这便是数据点和锚图之间的局部信息。所以,认为融合每个视图数据和锚图之间的局部结构信息,可以获得更有效的子空间锚图矩阵。于是在第一阶段,提出目标式学习子空间锚图矩阵,

$$\begin{aligned}&\min_{\boldsymbol{Z}^v,\boldsymbol{\alpha}^v}\sum_{v=1}^c \alpha^v\left(\|\boldsymbol{X}^v - \boldsymbol{A}^v\boldsymbol{Z}^{v\mathrm{T}}\|_F^2 + \gamma\|\boldsymbol{Z}^v\|_F^2 + \right.\\ &\left.\beta\sum_{i=1}^n\sum_{j=1}^m \|\boldsymbol{x}_i^v - \boldsymbol{a}_j^v\|_F^2 z_{ij}^v\right) + \mu\|\boldsymbol{\alpha}\|_2^2,\\ &\text{s.t. } 0 \leqslant \boldsymbol{Z}^v, \boldsymbol{Z}^{v\mathrm{T}}\mathbf{1}_n = 1_m, \boldsymbol{\alpha}^v > 0, \boldsymbol{\alpha}^{\mathrm{T}}\mathbf{1}_c = 1,\end{aligned} \tag{4}$$

其中:$\boldsymbol{X}^v \in \mathbf{R}^{d_v\times n}$ 是第 v 个视图数据,其维度为 d_v;$\boldsymbol{A}^v = \{\boldsymbol{a}_j\}_{j\in[1,m]}$ 是从 $\boldsymbol{X}^v$ 中通过 k-means 算法选取的锚点矩阵;$\boldsymbol{Z}^v \in \mathbf{R}^{n\times m}(m \ll n)$ 是期望学到的子空间锚图矩阵;$\beta > 0, \gamma > 0$ 和 $\mu > 0$ 是平衡参数;$\boldsymbol{\alpha} = [\alpha^1, \cdots, \alpha^v, \cdots, \alpha^c]^{\mathrm{T}}$,$\alpha^v$ 是衡量第 v 个视图重要性程度的权重系数。

考虑各视图数据对聚类效果的贡献高低不同,所以为各视图分配权重 α^v。不同于传统做法,受 Nie 等提出的自加权多视图聚类算法[7-8]的启发,不把 α^v 当作变量来进行优化学习,而是自适应为每个视图分配合适的权重,以学习更有效的子空间锚图矩阵。而且,自适应分配权重 α^v,能够减少式(4)中的优化变量 α^v 和超参数 μ,可以进一步将式(4)改写为

$$\begin{aligned}&\min_{\boldsymbol{Z}^v}\sum_{v=1}^c \alpha^v\left(\|\boldsymbol{X}^v - \boldsymbol{A}^v\boldsymbol{Z}^{v\mathrm{T}}\|_F^2 + \gamma\|\boldsymbol{Z}^v\|_F^2 + \right.\\ &\left.\beta\sum_{i=1}^n\sum_{j=1}^m \|\boldsymbol{x}_i^v - \boldsymbol{a}_j^v\|_F^2 z_{ij}^v\right),\\ &\text{s.t. } 0 \leqslant \boldsymbol{Z}^v, \boldsymbol{Z}^{v\mathrm{T}}\mathbf{1}_n = 1_m,\end{aligned} \tag{5}$$

其中:权重 α^v 的自适应更新方案见 2.2 小节。

第二阶段是进行谱聚类。首先,根据文献[3]的算法,采用传统的按列拼接操作得到融合锚图 $\boldsymbol{Z}=[\boldsymbol{Z}^1,\cdots,\boldsymbol{Z}^v,\cdots,\boldsymbol{Z}^c]\in\mathbf{R}^{n\times m\times c}$。显然,对融合锚图 $\boldsymbol{Z}$ 直接进行谱聚类是不可行的,因为锚图 $\boldsymbol{Z}\in\mathbf{R}^{n\times m\times c}(m\ll n)$ 的维度不再是 $n\times n$。对多视图数据,遵循前人的算法,即利用公式(4)来重建自表达系数矩阵 $\boldsymbol{S}\in\mathbf{R}^{n\times n}$。接下来,可以对 $\boldsymbol{S}$ 进行谱聚类得到谱嵌入 $\boldsymbol{Q}\in\mathbf{R}^{n\times k}$,

$$\max_{\boldsymbol{Q}}\ \mathrm{Tr}(\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{Q}),\ \mathrm{s.t.}\ \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q}=\boldsymbol{I}。\tag{6}$$

公式(6)的解是 $\boldsymbol{S}$ 的前 k 个特征值对应的特征向量矩阵,但是其特征分解复杂度至少为 $O(n^2k)$。根据文献[11]和[18],融合锚图矩阵 $\boldsymbol{Z}$ 的左奇异向量与自表达系数矩阵 $\boldsymbol{S}$ 的特征向量是等价的。因此本文直接对 $\boldsymbol{Z}$ 进行奇异值分解(SVD),计算其前 k 个左奇异向量构成的矩阵 $\boldsymbol{Q}\in\mathbf{R}^{n\times k}$,然后对矩阵 $\boldsymbol{Q}$ 进行简单的 k-means 聚类即可得到聚类结果。

