Fast Multi-view Discrete Clustering with Anchor Graphs

Qianyao Qiang, [1] **Bin Zhang,** [1] **Fei Wang,** [2] **Feiping Nie** [3*]

[1]School of Software, Xi'an Jiaotong University, Xi'an 710049, China
[2]School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China
[3]School of Computer Science and Center for OPTical IMagery Analysis and Learning (OPTIMAL), Northwestern Polytechnical University, Xi'an 710072, Shaanxi, P. R. China
qiangqianyao@stu.xjtu.edu.cn, bzhang.xjtu@gmail.com, wfx@mail.xjtu.edu.cn, feipingnie@gmail.com

Abstract

Generally, the existing graph-based multi-view clustering models consists of two steps: (1) graph construction; (2) eigen-decomposition on the graph Laplacian matrix to compute a continuous cluster assignment matrix, followed by a post-processing algorithm to get the discrete one. However, both the graph construction and eigen-decomposition are time-consuming, and the two-stage process may deviate from directly solving the primal problem. To this end, we propose Fast Multi-view Discrete Clustering (FMDC) with anchor graphs, focusing on directly solving the spectral clustering problem with a small time cost. We efficiently generate representative anchors and construct anchor graphs on different views. The discrete cluster assignment matrix is directly obtained by performing clustering on the automatically aggregated graph. FMDC has a linear computational complexity with respect to the data scale, which is a significant improvement compared to the quadratic one. Extensive experiments on benchmark datasets demonstrate its efficiency and effectiveness.

Introduction

Multi-view clustering is a fundamental technique in multi-view data analysis. As an efficiency method for solving clustering problem, graph-based multi-view clustering has been widely investigated and used in pattern recognition, data mining, natural language processing, etc.

Most of the existing graph-based multi-view clustering models consists of two steps: (1) graph construction; (2) eigen-decomposition on the graph Laplacian matrix to compute a continuous cluster assignment matrix, followed by a post-processing algorithm such as k-means or spectral rotation to get the discrete one (Kang et al. 2020; Wang, Yang, and Liu 2020; Zhou et al. 2019; Nie et al. 2016a). Despite their attractive performance, there are still two drawbacks. One is the expensive time cost. Usually, the time complexity of traditional graph construction and eigen-decomposition operation are $O(n^2d)$ and $O(n^2c)$, respectively, where n is the data scale, d is the feature dimensionality and c is cluster number. The other is that the two-stage process of obtaining the interpretable discrete cluster assignment matrix may deviate from directly solving the primal problem.

Many efforts have been devoted to accelerate graph construction or eigen-decomposition, such as constructing anchor graph (Kang et al. 2020; Liu, He, and Chang 2010; Zhu, Nie, and Li 2017; Zhang et al. 2020) and sparse graph (Spielman and Teng 2011; Chen et al. 2010), approximate eigen-decomposition (Fowlkes et al. 2004). Anchors can be generated by random sampling or lightweight clustering methods such as k-means. Generally, anchors generated by k-means are more representative than random anchors. However, it is time-consuming and may generate unbalanced clusters. The Balanced K-means based Hierarchical K-means (BKHK) (Zhu, Nie, and Li 2017) hierarchically separates the data into two balanced clusters with k-means and obtains anchors by calculating the center of the sub-clusters. It has been successfully applied to accelerate the graph based methods including hashing (Li, Hu, and Nie 2017), clustering (Zhu, Nie, and Li 2017), dimensionality reduction (Nie, Zhu, and Li 2017), semi-supervised learning (Zhang et al. 2020), etc. In addition, it should be noted that in existing literatures on graph-based multi-view clustering, solving the label matrix in one step has not been well explored.

In this study, we propose a method termed Fast Multi-view Discrete Clustering (FMDC) with anchor graphs to address the aforementioned issues. We generate m ($m \ll n$) representative anchors more efficiently by replacing k-means in BKHK with k-means++ (Arthur and Vassilvitskii 2006), and separately construct anchor graph on each view to describe the geometric structure. By computing an automatically integrated similarity matrix, we directly optimize the primal spectral problem. The main contributions include:

- We propose FMDC, which generates anchors more efficiently, and directly calculates the discrete cluster assignment matrix.
- We automatically weigh different views to measure the diverse contributions, and compute a symmetric and doubly-stochastic aggregated similarity matrix.
- FMDC takes a linear computational complexity w.r.t. n. It is a major improvement over the quadratic time.
- Comprehensive experiments are conducted on benchmark datasets to verify the proposed model.

*Corresponding author.

Related Work

In this section, we will introduce the notations used throughout this paper and review some closely relevant works.

Notations

Let $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n]^T \in \mathbb{R}^{n\times d}$ denote the single-view dataset and $\mathbf{U} = [\mathbf{u}_1, \cdots, \mathbf{u}_m]^T \in \mathbb{R}^{m\times d}$ denote the generated anchors. $\mathbf{x}_i \in \mathbb{R}^{d\times 1}$ $(1 \leqslant i \leqslant n)$ is the ith sample, $\mathbf{u}_j \in \mathbb{R}^{d\times 1}$ $(1 \leqslant j \leqslant m)$ is the jth anchor, n is the data number, m $(m \ll n)$ is the anchor number and d is the dimensionality. Denote the cluster assignment matrix as $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_n]^T \in \{0,1\}^{n\times c}$, where $\mathbf{y}_i \in \{0,1\}^{c\times 1}$ is the cluster assignment vector of $\mathbf{x}_i$, c is the cluster number. $y_{ij} = 1$ if $\mathbf{x}_i$ is assigned to the jth cluster; 0, otherwise. The ℓ_2-norm is $\|\cdot\|_2$. The Frobenius norm is $\|\cdot\|_F$. $\mathbf{I}$ is the identity matrix. $\mathbf{1}$ is a vector with all the elements as 1.

k-means++

The k-means algorithm randomly chooses cluster centers. k-means++ is a careful seeding k-means. It chooses the first center at random and select other centers by weighing the data points according to their distance squared from the closest center already chosen (Arthur and Vassilvitskii 2006). A data point farther from the current centers has a higher probability of being selected as the next center. k-means++ generally outperforms k-means in terms of both accuracy and speed.