本文完整的算法框架示意图如图 1 所示。

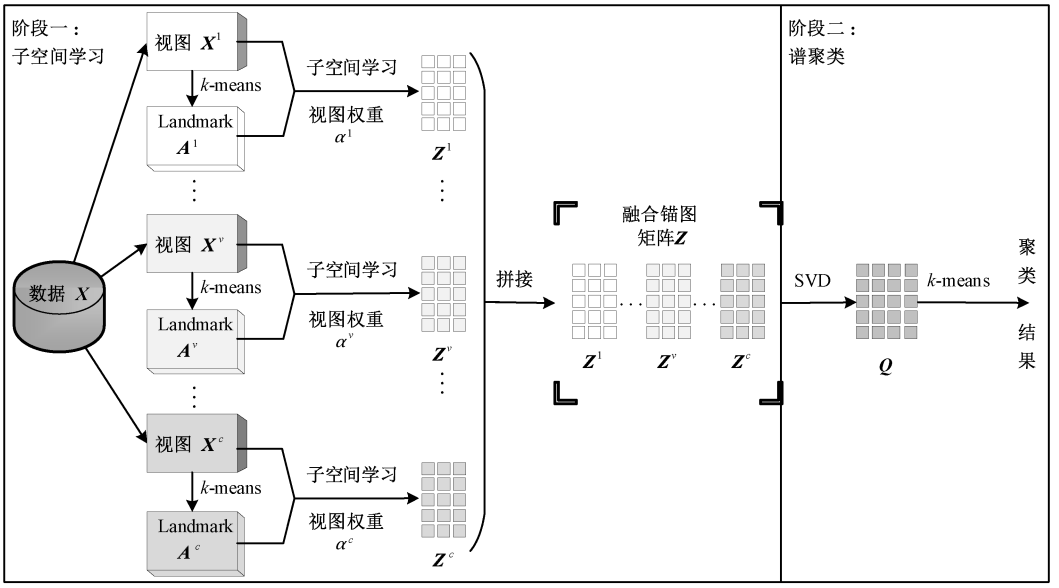

图 1　AMVC-AFGL 示意图

Figure 1　Illustration of AMVC-AFGL

2.2　解决方案

受自加权多视图聚类算法[7-8]的启发,可以提出式(7)来自动引导权重的学习,

$$\min_{0\leqslant\boldsymbol{Z}^v,\boldsymbol{Z}^{v\mathrm{T}}\mathbf{1}_n=1_m}\ \sum_{v=1}^{c}\sqrt{\mathcal{F}(\boldsymbol{Z}^v)},\tag{7}$$

其中:$\mathcal{F}(\boldsymbol{Z}^v)$ 计算为

$$\mathcal{F}(\boldsymbol{Z}^v)=\|\boldsymbol{X}^v-\boldsymbol{A}^v\boldsymbol{Z}^{v\mathrm{T}}\|_F^2+\gamma\|\boldsymbol{Z}^v\|_F^2+\beta\sum_{i=1}^{n}\sum_{j=1}^{m}\|\boldsymbol{x}_i^v-\boldsymbol{a}_j^v\|_F^2 z_{ij}^v。$$

式(7)看起来非常简化和紧凑,但是没有显式定义权重因子。将它当作一个正常的问题并利用拉格朗日乘子法来求解,首先写出式(7)关于 $\boldsymbol{Z}^v$ 的拉格朗日函数,

$$\sum_{v=1}^{c}\sqrt{\mathcal{F}(\boldsymbol{Z}^v)}+\mathcal{L}(\Lambda,\boldsymbol{Z}^v),\tag{8}$$

其中:Λ 是拉格朗日乘子;$\mathcal{L}(\Lambda,\boldsymbol{Z}^v)$ 是由约束条件导出的,求式(8)关于 $\boldsymbol{Z}^v$ 的导数并令其为 0,得到

$$\sum_{v=1}^{c}\alpha^v\frac{\partial\mathcal{F}(\boldsymbol{Z}^v)}{\partial\boldsymbol{Z}^v}+\frac{\partial\mathcal{L}(\Lambda,\boldsymbol{Z}^v)}{\partial\boldsymbol{Z}^v}=0,\tag{9}$$

α^v 计算为

$$\alpha^v=\frac{1}{2\sqrt{\mathcal{F}(\boldsymbol{Z}^v)}}。\tag{10}$$

显然,从式(10)知道 α^v 依赖于 $\boldsymbol{Z}^v$,这就意味着式(9)中第一项的两个因子是相互耦合的。但是如果把 α^v 固定下来,那么式(9)可以认为是式(5)的求解。所以,可以用式(4)的计算结果通过公式(10)进一步更新 α^v。这启示通过迭代地交替优化 $\boldsymbol{Z}^v$ 和更新 α^v 来解决问题(5)。

当视图权重 α^v 固定时,优化求解锚图 $\boldsymbol{Z}^v$。通过化简并移除问题(5)中的无关项,可以把式(5)重写为

$$\min_{\boldsymbol{Z}^v}\sum_{v=1}^{c}\alpha^v(\mathrm{tr}(\boldsymbol{Z}^{v\mathrm{T}}\boldsymbol{Z}^v(\boldsymbol{A}^{v\mathrm{T}}\boldsymbol{A}^v+\gamma\boldsymbol{I}))-2\mathrm{tr}(\boldsymbol{Z}^{v\mathrm{T}}\boldsymbol{X}^{v\mathrm{T}}\boldsymbol{A}^v)+\beta\sum_{i=1}^{n}\sum_{j=1}^{m}\|\boldsymbol{x}_i^v-\boldsymbol{a}_j^v\|_F^2 z_{ij}^v),$$
$$\mathrm{s.t.}\ \ 0\leqslant\boldsymbol{Z}^v,\boldsymbol{Z}^{v\mathrm{T}}\mathbf{1}_n=1_m。\tag{11}$$

式(11)中 $\boldsymbol{Z}^v$ 的求解在各个视图上是彼此独立的,因此能够在各个视图上依次求解 $\boldsymbol{Z}^v$。$\boldsymbol{Z}^v$ 中每个列向量是彼此独立的,记 $\boldsymbol{z}_j^v$ 为 $\boldsymbol{Z}^v$ 的第 j 个列向量,可以把式(11)表述为优化求解 $\boldsymbol{z}_j^v$ 的形式,具体展开化简为

$$\min_{\boldsymbol{z}_j^v} \frac{1}{2}\boldsymbol{z}_j^v \boldsymbol{H}^v \boldsymbol{z}_j^v + \boldsymbol{f}^{v\mathrm{T}} \boldsymbol{z}_j^v,\ \text{s.t.}\ 0 \leqslant \boldsymbol{z}_j^v, \boldsymbol{z}_j^{v\mathrm{T}} \boldsymbol{1}_n = 1, \tag{12}$$

其中:$\boldsymbol{H}^v = \alpha^v(\boldsymbol{A}^{v\mathrm{T}}\boldsymbol{A}^v + \gamma \boldsymbol{I})$;$\boldsymbol{f}^v = [f_1^v, \cdots, f_m^v]^{\mathrm{T}}$ 是1个向量,$\boldsymbol{f}^v$ 中的第 j 个元素为 $f_j^v = \alpha^v\beta([\boldsymbol{X}^{v\mathrm{T}}\boldsymbol{A}^v]_{ij} + \|\boldsymbol{x}_i^v - \boldsymbol{a}_j^v\|_F^2)$。至此,对式(12),可以很容易地利用MATLAB的内置工具包 *quadprog* 进行凸二次规划求解 $\boldsymbol{Z}^v$。

完整的流程总结如算法1所示。

算法1 AMVC-AFGL算法。

输入:多视图数据 $(\boldsymbol{X}^1, \cdots, \boldsymbol{X}^c)$,簇类别数 k,参数 β 和 γ,锚点 $\{\boldsymbol{A}^v\}_{v\in[1,c]}$。

输出:数据聚类标签结果 $\hat{\boldsymbol{y}} \in \mathbf{R}^{n\times 1}$。

1) 初始化 $\alpha^v = 1/c$

2) repeat

3) for $v = 1:c$

4) 求解目标式(12)得到锚图矩阵 $\boldsymbol{Z}^v$

5) 通过公式(10)更新 α^v

6) end for

7) until 达到预设的迭代次数或者目标函数值变化低于预设阈值

8) 将锚图矩阵 $\boldsymbol{Z}^v$ 拼接得到融合锚图 $\boldsymbol{Z} = [\boldsymbol{Z}^1, \cdots, \boldsymbol{Z}^v, \cdots, \boldsymbol{Z}^c] \in \mathbf{R}^{n\times m\times c}$

9) 对融合锚图矩阵 $\boldsymbol{Z}$ 进行奇异值分解,得到其前 k 个左奇异向量构成的矩阵 $\boldsymbol{Q}$

10) 在 $\boldsymbol{Q}$ 上运行 k-means 聚类算法得到数据聚类标签结果 $\hat{\boldsymbol{y}} \in \mathbf{R}^{n\times 1}$

2.3 算法复杂度和收敛性分析

本文算法模型得益于图构建的锚策略和低维矩阵上特征分解的低复杂度。具体来说,由式(5)可以知道,锚图矩阵 $\boldsymbol{Z}^v$ 的构造学习复杂度为 $O(nm^3cT)$,其中:n 是数据点的样本数;m 是锚点数,且 $m \ll n$;c 是数据的总视图数;T 是优化求解迭代次数。在本文算法实验中,应用Matlab的内置二次规划函数 *quadprog* 来求解目标式(12),其中使用了复杂度为 $O(m^3)$ 的interior-point-convex算法。值得一提的是,锚图矩阵 $\boldsymbol{Z}^v$ 的构建可以很容易地在多个核上并行化,从而实现更高效的计算。谱嵌入矩阵 $\boldsymbol{Q}$ 的计算复杂度为 $O(m^3c^3 + 2mcn)$。此外,在开始时使用 k-means 算法来为每个数据视图 $\boldsymbol{X}^v$ 选择对应的锚点矩阵 $\boldsymbol{A}^v$,其复杂度为 $O(nmtd)$,并需要复杂度 $O(nk^2t)$ 用于 $\boldsymbol{Q}$ 上执行最后的 k-means 算法得到聚类结果,其中 t 是 k-means 迭代次数,k 是聚类簇的数量,$d = \sum_{v=1}^{c} d_v$ 是多视图数据所有视图特征维度的总和。因此,提出的AMVC-AFGL算法模型在解决大规模多视图聚类问题中的时间开销是关于数据样本数 n 的线性复杂度。

本文提出的优化算法,是通过迭代地交替优化 $\boldsymbol{Z}^v$ 和更新 α^v 来解决问题(5),当 α^v 固定时,需要求解的问题(12)便是凸的,这确保了本文的算法能够收敛到最优解,而且可以很容易地利用凸二次规划进行求解。

3 实验设计与结果分析

所提算法在10个公开的真实数据集上进行了充分实验,并且与当前12个先进的多视图聚类算法进行对比,以评估本文所提算法的性能。实验环境配置如下。

1) 硬件:Intel core i9-10900X 处理器八核,3.7 GHz 主频,64 GB RAM;