Anchor Graph Construction

Constructing anchor graph is a well-researched problem (Liu, He, and Chang 2010). To improve the efficiency and performance, we adopt a parameter-free but effective neighbor assignment strategy (Nie et al. 2016b) to construct a sparse k-nn anchor graph $\mathbf{Z} = [\mathbf{z}_1, \cdots, \mathbf{z}_n]^T \in \mathbb{R}^{n\times m}$ between $\mathbf{X}$ and $\mathbf{U}$. According to (Nie et al. 2016b), the neighbor assignment for $\mathbf{x}_i$ can be modeled by solving

$$\min_{\mathbf{z}_i^T\mathbf{1}=1, \mathbf{z}_i \geqslant \mathbf{0}} \sum_{j=1}^{m} \|\mathbf{x}_i - \mathbf{u}_j\|_2^2 z_{ij} + \gamma(z_{ij})^2, \tag{1}$$

where z_{ij} encodes the similarity between $\mathbf{x}_i$ and $\mathbf{u}_j$, and γ is the regularization parameter. Following (Nie et al. 2016b), γ can be set as $\gamma = \frac{k}{2}\|\mathbf{x}_i - \mathbf{u}_{k+1}\|_2^2 - \frac{1}{2}\sum_{h=1}^{k}\|\mathbf{x}_i - \mathbf{u}_h\|_2^2$, and the solution of Eq. (1) is

$$z_{ij} = \begin{cases} \frac{\|\mathbf{x}_i - \mathbf{u}_{k+1}\|_2^2 - \|\mathbf{x}_i - \mathbf{u}_j\|_2^2}{k\|\mathbf{x}_i - \mathbf{u}_{k+1}\|_2^2 - \sum_{h=1}^{k}\|\mathbf{x}_i - \mathbf{u}_h\|_2^2} & j \leq k \\ 0 & j > k \end{cases}. \tag{2}$$

Constructing $\mathbf{Z}$ is extremely efficient because it only needs $O(ndm)$ to calculate distances. After obtaining $\mathbf{Z}$, a symmetric and doubly-stochastic similarity matrix $\mathbf{S}$ can be computed by (Liu, He, and Chang 2010)

$$\mathbf{S} = \mathbf{Z}\Delta^{-1}\mathbf{Z}^T, \tag{3}$$

where $\Delta \in \mathbb{R}^{m\times m}$ is a diagonal matrix and $\triangle_{jj} = \sum_{i=1}^{n} z_{ij}$ is the jth diagonal element of it.

Spectral Clustering Algorithm

Let $\mathcal{G}(\mathbf{X}, \mathbf{S})$ be a weighted undirected graph with dataset $\mathbf{X}$ and similarity matrix $\mathbf{S}$. The goal of graph cutting is to cut it into c unconnected subgraphs with nonempty subsets $\mathbf{X}_1, \cdots, \mathbf{X}_c$, satisfying $\mathbf{X}_l \cap \mathbf{X}_h = \emptyset$ and $\mathbf{X}_1 \cup, \cdots, \cup \mathbf{X}_c = \mathbf{X}$ $(1 \leqslant l, h \leqslant c)$. In graph cutting, there are two ways to measure the volume of $\mathbf{X}_l$

$$|\mathbf{X}_l| := \text{the number of entrties in } \mathbf{X}_l, \tag{4}$$

$$vol(\mathbf{X}_l) := \sum\nolimits_{i\in\mathbf{X}_l} d_{ii}, \tag{5}$$

where $d_{ii} = \sum_{j=1}^{n} s_{ij}$ is the degree of $\mathbf{x}_i$. To avoid the poor cutting caused by the minimum cut and ensure the subgraphs have balanced scales, there are two cutting methods: RatioCut (Rcut) (Hagen and Kahng 1992) and Normalized cut (Ncut) (Shi and Malik 1997) whose definitions are

$$Rcut(\mathbf{X}_1, \cdots, \mathbf{X}_c) = \sum_{l=1}^{c} \frac{cut(\mathbf{X}_l, \overline{\mathbf{X}}_l)}{|\mathbf{X}_l|}, \tag{6}$$

$$Ncut(\mathbf{X}_1, \cdots, \mathbf{X}_c) = \sum_{l=1}^{c} \frac{cut(\mathbf{X}_l, \overline{\mathbf{X}}_l)}{vol(\mathbf{X}_l)}, \tag{7}$$

where $\overline{\mathbf{X}}_l$ is the complement of $\mathbf{X}_l$, and $cut(\mathbf{X}_l, \overline{\mathbf{X}}_l) = \sum_{i\in\mathbf{X}_l, j\in\overline{\mathbf{X}}_l} s_{ij}$ is defined as the "cut" between $\mathbf{X}_l$ and $\overline{\mathbf{X}}_l$. Taking in $\sum_{i=1}^{c}\frac{cut(\mathbf{X}_i,\overline{\mathbf{X}}_i)}{|\mathbf{X}_i|} = \sum_{i=1}^{c}\frac{\mathbf{y}_l^T\mathbf{L}\mathbf{y}_l}{\mathbf{y}_l^T\mathbf{y}_l}$, Eq. (6) can be written in the following form of minimizing trace

$$\min_{\mathbf{Y}\in Ind} Tr((\mathbf{Y}^T\mathbf{Y})^{-1/2}\mathbf{Y}^T\mathbf{L}\mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1/2}), \tag{8}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{S}$ is the graph Laplacian matrix, $\mathbf{D}$ is the degree matrix with the ith element d_{ii}. Considering the computational difficulty of Eq. (8), the scaled indicator matrix

$$\mathbf{H} = [\mathbf{h}_1, \cdots, \mathbf{h}_c] = \mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1/2} \tag{9}$$

is introduced. The lth column of $\mathbf{H}$ is given by

$$\mathbf{h}_l = (0, \cdots, 0, \underbrace{1, \cdots, 1}_{|\mathbf{X}_l|}, 0, \cdots, 0)^T \Big/ \sqrt{|\mathbf{X}_l|}. \tag{10}$$

Noting $\mathbf{H}^T\mathbf{H} = \mathbf{I}$, the Rcut problem is relaxed as

$$\min_{\mathbf{H}=\mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1/2}} Tr((\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{L}\mathbf{H}). \tag{11}$$

In Ncut, $\mathbf{h}_l$ is defined the same as Eq. (10), except that $|\mathbf{X}_l|$ is replaced by $vol(\mathbf{X}_l)$. Now, $\mathbf{H}^T\mathbf{D}\mathbf{H} = \mathbf{I}$ and $\mathbf{h}_l^T\mathbf{L}\mathbf{h}_l = \frac{cut(\mathbf{X}_l,\overline{\mathbf{X}}_l)}{vol(\mathbf{X}_l)}$. Let $\mathbf{F} = \mathbf{D}^{1/2}\mathbf{H} = \mathbf{D}^{1/2}\mathbf{Y}(\mathbf{Y}^T\mathbf{D}\mathbf{Y})^{-1/2}$, the Ncut problem is relaxed as

$$\min_{\mathbf{F}=\mathbf{D}^{1/2}\mathbf{Y}(\mathbf{Y}^T\mathbf{D}\mathbf{Y})^{-1/2}} Tr((\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\tilde{\mathbf{L}}\mathbf{F}), \tag{12}$$

where $\tilde{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ is the normalized $\mathbf{L}$. The optimal solution of $\mathbf{H}$ or $\mathbf{F}$ is formed by the c eigenvectors of $\mathbf{L}$ or $\tilde{\mathbf{L}}$ corresponding to the c smallest eigenvalues. Finally, k-means is used to get the discrete cluster assignment matrix.

The Proposed Model

Formulation

In this section, we propose FMDC which performs clustering on anchor graphs and directly solves the primal spectral problem. We first generate anchors. Different from BKHK, we iteratively segment the raw data into two balanced clusters by adopting k-means++ instead of k-means to generate anchors more efficiently. For multiple views, considering that if we independently generate anchors on each view, different views will obtain distinct anchors, which leads to unreasonable results. Therefore, we use the concatenating features of all views to generate anchors, and then split the results into different views according to the dimensionality.

For the vth view ($1 \leq v \leq V$, V is the view number), we can obtain a symmetric and doubly-stochastic similarity matrix $\mathbf{S}^v = \mathbf{Z}^v(\Delta^v)^{-1}(\mathbf{Z}^v)^T$ with Eq. (3). To achieve the consistency of different views and measure their diverse contributions, we automatically assign coefficients to distinct views to express the correlation and complementarity between them. The problem is formulated as

$$\sum_{v=1}^{V} \alpha_v \mathbf{S}^v \quad s.t.\ \boldsymbol{\alpha}^T\mathbf{1} = 1, \alpha_v \geqslant 0, \tag{13}$$

where $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_V]^T \in \mathbb{R}^{V\times 1}$ is a non-negative normalized weight vector. $\sum_v \alpha_v \mathbf{S}^v$ can be regarded as an aggregated graph, which is symmetric and doubly-stochastic.

THEOREM 1. *Given the similarity matrix $\mathbf{S}^v$ calculated with Eq. (3), $\sum_v \alpha_v \mathbf{S}^v$ is symmetric and doubly-stochastic.*

Proof.

$$(\sum_{v=1}^{V} \alpha_v \mathbf{S}^v)^T = (\sum_{v=1}^{V} \alpha_v \mathbf{Z}^v(\Delta^v)^{-1}(\mathbf{Z}^v)^T)^T = \sum_{v=1}^{V} \alpha_v \mathbf{S}^v. \tag{14}$$

It verifies that $\sum_v \alpha_v \mathbf{S}^v$ is symmetric. Furthermore,

$$(\sum_{v=1}^{V} \alpha_v \mathbf{S}^v)\mathbf{1} = \sum_{v=1}^{V} \alpha_v \mathbf{Z}^v(\Delta^v)^{-1}(\mathbf{Z}^v)^T\mathbf{1} = \sum_{v=1}^{V} \alpha_v \mathbf{Z}^v\mathbf{1} = \mathbf{1}. \tag{15}$$

Accordingly, $\sum_v \alpha_v \mathbf{S}^v$ is doubly-stochastic. □

With above discussion, $\mathbf{L} = \mathbf{I} - \sum_v \alpha_v \mathbf{S}^v = \tilde{\mathbf{L}}$. Now, the solution of $\mathbf{H}$ in Eq. (11) equals to that of $\mathbf{F}$ in Eq. (12), which shows the unified viewpoint of Rcut and Ncut. In FMDC, we are committed to getting rid of the two-stage process, and directly solving the primal spectral problem to obtain a multi-view shared discrete cluster assignment matrix $\mathbf{Y}$. Therefore, we fulfill the idea as follows

$$\min_{\substack{\mathbf{Y}\in Ind,\\ \boldsymbol{\alpha}^T\mathbf{1}=1,\boldsymbol{\alpha}\geq\mathbf{0}}} Tr((\mathbf{Y}^T\mathbf{Y})^{-1}(\mathbf{Y}^T(\mathbf{I} - \sum_{v=1}^{V} \alpha_v \mathbf{S}^v)\mathbf{Y})). \tag{16}$$

The non-negative and discrete $\mathbf{Y}$ provides sufficient interpretability of the relationship between data points and clusters. Compared with the relaxed continuous value, Eq. (16) is much closer to Rcut and Ncut. However, Eq. (16) is difficult to compute and $\boldsymbol{\alpha}$ will obtain the trivial solution, i.e., only one single view will be active. Thus, we propose a new form that is easy to solve and smooths the weight distribution to obtain the non-trivial solution as follows

$$\min_{\mathbf{Y}\in Ind,\boldsymbol{\alpha}^T\mathbf{1}=1,\boldsymbol{\alpha}\geqslant\mathbf{0}} \left\| \sum_{v=1}^{V} \alpha_v \mathbf{S}^v - \mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T \right\|_F^2. \tag{17}$$

Algorithm 1 Algorithm to solve the problem in Eq. (17)

Input: A multi-view dataset $\{\mathbf{X}^1, \cdots, \mathbf{X}^V\}$, anchor number m, cluster number c.
Construct anchor graphs according to Eq. (2).
Initialize $\alpha_{(v)} = 1/V$ and $\mathbf{Y}$.
repeat
  Update $\mathbf{Y}$ according to Eq. (22).
  Update $\boldsymbol{\alpha}$ by using Algorithm 2.
until converge
Output: The cluster assignment matrix $\mathbf{Y}$.