2) 软件:Windows 10 操作系统,Matlab 2020b 仿真环境。

3.1 数据集简介

本文所用的10个公开真实多视图数据集,Caltech101-7和Caltech101-20均是被广泛使用的物体识别数据集Caltech101(https://www.vision.caltech.edu/datasets/)的一个子集,前者包含7类共1 474张图像,后者包含20类共2 386张图像。Mfeat(http://archive.ics.uci.edu/ml/datasets/Multiple+Features)是一个0~9数字组成的手写数据集。BDGP(https://www.fruitfly.org/)是包含5种不同果蝇胚胎样本组成的共2 500张图像数据集。Wiki(http://svcl.ucsd.edu/projects/crossmodal/)数据集包含了2 866个维基百科的条目,每个条目由图像和文本两种表示。Reuters-7200和Reuters是路透社多语种文件和相应翻译数据集。YTF10、YTF20和YTF50分别是从YoutubeFace(https://www.cs.tau.ac.il/~wolf/ytfaces/)面部视频数据集上生成的类别数分别为10、20和50的子集。

3.2 对比算法

1) MSC-IAS算法[3]通过融合各个视图间的互补信息来学习完备空间,利用Hilbert-Schmidt独立

性准则保证构建的相似度与潜在的完备点具有最大的相关性。

2) PMSC 算法[16]在统一框架下协同优化各视图的图学习、基本划分生成和共识划分融合。

3) UOMVSC 算法[21]将不同视图的图和嵌入矩阵结合起来,得到一个统一的图,并且用一种有效的优化算法直接从统一图中获取离散聚类指标矩阵。

4) FMR 算法[22]为了充分利用数据信息进行数据重构,灵活地对来自各个视图之间的互补信息进行编码。

5) SFMC 算法[20]采用一种新的策略,根据锚点的得分对其进行采样,同时对构建的锚点图施加秩约束,并直接输出聚类结果,而无须后处理过程。

6) MLRSSC 算法[10]对构建的亲和矩阵施加低秩约束和稀疏约束,学习所有视图之间共享的子空间自表示矩阵。

7) AMGL 算法[7]提出一种自加权技术,自适应地为每个视图数据分配权重,并能确保模型收敛到全局最优。

8) RMKM 算法[23]有效集成大规模多视图数据的异构表示,提升大规模多视图聚类性能。

9) BMVC 算法[24]是第一个使用二进制编码技术解决大规模多视图聚类问题的算法,能同时从多个视图联合优化二进制编码和聚类,有效降低了算法的计算开销和存储空间。

10) LMVSC 算法[11]受锚点策略的启发,利用 k-means 算法得到每个视图数据的初始锚点矩阵,而后为每个视图学习一个子空间自表达矩阵,最后在集成的较小自表达矩阵上进行谱聚类,有效降低了计算复杂度。

11) SMVSC 算法[17]通过优化学习策略得到锚点,并非通过传统的采样策略获得,有效解决了聚类性能受初始锚点限制的问题。

12) FPMVS 算法[18]将锚点学习和锚图构造结合成一个统一的框架来共同优化,两者协同促进聚类性能提升。

3.3　实验设置

本文所提算法有 3 个超参数:锚点数 m,平衡参数 β 和 γ。其中:锚点数 m 是从 $[k,2k,3k,5k,10k]$ 中遍历选择;k 是数据类别数;β 和 γ 均在 $[0.001,0.01,0.1,1,10]$ 中遍历选择。对比算法的参数设置,则是根据对应文献中给出的建议进行实验,没有给出最佳参数的,则通过网格化搜索对应算法的最佳参数进行公平比较。值得注意的是,对于算法在运行过程中需要进行 k-means 得到聚类结果的实验,k-means 运行 30 次后取平均值作为最终评估。在实验中,采用 3 个常用聚类评价标准评估本文模型的聚类性能,包括准确率(accuracy,ACC)、标准化互信息(normalized mutual information,NMI)和纯度(Purity)。另外,在实验中也记录算法在取得最佳聚类性能时,算法去掉数据预处理后的运行时间(Time),以衡量本文模型在大型数据集上的实用性和可扩展性。

3.4　实验结果分析

本文选取了 10 个广泛使用的真实数据集来评估所提算法的性能,表 1~3 展示了本文算法与其他对比算法在 10 个基准数据集上的聚类结果。由表 1~3 可得以下结论。

1) 就 *ACC* 指标来说,本文所提出的算法在 10 个基准数据集上的聚类效果均能达到最优或次优。本文模型在除了 Caltech101-20 数据集外的 9 个数据集,*Purity* 指标都能达到最优或次优,其中,在数据集 BDGP、Wiki、Reuters-7200、YTF10、YTF20、YTF50 上的最优结果分别超出次优结果 6.2%、4.5%、21%、3.6%、1.6%、2.1%。相比传统子空间聚类算法的突出性能,本文算法仍然在大部分数据集上取得了更佳的效果,这显示了基于锚点的子空间聚类算法的有效性。

2) LMVSC 算法是基于 k-means 得到锚点矩阵,然后学习子空间表示矩阵再进行谱聚类,其在各个数据集上的聚类性能都有着不俗的表现,但它没有考虑视图间的差异和冗余信息,也忽视了数据与锚点的局部结构信息。而本文采用自适应加权机制来融合数据与锚点间的全局和局部结构信息,依据视图权重自适应更新来学习更有效的子空间锚图矩阵,在各个数据集上的聚类性能显示了本文算法的有效性。

3) SMVSC 与 FPMVS 算法是通过学习策略获取相适应的锚点,规避了采样策略固定锚点影响聚类性能的问题,而本文提出的 AMVC-AFGL 算法使用了更具可扩展性和稳定性的启发式采样策略来获取数据锚点,在效果上与学习策略更新锚点的算法相比,AMVC-AFGL 算法在部分数据集上具有相近