THEOREM 2. *When $\boldsymbol{\alpha}$ is fixed, solving Eq. (17) is equivalent to solving Eq. (16).*

Proof. $Tr(\sum_v \alpha_v \mathbf{S}^v(\sum_v \alpha_v \mathbf{S}^v)^T)$ is a constant once the graphs are constructed and $\sum_v \alpha_v \mathbf{S}^v = (\sum_v \alpha_v \mathbf{S}^v)^T$. Besides, $Tr(\mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T) = c$. Consequently, we have

$$\begin{aligned} &\min_{\mathbf{Y}\in Ind} \left\| \sum_{v=1}^{V} \alpha_v \mathbf{S}^v - \mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T \right\|_F^2 \\ \Leftrightarrow &\min_{\mathbf{Y}\in Ind} Tr(-(\sum_{v=1}^{V} \alpha_v \mathbf{S}^v)\mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T) \\ \Leftrightarrow &\min_{\mathbf{Y}\in Ind} Tr((\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T(\mathbf{I} - (\sum_{v=1}^{V} \alpha_v \mathbf{S}^v))\mathbf{Y}) \end{aligned}. \tag{18}$$

This completes the proof. □

Optimization

We design an efficient two-step optimization algorithm which is summarized in Algorithm 1 to solve Eq. (17) iteratively.

Solving $\mathbf{Y}$ with fixed $\boldsymbol{\alpha}$. When $\boldsymbol{\alpha}$ is fixed, we have

$$\begin{aligned} &\min_{\mathbf{Y}\in Ind} \left\| \sum_{v=1}^{V} \alpha_v \mathbf{S}^v - \mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T \right\|_F^2 \\ \Leftrightarrow &\max_{\mathbf{Y}\in Ind} Tr((\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{S}\mathbf{Y}) \end{aligned}. \tag{19}$$

where $\mathbf{S} = \sum_v \alpha_v \mathbf{S}^v$. This problem can be written as

$$\max_{\mathbf{Y}\in Ind} \sum_{l=1}^{c} \frac{\mathbf{y}_l^T\mathbf{S}\mathbf{y}_l}{\mathbf{y}_l^T\mathbf{y}_l}. \tag{20}$$

Since $\mathbf{y}_l^T\mathbf{S}\mathbf{y}_l / \mathbf{y}_l^T\mathbf{y}_l$ involves all rows of $\mathbf{Y}$, we sequentially solve $\mathbf{Y}$ row by row and fix the other rows as constants. To determine the solution of the ith row $\mathbf{y}_i^T$, we need to compare the objective value when it goes from $[1, 0, \cdots, 0]$ to $[0, 0, \cdots, 1]$, and select the one with the largest objective function value as the optimal solution. Record $\mathbf{Y}$ in these c

cases as $\mathbf{Y}_{(1)}, \cdots, \mathbf{Y}_{(c)}$. In the hth ($1 \leq h \leq c$) case $\mathbf{Y}_{(h)}$, the hth element in the ith row is 1. The other rows of the c cases are identical. Define the objective vale of the hth case as $obj(\mathbf{Y}_{(h)})$, then, we need to solve $\max_h obj(\mathbf{Y}_{(h)})$.

Assume that in $\mathbf{Y}_{(0)}$, $\mathbf{y}_i^T$ is $[0, 0, \cdots, 0]$. Defining $\Delta(h) = obj(\mathbf{Y}_{(h)}) - obj(\mathbf{Y}_{(0)})$, solving problem $\max_h obj(\mathbf{Y}_{(h)})$ is equivalent to solving $\max_h \Delta(h)$:

$$\begin{aligned}\max_h \Delta(h) &\Leftrightarrow \max_h \sum_{l=1}^{c} \frac{(\mathbf{y}_l^{(h)})^T \mathbf{S} \mathbf{y}_l^{(h)}}{(\mathbf{y}_l^{(h)})^T \mathbf{y}_l^{(h)}} - \sum_{l=1}^{c} \frac{(\mathbf{y}_l^{(0)})^T \mathbf{S} \mathbf{y}_l^{(0)}}{(\mathbf{y}_l^{(0)})^T \mathbf{y}_l^{(0)}} \\ &\Leftrightarrow \max_h \frac{(\mathbf{y}_l^{(h)})^T \mathbf{S} \mathbf{y}_l^{(h)}}{(\mathbf{y}_l^{(h)})^T \mathbf{y}_l^{(h)}} - \frac{(\mathbf{y}_l^{(0)})^T \mathbf{S} \mathbf{y}_l^{(0)}}{(\mathbf{y}_l^{(0)})^T \mathbf{y}_l^{(0)}}\end{aligned} \tag{21}$$

Then, we obtain the optimal solution:

$$y_{ip} =< p = \arg\max_h \Delta(h) >, \tag{22}$$

where if the argument is true, $< \cdot >$ is 1, otherwise, it is 0.

Solving $\boldsymbol{\alpha}$ with fixed $\mathbf{Y}$. When $\mathbf{Y}$ is fixed, we have

$$\min_{\boldsymbol{\alpha}^T \mathbf{1}=1, \boldsymbol{\alpha} \geq \mathbf{0}} \left\| \sum_{v=1}^{V} \alpha_v \mathbf{S}^v - \mathbf{Y}(\mathbf{Y}^T \mathbf{Y})^{-1} \mathbf{Y}^T \right\|_F^2. \tag{23}$$

Expanding $\mathbf{S}^v$ and $\mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T$ to large vectors vec($\mathbf{S}^v$) and $\mathbf{y}$, Eq. (23) can be reformulated as

$$\begin{aligned}&\min_{\boldsymbol{\alpha}^T \mathbf{1}=1, \boldsymbol{\alpha} \geq \mathbf{0}} \left\| \tilde{\mathbf{S}} \boldsymbol{\alpha} - \mathbf{y} \right\|_F^2 \\ \Leftrightarrow &\min_{\boldsymbol{\alpha}^T \mathbf{1}=1, \boldsymbol{\alpha} \geq \mathbf{0}} Tr(\boldsymbol{\alpha}^T \tilde{\mathbf{S}}^T \tilde{\mathbf{S}} \boldsymbol{\alpha} - 2\boldsymbol{\alpha}^T \tilde{\mathbf{S}}^T \mathbf{y})\end{aligned}, \tag{24}$$

where $\tilde{\mathbf{S}} = [\text{vec}(\mathbf{S}^1), \cdots, \text{vec}(\mathbf{S}^V)]$. Assuming $\mathbf{M} = \tilde{\mathbf{S}}^T \tilde{\mathbf{S}}$ and $\mathbf{b} = 2\tilde{\mathbf{S}}^T \mathbf{y}$, Eq. (24) is equivalent to

$$\min_{\boldsymbol{\alpha}^T \mathbf{1}=1, \boldsymbol{\alpha} \geq \mathbf{0}} \boldsymbol{\alpha}^T \mathbf{M} \boldsymbol{\alpha} - \boldsymbol{\alpha}^T \mathbf{b}. \tag{25}$$

The Augmented Lagrangian Multiplier (ALM) method is used to solve this problem. Introducing an auxiliary variable $\boldsymbol{\beta}$ and let $\boldsymbol{\alpha} = \boldsymbol{\beta}$, Eq. (25) becomes

$$\min_{\boldsymbol{\alpha}^T \mathbf{1}=1, \boldsymbol{\alpha} \geq \mathbf{0}, \boldsymbol{\alpha}=\boldsymbol{\beta}} \boldsymbol{\alpha}^T \mathbf{M} \boldsymbol{\beta} - \boldsymbol{\alpha}^T \mathbf{b}. \tag{26}$$

We solve the following problem alternatively and iteratively

$$\min_{\boldsymbol{\alpha}^T \mathbf{1}=1, \boldsymbol{\alpha} \geq \mathbf{0}, \boldsymbol{\beta}} \boldsymbol{\alpha}^T \mathbf{M} \boldsymbol{\beta} - \boldsymbol{\alpha}^T \mathbf{b} + \frac{\mu}{2} \left\| \boldsymbol{\alpha} - \boldsymbol{\beta} + \frac{1}{\mu} \boldsymbol{\Lambda} \right\|_2^2, \tag{27}$$

where μ is the quadratic penalty parameter and $\boldsymbol{\Lambda}$ is the Lagrangian multiplier. Algorithm for solving $\boldsymbol{\alpha}$ is outlined in Algorithm 2 and the first two steps are as follows.

Step 1: Fix $\boldsymbol{\alpha}$, solve $\boldsymbol{\beta}$. The subproblem becomes

$$\min_{\boldsymbol{\beta}} \boldsymbol{\alpha}^T \mathbf{M} \boldsymbol{\beta} + \frac{\mu}{2} \left\| \boldsymbol{\alpha} - \boldsymbol{\beta} + \frac{1}{\mu} \boldsymbol{\Lambda} \right\|_2^2. \tag{28}$$

Vanishing the derivative of Eq. (28) w.r.t $\boldsymbol{\beta}$, we have

$$\boldsymbol{\beta} = \boldsymbol{\alpha} + \frac{1}{\mu}(\boldsymbol{\Lambda} - \mathbf{M}^T \boldsymbol{\alpha}). \tag{29}$$

Then, $\boldsymbol{\beta}$ is updated.

Algorithm 2 Algorithm to solve the problem in Eq. (25)

Initialization: Initialize $\boldsymbol{\alpha}$, $\boldsymbol{\Lambda}$, $\mu > 0$, $1 < \rho < 2$.
  repeat
    Update $\boldsymbol{\beta}$ by using Eq. (29).
    Update $\boldsymbol{\alpha}$ by using Eq. (33).
    Update $\boldsymbol{\Lambda}$ by $\boldsymbol{\Lambda} = \boldsymbol{\Lambda} + \mu(\boldsymbol{\alpha} - \boldsymbol{\beta})$.
    Update μ by $\mu = \rho\mu$.
  until converge
Output: Weight vector $\boldsymbol{\alpha}$.

Step 2: Fix $\boldsymbol{\beta}$, solve $\boldsymbol{\alpha}$. The subproblem becomes

$$\begin{aligned}&\min_{\boldsymbol{\alpha}^T \mathbf{1}=1, \boldsymbol{\alpha} \geq \mathbf{0}} \boldsymbol{\alpha}^T \mathbf{M} \boldsymbol{\beta} - \boldsymbol{\alpha}^T \mathbf{b} + \tfrac{\mu}{2} \left\| \boldsymbol{\alpha} - \boldsymbol{\beta} + \tfrac{1}{\mu} \boldsymbol{\Lambda} \right\|_2^2 \\ \Leftrightarrow &\min_{\boldsymbol{\alpha}^T \mathbf{1}=1, \boldsymbol{\alpha} \geq \mathbf{0}} \|\boldsymbol{\alpha} - \mathbf{r}\|_2^2\end{aligned} \tag{30}$$

where $\mathbf{r} = \frac{1}{\mu}(\mu\boldsymbol{\beta} - \boldsymbol{\Lambda} - \mathbf{M}\boldsymbol{\beta} + \mathbf{b})$. Eq. (30) can be solved with a closed-form solution. Its Lagrangian function is

$$L(\boldsymbol{\alpha}, \varphi, \boldsymbol{\zeta}) = \frac{1}{2} \|\boldsymbol{\alpha} - \mathbf{r}\|_2^2 - \varphi(\boldsymbol{\alpha}^T \mathbf{1} - 1) - \boldsymbol{\zeta}^T \boldsymbol{\alpha}, \tag{31}$$

where φ and $\boldsymbol{\zeta} \geq \mathbf{0}$ are the Lagrange multipliers. The optimal solution of $\boldsymbol{\alpha}$ should satisfy that the derivative of Eq. (31) w.r.t. $\boldsymbol{\alpha}$ is equal to zero, so we have

$$\boldsymbol{\alpha} - \mathbf{r} - \varphi\mathbf{1} - \boldsymbol{\zeta} = \mathbf{0}. \tag{32}$$

For the lth element of $\boldsymbol{\alpha}$, noting $\alpha_l \zeta_l = 0$ according to the KKT condition (Huang, Nie, and Huang 2015), we obtain

$$\alpha_l = (r_l + \varphi)_+. \tag{33}$$

Discussion

We end up this section by briefly analyzing the convergence and discussing the complexities.