表1 不同聚类算法在多视图数据集上的 *ACC* 值

Table 1 *ACC* of different clustering algorithms on multi-view datasets 单位:%

数据集	MSC-IAS	PMSC	FMR	UOMVSC	SFMC	MLRSSC	AMGL	RMKM	BMVC	LMVSC	SMVSC	FPMVS	本文算法
Caltech101-7	39.76	58.34	43.96	67.10	67.71	37.31	39.97	38.13	**82.90**	59.91	46.27	58.82	**73.17**
Mfeat	85.95	72.85	68.40	**97.25**	56.90	20.01	71.42	80.80	65.80	91.75	70.60	71.30	**93.45**
Caltech101-20	31.27	48.91	36.97	46.56	40.32	28.21	28.90	40.74	**55.28**	47.82	41.79	50.17	**50.78**
BDGP	**52.10**	29.20	41.88	40.96	41.26	36.12	34.48	41.36	39.76	50.12	38.68	35.04	**52.36**
Wiki	23.91	59.49	60.68	55.97	16.22	15.77	12.23	54.36	47.21	**60.85**	57.47	58.20	**61.95**
Reuters-7200	—	—	—	—	16.00	18.44	16.79	27.36	**36.06**	28.82	29.38	17.60	**30.21**
Reuters	—	—	—	—	—	—	16.72	40.12	52.85	54.34	**58.90**	46.59	**57.71**
YTF10	—	—	—	—	—	—	—	**75.68**	58.58	69.74	73.29	69.89	**74.96**
YTF20	—	—	—	—	—	—	—	57.68	57.39	64.64	**68.21**	65.37	**70.22**
YTF50	—	—	—	—	—	—	—	—	66.00	**69.32**	68.23	67.21	**68.65**

注:最优结果用黑体加下划线数字表示;次优结果用黑体数字表示;"—"代表算法由于内存不足无法在该数据集上运行。

表2 不同聚类算法在多视图数据集上的 *NMI* 值

Table 2 *NMI* of different clustering algorithms on multi-view datasets 单位:%

数据集	MSC-IAS	PMSC	FMR	UOMVSC	SFMC	MLRSSC	AMGL	RMKM	BMVC	LMVSC	SMVSC	FPMVS	本文算法
Caltech101-7	24.55	53.20	40.77	**65.07**	55.51	21.11	44.99	40.55	**67.25**	49.89	32.49	36.12	56.38
Mfeat	**87.68**	65.24	67.67	**94.05**	68.15	28.63	77.12	82.28	59.39	85.54	60.02	61.24	85.83
Caltech101-20	31.38	53.89	50.52	**63.88**	36.27	26.70	47.95	50.60	49.79	55.01	39.90	45.57	**56.65**
BDGP	**33.07**	5.04	12.64	16.91	16.62	26.33	17.21	16.98	15.74	26.36	10.34	9.69	**29.09**
Wiki	8.65	52.03	51.90	52.46	1.36	0.08	0.83	**53.42**	41.59	50.87	52.58	53.23	**55.44**
Reuters-7200	—	—	—	—	**12.86**	4.16	0.27	5.56	**12.82**	7.61	8.19	0.19	8.14
Reuters	—	—	—	—	—	—	0.19	20.43	26.51	33.82	**37.21**	31.81	**35.31**
YTF10	—	—	—	—	—	—	—	78.83	54.66	**81.98**	79.28	**80.32**	79.35
YTF20	—	—	—	—	—	—	—	73.84	71.65	77.36	**81.99**	77.59	**78.25**
YTF50	—	—	—	—	—	—	—	—	81.90	83.10	**83.94**	**83.15**	81.98

注:最优结果用黑体加下划线数字表示;次优结果用黑体数字表示;"—"代表算法由于内存不足无法在该数据集上运行。

表3 不同聚类算法在多视图数据集上的 *Purity* 值

Table 3 *Purity* of different clustering algorithms on multi-view datasets 单位:%

数据集	MSC-IAS	PMSC	FMR	UOMVSC	SFMC	MLRSSC	AMGL	RMKM	BMVC	LMVSC	SMVSC	FPMVS	本文算法
Caltech101-7	44.44	85.14	80.87	**88.26**	85.48	41.45	40.40	82.77	82.90	83.58	75.64	78.02	**85.65**
Mfeat	87.20	72.85	71.80	**97.25**	57.40	20.01	73.77	83.50	65.80	91.75	70.60	71.30	**93.45**
Caltech101-20	33.74	69.74	69.40	**78.29**	45.05	30.39	30.97	70.75	60.44	**73.43**	59.81	63.70	55.02
BDGP	**53.52**	30.92	42.44	43.12	42.37	36.12	35.75	42.12	40.52	50.12	38.68	35.80	**56.84**
Wiki	26.68	61.34	60.89	60.57	16.40	15.77	12.51	**62.11**	49.27	61.34	61.41	61.79	**64.96**
Reuters-7200	—	—	—	—	25.31	18.44	16.80	28.26	37.54	29.29	**29.92**	17.96	**36.21**
Reuters	—	—	—	—	—	—	16.76	40.16	56.47	58.99	**64.17**	56.59	**61.42**
YTF10	—	—	—	—	—	—	—	**79.70**	63.87	74.78	79.29	77.05	**82.62**
YTF20	—	—	—	—	—	—	—	68.78	64.76	71.89	**75.79**	65.97	**77.01**
YTF50	—	—	—	—	—	—	—	—	73.64	**75.32**	73.45	71.67	**76.90**