Convergence analysis: The objective of Algorithm 1 is guaranteed to be monotonically decreased when solving one variable with the other fixed at each iteration. Besides, Eq. (17) is lower-bounded by 0. As a result, the algorithm can be guaranteed to be convergent.

Computational complexity: FMDC consists of three parts: (1) anchor generation, which needs $O(nd\log(m)t_1)$, where d is the total dimensionality of V views and t_1 is the iteration number of balanced k-means++; (2) anchor graph construction, which costs $O(nmd)$; (3) iterative optimization, which needs $O(nc)$ to update $\mathbf{Y}$ and $O(V)$ to solve $\boldsymbol{\alpha}$. Since V is

Dataset	n	V	c	$d_1/d_2 \cdots /d_V$
BBCSport	544	2	5	3183/3203
WebKB	1051	2	2	1840/3000
Reuters1	1500	5	6	21531/24892/34251/15506/11547
MNIST	10000	3	10	30/9/30
Reuters2	15000	5	6	21531/24892/34251/15506/11547
NUS	20721	5	24	65/226/145/74/129

Table 1: Overview of the adopted datasets.

Dataset	Cotrain	CoregSC	SwMC	MVGL	MLAN	OMSC	LMVSC	GMC	FMDC
					ACC				
BBCSport	0.7518	0.7243	0.6618	0.7004	0.7279	0.5581	0.4485	0.7390	**0.7657**
WebKB	0.8976	0.9581	0.7774	0.8259	0.7793	0.7831	0.8344	0.7764	**0.9657**
Reuters1	0.4160	0.4340	0.3247	0.3153	0.2860	OM	0.3660	0.3053	**0.4593**
MNIST	0.5845	0.5926	0.5760	0.7335	0.5886	0.5005	0.4669	**0.7437**	0.7389
Reuters2	0.4510	0.4251	0.2739	0.2877	0.3237	OM	0.4900	0.2839	**0.5108**
NUS	OM	0.1431	0.1328	0.1425	0.1635	OM	0.1420	OM	**0.1771**
					NMI				
BBCSport	0.6176	0.6467	0.5186	0.6551	0.6889	0.4122	0.1145	0.6047	**0.7657**
WebKB	0.3738	0.6760	0.0018	0.1314	0.0022	0.0056	0.1648	0.0010	**0.7259**
Reuters1	0.2227	0.2304	0.0524	0.0979	0.0807	OM	0.1482	0.0883	**0.2371**
MNIST	0.5112	0.5057	0.5817	0.6152	0.6168	0.3927	0.4391	0.6339	**0.6378**
Reuters2	0.2793	0.2386	0.0134	0.0384	0.0622	OM	0.3140	0.0321	**0.3226**
NUS-WIDE	OM	0.1166	0.0568	0.1132	0.0083	OM	0.0966	OM	**0.1269**
					purity				
BBCSport	0.7235	0.7511	0.6746	0.7210	0.7647	0.5643	0.7354	0.7529	**0.7714**
WebKB	0.8976	0.9581	0.7812	0.8259	0.7812	0.7831	0.8937	0.7812	**0.9657**
Reuters1	0.4604	0.4913	0.3307	0.3307	0.3134	OM	0.4501	0.3233	**0.5039**
MNIST	0.5761	0.5925	0.5782	0.7335	0.6302	0.5011	0.5987	**0.7437**	0.7392
Reuters2	0.5333	0.5199	0.2779	0.2943	0.3263	OM	0.5458	0.2927	**0.5567**
NUS	OM	0.2651	0.1390	0.2518	0.1664	OM	0.2476	OM	**0.2760**

Table 2: Clustering performance comparison of different methods on all datasets.

Dataset	Cotrain	CoregSC	SwMC	MVGL	MLAN	OMSC	LMVSC	GMC	FMDC
BBCSport	8.36	7.75	30.37	36.13	29.39	228.73	2.71	11.28	**0.31**
WebKB	14.76	10.99	41.11	8.35	2.54	127.85	3.87	2.21	**1.02**
Reuters1	80.64	34.25	215.26	133.38	19.98	OM	78.07	13.85	**6.96**
MNIST	6756.06	2599.33	96021.83	34901.54	1249.61	1143.28	3926.07	1268.18	**263.82**
Reuters2	35306.71	5889.54	16196.32	260553.93	6961.28	OM	2297.85	2529.91	**1537.12**
NUS	OM	13581.59	190479.31	196671.95	10190.42	OM	4286.32	OM	**1932.54**

Table 3: Running time (seconds) of different methods on all datasets (seconds).

usually negligible, the third part costs $O((nc)t_2)$, where t_2 is the iteration number. Considering $m \ll n$, $c \ll m$, and t_1 and t_2 are pretty small, FMDC approximately takes $O(nmd)$. It is much faster than the conventional graph-based method which needs $O(n^2d)$ as least.

Storage complexity: During the process, it needs to store $\mathbf{Z}^v$ and $\mathbf{Y}$. The storage complexity is $O(nmV + nc)$, which is much less than the quadratic complexity.

Experiments

In this section, we experimentally evaluate the proposed model. All the experiments are implemented on a Windows 10 desktop computer with a 3.6 GHz Intel Core i7-7700 CPU, 64 GB RAM and Matlab R2018b (64 bit).