注:最优结果用黑体加下划线数字表示;次优结果用黑体数字表示;"—"代表算法由于内存不足无法在该数据集上运行。

甚至更好的结果。

3.5 参数敏感性分析

由于本文所提算法的平衡参数 β 和 γ 均在 $[0.001, 0.01, 0.1, 1, 10]$ 中遍历选择,其取值范围变化较大,为进一步分析两个平衡参数对本文所提算法聚类性能的影响,在8个数据集上开展了对比实验。如图2所示,在所有数据集上固定锚点数为类别数 k。可以看出,在大部分数据集上,算法的聚类性能较为稳定,*ACC* 变化都不大。只在 Wiki 数据集上,当 $\beta = 10$ 时,*ACC* 随着 γ 的改变有较大变化。总体来讲,本文提出的 AMVC-AFGL 算法对平衡参数 β 和 γ 不敏感。

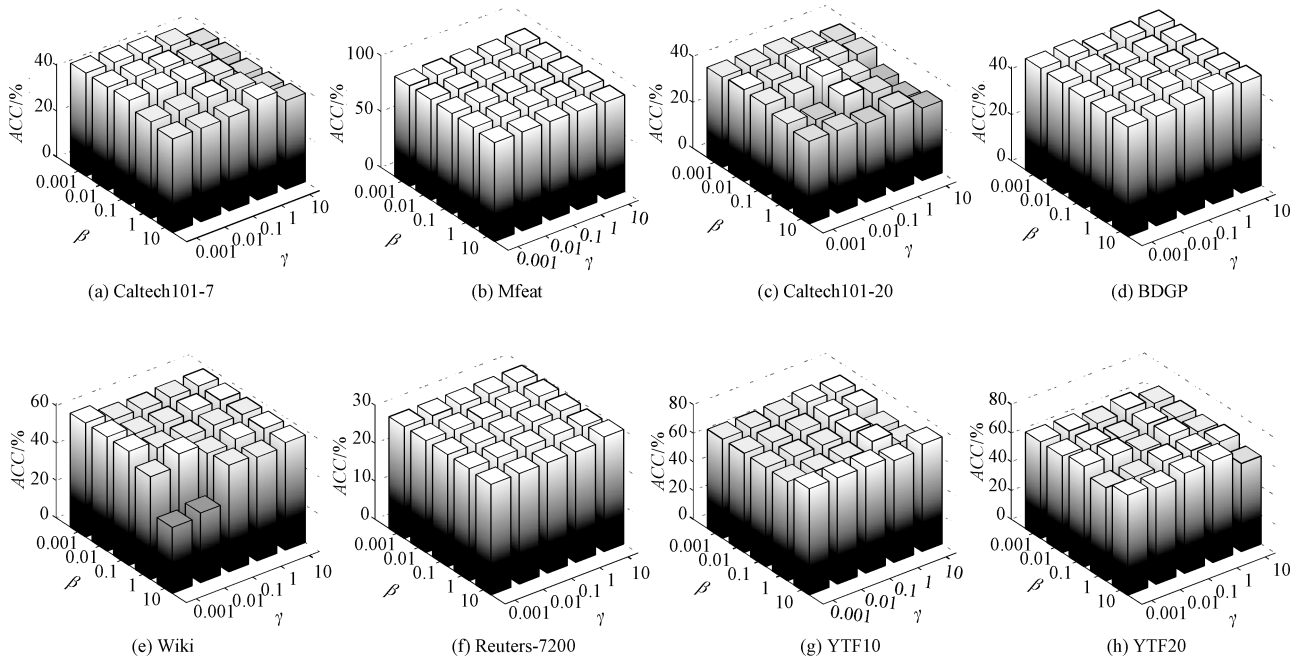

图 2　平衡参数对聚类性能的影响

Figure 2　The influence of balance parameters on clustering performance

3.6　运行效率分析

表 4 列举了本文算法和各个对比算法在 10 个基准数据集上取得最佳聚类效果时,免去数据预处理部分之后的运行时间。从表 4 中可知,大部分算法随着数据集样本数量的增加,其运行时间也相应延长,只是不同算法的运行时间随样本数增加的幅度不完全相同。大部分传统多视图聚类算法由于计算复杂度和存储开销较高,在大型多视图数据集上进行聚类任务会受到内存的限制,甚至无法在大规模数据集上运行。从表 4 中可以看到,与大多数的对比算法相比,AMVC-AFGL 算法具有可观的运行效率。虽然在一些中小型数据集上,BMVC 与 LMVSC 算法的运行时间较短,但它们的聚类性能偏弱,而本文提出的 AMVC-AFGL 算法在一定程度上可以统筹兼顾聚类性能和运行效率,具有较强的可扩展性。

表 4　对比算法在多视图数据集上获得最佳聚类效果的运行时间

Table 4　Running time of compared methods to achieve optimal clustering performance on multi-view datasets 单位:s