Experimental settings

The proposed model is evaluated on five widely used benchmark datasets with different scales and cluster numbers: BBCSport (Greene and Cunningham 2006), WebKB (Sun and Chao 2013), Reuters[1], MNIST (LeCun et al. 1998) and NUS-WIDE (Chua et al. 2009). BBCSport, WebKB and Reuters exhibit high dimensionality. NUS-WIDE have large scale. Since certain approaches encounter a memory overflow problem when experimenting, we sample the first 24 clusters of NUS-WIDE namely NUS, and two different scales of Reuters namely Reuters1 and Reuters2 for comparative experiments. The details are shown in Table 1.

We compare FMDC with the classical single-view Spectral Clustering (SC) (Ng, Jordan, and Weiss 2001) and several graph-based multi-view methods: Graph-based Multi-view Clustering (GMC) (Wang, Yang, and Liu 2020), Large-scale Multi-View Subspace Clustering in linear time (LMVSC) (Kang et al. 2020), One-step Multi-view Spectral Clustering (OMSC) (Zhu et al. 2019), Multi-view Learning with Adaptive Neighbors (MLAN) (Nie et al. 2017a), Multi-view Clustering with Graph Learning (MVGL) (Zhan et al. 2017), Self-weighted Multiview Clustering (SwMC) (Nie et al. 2017b), Co-training approach (Cotrain) (Kumar and Daumé 2011), Coregularized multiview Spectral Clestering (CoregSC) (Kumar, Rai, and Daume 2011).

We download the source codes from the author's websites and follow the experimental settings just as reported in each paper. We empirically construct fifteen-nearest graphs. The anchors need to be sufficiently dense for effective adjacency

[1] http://archive.ics.uci.edu/ml

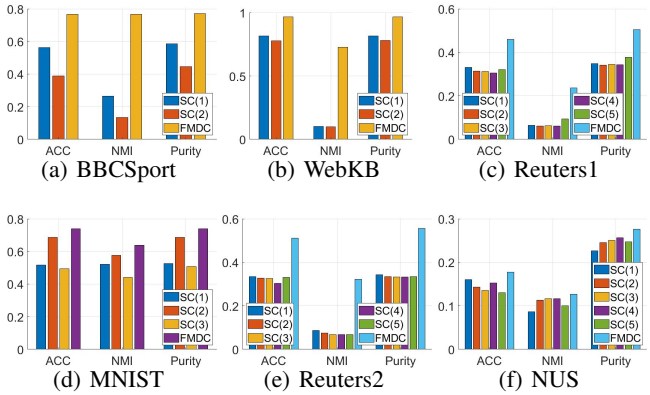

Figure 1: Comparison of FMDC and SC.

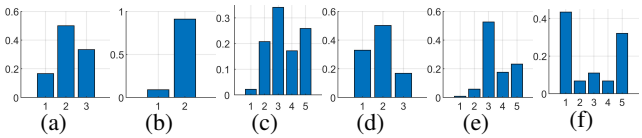

Figure 2: The learned weights for different views on: (a) BBCSport, (b) WebKB, (c) Reuters1, (d) MNIST, (e) Reuters2, (f) NUS.

relations, so we set anchor number m according to the scale in different datasets: m was set to 128 on BBCSports, WebKB and Reuters1, and 1024 on MIST, Reuters2 and NUS. The proposed model is not terminated until the difference of objective is less than $1\mathrm{e}{-10}$. All experiments are repeated tenfold and we report the average results and running time. For the results, we report three evaluation metrics: accuracy (ACC), normalized mutual information (NMI) and purity.

In the following subsection, we verify FMDC from five aspects: clustering performance, running time, learned weights, parameter sensitivity and convergence.

Clustering Results

Clustering performance and running time. Figure 1 shows the results of FMDC and SC on each view of all datasets. We observe that FMDC constantly outperforms the single-view method, verifying that combing information from different views can strengthen the performance. Tables 2 and 3 show the clustering results and running time, respectively. "OM": "Out-of-memory error" while running the experiment. The best performance is shown in bold face and the second-best is underlined. In general, FMDC achieves comparable or even better performance with less time cost than the other methods. The proposed method can greatly reduce the running time and needs less associated computational resources, especially in the large scale datasets. Accordingly, our model is a good choice for applications in real life.

Learned weights. The resultant weights learned by FMDC are plotted in Figure 2. The x-axis and y-axis represent view index and resultant weights, respectively. We observe that the

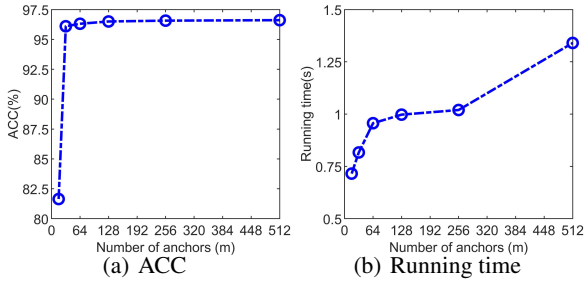

Figure 3: ACC and running time versus m.

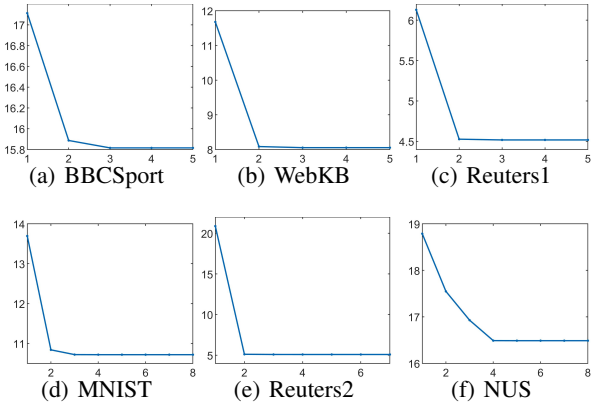

Figure 4: The objective values with iterations.

proposed model clearly weighs each view and simultaneously assigns a nonzero weight to each view. Despite different views own specific geometric structures and internal properties, FMDC is capable of well exploring the complementary or independent relations among them.

Parameter Sensitivity. The anchor number m needs to be tuned to obtain efficient performance. For simplicity, the WebKB dataset is used to analyze the influence of m on computational time and clustering accuracy. We conduct experiments by varying $m = \{16, 32, 64, 128, 512\}$. Figure 3 shows the experimental performance. As shown in Figure 3(a), a large m improves the clustering accuracy and the overly large m does not improve the performance. Figure 4(f) indicates that the running time increases as m increases. Based on the preceding analysis, it is necessary to select an appropriate m in practical applications to obtain enhanced clustering results and acceptable running time.

Convergence. Algorithm 1 is theoretically guaranteed to convergence. Here, we experimentally study the convergence rate of FMDC. We visualize the objective value after each iteration on all datasets. The results are recorded in Figure 4. The x-axis and y-axis represent iteration number and objective value, respectively. As observed, FMDC absolutely converges with few iterations.

Conclusion

In this paper, we propose a multi-view clustering model FMD-C. For each view, representative anchors are generated and an anchor graph is efficiently constructed. We automatically weigh different views in the clustering process. By performing clustering on the learned symmetric and doubly-stochastic aggregated similarity matrix, we directly solve the primal spectral problem and get the discrete cluster assignment matrix. The clustering process can be significantly accelerated and the computational complexity is linear to the data size. The proposed model is evaluated on benchmark datasets, and the the experimental results show its superiority, efficiency and practicality.

Acknowledgments

This work is supported by the National Major Science and Technology Projects of China 2019ZX01008103, the Fundamental Research Funds for the Central Universities and the National Natural Science Foundation of China under Grant 61772427, 61751202.

References

Arthur, D.; and Vassilvitskii, S. 2006. k-means++: The advantages of careful seeding. Technical report, Stanford.

Chen, W.; Song, Y.; Bai, H.; et al. 2010. Parallel spectral clustering in distributed systems. *IEEE transactions on pattern analysis and machine intelligence* 33(3): 568–586.

Chua, T.-S.; Tang, J.; Hong, R.; Li, H.; Luo, Z.; and Zheng, Y. 2009. NUS-WIDE: a real-world web image database from National University of Singapore. In *Proceedings of the ACM international conference on image and video retrieval*, 1–9.

Fowlkes, C.; Belongie, S.; Chung, F.; and Malik, J. 2004. Spectral grouping using the Nystrom method. *IEEE transactions on pattern analysis and machine intelligence* 26(2): 214–225.

Greene, D.; and Cunningham, P. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the 23rd international conference on Machine learning*, 377–384.

Hagen, L.; and Kahng, A. B. 1992. New spectral methods for ratio cut partitioning and clustering. *IEEE transactions on computer-aided design of integrated circuits and systems* 11(9): 1074–1085.

Huang, J.; Nie, F.; and Huang, H. 2015. A new simplex sparse learning model to measure data similarity for clustering. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Kang, Z.; Zhou, W.; Zhao, Z.; Shao, J.; Han, M.; and Xu, Z. 2020. Large-Scale Multi-View Subspace Clustering in Linear Time. In *AAAI*, 4412–4419.

Kumar, A.; and Daumé, H. 2011. A co-training approach for multi-view spectral clustering. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, 393–400.

Kumar, A.; Rai, P.; and Daume, H. 2011. Co-regularized multi-view spectral clustering. In *Advances in neural information processing systems*, 1413–1421.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11): 2278–2324.

Li, X.; Hu, D.; and Nie, F. 2017. Large graph hashing with spectral rotation. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Liu, W.; He, J.; and Chang, S.-F. 2010. Large graph construction for scalable semi-supervised learning. In *ICML*.

Ng, A.; Jordan, M.; and Weiss, Y. 2001. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems* 14: 849–856.

Nie, F.; Cai, G.; Li, J.; and Li, X. 2017a. Auto-weighted multi-view learning for image clustering and semi-supervised classification. *IEEE Transactions on Image Processing* 27(3): 1501–1511.

Nie, F.; Li, J.; Li, X.; et al. 2016a. Parameter-free auto-weighted multiple graph learning: A framework for multiview clustering and semi-supervised classification. In *IJCAI*, 1881–1887.

Nie, F.; Li, J.; Li, X.; et al. 2017b. Self-weighted Multiview Clustering with Multiple Graphs. In *IJCAI*, 2564–2570.

Nie, F.; Wang, X.; Jordan, M. I.; and Huang, H. 2016b. The constrained laplacian rank algorithm for graph-based clustering. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Nie, F.; Zhu, W.; and Li, X. 2017. Unsupervised large graph embedding. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2422–2428.

Shi, J.; and Malik, J. 1997. Normalized cuts and image segmentation. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, 731–737. IEEE.

Spielman, D. A.; and Teng, S.-H. 2011. Spectral sparsification of graphs. *SIAM Journal on Computing* 40(4): 981–1025.

Sun, S.; and Chao, G. 2013. Multi-view maximum entropy discrimination. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

Wang, H.; Yang, Y.; and Liu, B. 2020. GMC: Graph-Based Multi-View Clustering. *IEEE Transactions on Knowledge and Data Engineering* 32(6): 1116–1129.

Zhan, K.; Zhang, C.; Guan, J.; and Wang, J. 2017. Graph learning for multiview clustering. *IEEE transactions on cybernetics* 48(10): 2887–2895.

Zhang, B.; Qiang, Q.; Wang, F.; and Nie, F. 2020. Fast Multi-view Semi-supervised Learning with Learned Graph. *IEEE Transactions on Knowledge and Data Engineering* .

Zhou, P.; Shen, Y.-D.; Du, L.; Ye, F.; and Li, X. 2019. Incremental multi-view spectral clustering. *Knowledge-Based Systems* 174: 73–86.

Zhu, W.; Nie, F.; and Li, X. 2017. Fast spectral clustering with efficient large graph construction. In *2017 IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2492–2496. IEEE.

Zhu, X.; Zhang, S.; He, W.; et al. 2019. One-Step Multi-View Spectral Clustering. *IEEE Transactions on Knowledge and Data Engineering* 31(10): 2022–2034.