数据集	MSC-IAS	PMSC	FMR	UOMVSC	SFMC	MLRSSC	AMGL	RMKM	BMVC	LMVSC	SMVSC	FPMVS	本文算法
Caltech101-7	6.85	1 516.23	18.42	410.56	8.62	3.32	2.86	2.45	15.23	2.65	2.89	4.26	60.56
Mfeat	16.81	3 300.30	251.03	11 528.80	88.62	27.94	19.62	3.95	0.43	2.96	1.38	1.42	1.82
Caltech101-20	14.62	4 512.60	563.20	2 356.20	58.21	25.61	30.56	5.28	20.87	2.98	2.96	5.41	28.15
BDGP	13.26	15 215.20	1 070.40	34 800.10	39.23	26.89	73.71	7.53	0.35	2.86	1.63	3.36	2.12
Wiki	15.92	14 386.60	1 068.80	9 991.70	9.84	30.72	180.62	6.27	0.11	2.86	3.15	21.12	0.85
Reuters-7200	—	—	—	—	39.51	486.68	1 251.23	25.00	0.88	20.46	13.79	12.56	12.92
Reuters	—	—	—	—	—	—	10 594.21	202.32	8.41	68.53	66.24	62.23	117.12
YTF10	—	—	—	—	—	—	—	675.42	108.20	196.70	253.22	995.62	105.45
YTF20	—	—	—	—	—	—	—	1 780.50	80.53	513.52	720.22	1 685.24	219.81
YTF50	—	—	—	—	—	—	—	—	65.71	3 535.72	2 254.32	9 176.46	206.57

注:“—”代表算法由于内存不足无法在该数据集上运行。

3.7　消融实验

为了验证正则化项、视图局部结构信息和多视图自加权机制的有效性,去掉优化目标式中的正则化项作为无正则化项的对比算法,去掉视图局部结构信息项作为无局部结构的对比算法,去掉视图加权机制作为无视图加权的对比算法,在 10 个基准数据集上开展了充分的消融实验。如表 5 所示,本文算法相比于三种对比算法更具先进性,这表明了本文所提 AMVC-AFGL 算法中的多视图自适应加权机制和融合局部结构信息的有效性。

表5 消融实验结果

Table 5 Results of ablation experiments

单位:%

数据集	*ACC*				*NMI*				*Purity*			
	无正则化项	无局部结构	无视图加权	本文算法	无正则化项	无局部结构	无视图加权	本文算法	无正则化项	无局部结构	无视图加权	本文算法
Caltech101-7	70.23	70.62	71.85	**73.17**	48.65	52.24	53.12	**56.38**	80.45	79.09	81.85	**85.65**
Mfeat	90.21	89.15	91.36	**93.45**	81.28	80.31	82.25	**85.83**	89.81	89.65	90.24	**93.45**
Caltech101-20	46.24	45.62	48.86	**50.78**	53.46	52.98	55.21	**56.65**	51.25	50.09	52.26	**55.02**
BDGP	49.63	48.52	50.15	**52.36**	26.58	24.02	25.41	**29.09**	54.22	53.85	55.06	**56.84**
Wiki	58.65	57.28	59.69	**61.95**	52.06	51.28	53.49	**55.44**	62.24	61.51	62.82	**64.96**
Reuters-7200	26.14	25.56	27.25	**30.21**	6.95	6.89	7.12	**8.14**	32.05	31.89	35.08	**36.21**
Reuters	52.39	55.81	56.47	**57.71**	28.06	29.51	31.25	**35.31**	58.12	56.38	58.61	**61.42**
YTF10	67.12	65.25	68.89	**74.96**	74.96	73.85	75.54	**79.35**	78.24	75.68	79.02	**82.62**
YTF20	62.42	65.63	66.58	**70.22**	72.08	72.86	75.18	**78.25**	75.65	73.48	75.81	**77.01**
YTF50	61.21	62.45	65.22	**68.65**	74.66	76.84	78.45	**81.98**	73.21	70.68	73.62	**76.90**

注:最优结果用黑体表示。

4 结语

为解决多视图聚类算法复杂度高,难以应用在大规模、高维度多视图数据上的问题,基于锚点策略的多视图子空间聚类算法受到广泛关注。受多视图自加权多图学习算法的启发,本文提出了基于自适应融合全局和局部信息的锚点多视图聚类算法(AMVC-AFGL)。所提算法旨在通过自适应加权视图的重要性,在一个统一的框架中融合原始数据、锚点数据和所学锚图矩阵之间的全局信息和局部信息。在公开的10个真实基准数据集上开展了充分的实验,与其他的12个多视图聚类算法相比,所提算法具有有效性和可扩展性。

参考文献:

[1] 刘振鹏, 陈杰, 王仕磊, 等. 基于聚类和压缩感知的高维数据发布算法[J]. 郑州大学学报(理学版), 2023, 55(2): 63-69.
LIU Z P, CHEN J, WANG S L, et al. High dimensional data publishing algorithm based on clustering and compressed sensing[J]. Journal of Zhengzhou university (natural science edition), 2023, 55(2): 63-69.

[2] ŽITNIK M, ZUPAN B. Data fusion by matrix factorization[J]. IEEE transactions on pattern analysis and machine intelligence, 2015, 37(1): 41-53.

[3] WANG X B, LEI Z, GUO X J, et al. Multi-view subspace clustering with intactness-aware similarity[J]. Pattern recognition, 2019, 88: 50-63.

[4] LIU J L, WANG C, GAO J, et al. Multi-view clustering via joint nonnegative matrix factorization[C]//Proceedings of the SIAM International Conference on Data Mining. Philadelphia: SIAM, 2013: 252-260.

[5] ZONG L L, ZHANG X C, ZHAO L, et al. Multi-view clustering via multi-manifold regularized non-negative matrix factorization[J]. Neural networks, 2017, 88: 74-89.

[6] KUMAR A, RAI P, DAUMÉ H. Co-regularized multi-view spectral clustering[C]//Proceedings of the 24th International Conference on Neural Information Processing Systems. New York: ACM Press, 2011: 1413-1421.

[7] NIE F P, LI J, LI X L. Parameter-free auto-weighted multiple graph learning: a framework for multiview clustering and semi-supervised classification[C]//Proceedings of the Twenty-fifth International Joint Conference on Artificial Intelligence. New York: ACM Press, 2016: 1881-1887.

[8] NIE F P, LI J, LI X L. Self-weighted multiview clustering with multiple graphs[C]//Proceedings of the Twenty-sixth International Joint Conference on Artificial Intelligence. New Tork: ACM Press, 2017: 2564-2570.

[9] 黄宗超, 王思为, 祝恩, 等. 基于子空间融合的多视图聚类算法[J]. 郑州大学学报(理学版), 2021, 53(1): 68-73.
HUANG Z C, WANG S W, ZHU E, et al. Multi-view clustering algorithm based on subspace fusion[J]. Journal of Zhengzhou university (natural science edition), 2021, 53(1): 68-73.

[10] BRBIĆ M, KOPRIVA I. Multi-view low-rank sparse subspace clustering[J]. Pattern recognition, 2018, 73: 247-258.

[11] KANG Z, ZHOU W T, ZHAO Z T, et al. Large-scale multi-view subspace clustering in linear time[C]//Proceedings of the AAAI Conference on Artificial Intelligence. Palo Alto: AAAI Press, 2020, 4412-4419.

[12] CHEN X L, CAI D. Large scale spectral clustering with

landmark-based representation[C]//Proceedings of the Twenty-fifth AAAI Conference on Artificial Intelligence. Palo Alto: AAAI Press, 2011: 313-318.

[13] GAO H C, NIE F P, LI X L, et al. Multi-view subspace clustering[C]//Proceedings of the 2015 IEEE International Conference on Computer Vision. New York: ACM Press, 2015: 4238-4246.

[14] LUO S R, ZHANG C Q, ZHANG W, et al. Consistent and specific multi-view subspace clustering[C]//Proceedings of the AAAI Conference on Artificial Intelligence. Palo Alto: AAAI Press, 2018: 3730-3737.

[15] ZHANG C Q, HU Q H, FU H Z, et al. Latent multi-view subspace clustering[C]//IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE Press, 2017: 4279-4287.

[16] KANG Z, ZHAO X J, PENG C, et al. Partition level multi-view subspace clustering[J]. Neural networks, 2020, 122: 279-288.

[17] SUN M J, ZHANG P, WANG S W, et al. Scalable multi-view subspace clustering with unified anchors[C]//Proceedings of the 29th ACM International Conference on Multimedia. New York: ACM Press, 2021: 3528-3536.

[18] WANG S W, LIU X W, ZHU X Z, et al. Fast parameter-free multi-view subspace clustering with consensus anchor guidance[J]. IEEE transactions on image processing, 2022, 31: 556-568.

[19] LIU S Y, WANG S W, ZHANG P, et al. Efficient one-pass multi-view subspace clustering with consensus anchors[C]//Proceedings of the AAAI Conference on Artificial Intelligence. Palo Alto: AAAI Press, 2022: 7576-7584.

[20] LI X L, ZHANG H, WANG R, et al. Multiview clustering: a scalable and parameter-free bipartite graph fusion method[J]. IEEE transactions on pattern analysis and machine intelligence, 2022, 44(1): 330-344.

[21] TANG C, LI Z L, WANG J, et al. Unified one-step multi-view spectral clustering[J]. IEEE transactions on knowledge and data engineering, 2023, 35(6): 6449-6460.

[22] LI R H, ZHANG C Q, HU Q H, et al. Flexible multi-view representation learning for subspace clustering[C]//Proceedings of the 28th International Joint Conference on Artificial Intelligence. New York: ACM Press, 2019: 2916-2922.

[23] CAI X, NIE F P, HUANG H. Multi-view K-means clustering on big data[C]//Proceedings of the Twenty-third International Joint Conference on Artificial Intelligence. New York: ACM Press, 2013: 2598-2604.

[24] ZHANG Z, LIU L, SHEN F M, et al. Binary multi-view clustering[J]. IEEE transactions on pattern analysis and machine intelligence, 2019, 41(7): 1774-1782.

(上接第 22 页)

[12] TANG T T, LI C, LIU F G. Collaborative cloud-edge-end task offloading with task dependency based on deep reinforcement learning[J]. Computer communications, 2023, 209: 78-90.

[13] XU X L, ZHANG X Y, GAO H H, et al. BeCome: blockchain-enabled computation offloading for IoT in mobile edge computing[J]. IEEE transactions on industrial informatics, 2020, 16(6): 4187-4195.

[14] STAROVOITOV A. Net: filter: rework/optimize internal BPF interpreter's instruction set[EB/OL]. (2014-03-28)[2023-04-10]. https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=bd4cf0ed331a275e9bf5a49e6d0fd55dffc551b8.

[15] ISOVALENT. Cilium GitHub repository[EB/OL]. (2023-03-17)[2023-04-10]. https://github.com/cilium/cilium.

[16] HE Y, ZOU Z H, SUN K, et al. rapidpatch: firmware hotpatching for real-time embedded devices[C]//31st USENIX Security Symposium. Boston: USENIX Association, 2022: 2225-2242.

[17] Zephyr. A proven RTOS ecosystem, by developers, for developers[EB/OL]. (2023-02-19)[2023-04-10]. https://zephyrproject.org/.

[18] FFmpeg. A complete, cross-platform solution to record, convert and stream audio and video[EB/OL]. (2022-05-14)[2023-04-10]. https://ffmpeg.org/.

[19] EasyDarwin. Open source, high performance, industrial rtsp streaming server[EB/OL]. (2017-03-07)[2023-04-10]. https://github.com/EasyDarwin/EasyDarwin/.