A Novel Normalized-Cut Solver with Nearest Neighbor Hierarchical Initialization

Feiping Nie, Jitao Lu, Danyang Wu, Rong Wang, and Xuelong Li

Abstract

Normalized-Cut (N-Cut) is a famous model of spectral clustering. The traditional N-Cut solvers are two-stage: 1) calculating the continuous spectral embedding of normalized Laplacian matrix; 2) discretization via K-means or spectral rotation. However, this paradigm brings two vital problems: 1) two-stage methods solve a relaxed version of the original problem, so they cannot obtain good solutions for the original N-Cut problem; 2) solving the relaxed problem requires eigenvalue decomposition, which has $\mathcal{O}(n^3)$ time complexity (n is the number of nodes). To address the problems, we propose a novel N-Cut solver designed based on the famous coordinate descent method. Since the vanilla coordinate descent method also has $\mathcal{O}(n^3)$ time complexity, we design various accelerating strategies to reduce the time complexity to $\mathcal{O}(|E|)$ ($|E|$ is the number of edges). To avoid reliance on random initialization which brings uncertainties to clustering, we propose an efficient initialization method that gives deterministic outputs. Extensive experiments on several benchmark datasets demonstrate that the proposed solver can obtain larger objective values of N-Cut, meanwhile achieving better clustering performance compared to traditional solvers.

Keywords: Coordinate descent method, Clustering, Graph cut.

1 Introduction

SPECTRAL clustering (SC) [32] partitions the nodes of a graph into disjoint clusters so that samples from the same cluster are more similar to samples from different clusters. The impacts of SC are far-reaching in clustering, and later graph-based clustering methods are deeply inspired by SC. The ancestor works of SC are Ratio-Cut (R-Cut) [9] and Normalized-Cut (N-Cut) [17, 28]. R-Cut firstly formulated clustering as a graph cut problem. It minimizes the ratio between inter-cluster similarities and cluster sizes to avoid the trivial solution of min-cut. Unlike R-Cut, N-Cut regularizes the clusters by their degrees to encourage more balanced clustering assignments and refines the objective as the ratio between inter-cluster similarities and cluster degrees. However, the original objectives of R-Cut and N-Cut are NP-hard to solve. Therefore, previous works follow a two-stage paradigm, which solves the relaxed versions of R-Cut and N-Cut and uncovers clustering results via K-means or spectral rotation [36].

Related works: In summary, the major issues of the two-stage paradigm are: 1. information loss due to relaxation of the original objective; 2. solving the relaxed objective as an eigenvalue problem is inefficient; 3. the discrete clustering labels could deviate far from the relaxed embeddings. Spectral rotation methods aim at addressing 3, while most existing works aim to improve the efficiency of 2, *i.e.*, speeding up the computation of

eigenvectors. [2, 16] uses power iteration to approximate the eigenvectors and thus avoid EVD. [5] sequentially subsamples the Laplacian matrix within the iterations of the EVD algorithm to reduce its cost. [35] applies SC to representative points and assigns their labels to associated samples. [8] subsamples the similarity matrix to accelerate EVD. [3] performs singular value decomposition on a landmark-based graph as a substitution for EVD. [31] generates node embeddings that approximate the pairwise distances of eigenvectors by filtering random graph signals. [34] uses Random Binning features to approximate the similarity matrix. Also, a number of methods are designed based on the objectives of SC. [21–23] combines spectral embedding learning with graph construction or graph approximation simultaneously. [26] proposed a representation learning method for multi-relational graphs with categorical node attributes. While efforts have been made to address the second and third issues, the first has hardly received any attention.

Unlike all the aforementioned methods, our work simultaneously addresses all 3 issues of the two-stage paradigm. We propose a fast heuristic solver to optimize the original objective of N-Cut directly *without any relaxations and approximations*. Hence, this work is orthogonal to SC variants that perform more approximations and we focus on comparison with the conventional solvers. Our solver, called Fast-CD, is designed based on the famous Coordinate Descent method. Via skillful equivalent transformations and accelerating techniques, the computational cost of Fast-CD is reduced to $\mathcal{O}(|E|)$ per iteration, which is more efficient than relaxed solvers based on EVD. To assist Fast-CD to converge to better local minimums without replicating clustering multiple times with distinct random initialization as the traditional solvers do, we propose an efficient Nearest Neighbor Hierarchical Initialization (N^2HI) method that leverages the first neighbor relations of the input graph. In experiments, we mainly compare the proposed Fast-CD solver with the traditional relaxed solver and spectral rotation. Besides, we also evaluate the clustering performance compared to some traditional and SOTA graph-based clustering models. From the results of several benchmarks, we observe that 1. the proposed Fast-CD solver can reach the solution with a lower objective value of the N-Cut model and can achieve better clustering performance; 2. in some benchmarks, the N-Cut model can achieve SOTA performance via the proposed Fast-CD solver. It is worthwhile summarizing the main contributions of this paper as follows:

- Different from previous works that solve the N-Cut model via relaxation, we propose a fast heuristic solver based on the coordinate descent method to optimize the objective problem directly with $\mathcal{O}(|E|)$ time complexity.
- The proposed Fast-CD solver is more efficient than relaxed solvers (based on EVD) in theory and practice.
- The proposed Fast-CD solver can achieve the solution with a higher objective value of N-Cut and better clustering performance relative to relaxed solvers.
- We propose an efficient initializer N^2HI to assist the Fast-CD solver. It outputs deterministic initial cluster assignments and thus effectively avoids randomness in the clustering procedure.

Notations: For any matrix $\mathbf{X}$, $\boldsymbol{x}_i$ denotes the i-th column, $\boldsymbol{x}^i$ denotes the i-th row, $\|\mathbf{X}\|_F$ denotes the Frobenius norm, $\mathrm{Tr}(\mathbf{X})$ denotes its trace, and $\mathrm{diag}(\mathbf{X})$ denotes its diagonal elements. n denotes the number of samples. c denotes the number of clusters. $\mathbf{A} \in \mathbb{R}^{n\times n}$ denotes the input similarity matrix, $\mathbf{D} \in \mathbb{R}^{n\times n}$ denotes its degree

matrix, and $\mathbf{L} = \mathbf{D} - \mathbf{A}$ denotes its Laplacian matrix. Ind denotes the set of cluster indicator matrices with $n \times c$ entries, so $\mathbf{Y} \in \text{Ind}$ implies $y_{ij} = 1$ if the i-th sample belongs to the j-th cluster, and $y_{ij} = 0$ if not.

2 Normalized-Cut Revisited

N-Cut defines the objective value as the sum of ratios between inter-cluster similarities and cluster degrees:

$$\min_{\mathbf{Y}\in\text{Ind}} \sum_{i=1}^{c} \frac{\boldsymbol{y}_i^T \mathbf{L} \boldsymbol{y}_i}{\boldsymbol{y}_i^T \mathbf{D} \boldsymbol{y}_i}, \tag{1}$$

which can be rewritten in matrix form as

$$\min_{\mathbf{Y}\in\text{Ind}} \text{Tr}\left((\mathbf{Y}^T\mathbf{D}\mathbf{Y})^{-\frac{1}{2}} \mathbf{Y}^T \mathbf{L} \mathbf{Y} (\mathbf{Y}^T\mathbf{D}\mathbf{Y})^{-\frac{1}{2}} \right). \tag{2}$$

However, the above equivalent problems are hard to optimize directly due to the discrete constraint of $\mathbf{Y}$. Hence, SC performs a two-step relaxed optimization for problem (2). Denoting $\mathbf{F} = \mathbf{Y}(\mathbf{Y}^T\mathbf{D}\mathbf{Y})^{-\frac{1}{2}}$, we have $\mathbf{F}^T\mathbf{D}\mathbf{F} = \mathbf{I}$. SC discards the discreteness condition of $\mathbf{F}$ and yields

$$\min_{\mathbf{F}^T\mathbf{D}\mathbf{F}=\mathbf{I}} \text{Tr}(\mathbf{F}^T\mathbf{L}\mathbf{F}) \xrightarrow{\mathbf{H}=\mathbf{D}^{\frac{1}{2}}\mathbf{F}} \min_{\mathbf{H}^T\mathbf{H}=\mathbf{I}} \text{Tr}(\mathbf{H}^T\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\mathbf{H}). \tag{3}$$

Ng *et al*. [17] applies EVD to the *symmetrically normalized Laplacian* $\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$ to solve $\mathbf{H}$, *i.e*., $\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\boldsymbol{h}_i = \lambda_i \boldsymbol{h}_i$. Alternatively, re-substituting $\boldsymbol{h}_i = \mathbf{D}^{\frac{1}{2}}\boldsymbol{f}_i$ to the eigenequation yields $\mathbf{D}^{-1}\mathbf{L}\boldsymbol{f}_i = \lambda_i \boldsymbol{f}_i$, so $\mathbf{F}$ can be obtained by applying EVD to the *random walk normalized Laplacian* $\mathbf{D}^{-1}\mathbf{L}$, or by solving the generalized eigenvalue system $\mathbf{L}\boldsymbol{f}_i = \lambda_i \mathbf{D}\boldsymbol{f}_i$ as proposed in [28]. Finally, the clustering label matrix $\mathbf{Y}$ can be generated from the learned embedding $\mathbf{F}$ in different ways. The most popular one is applying K-means clustering to $\mathbf{F}$ or $\mathbf{H}$ to obtain discrete labels. Apart from K-means, spectral rotation methods are also popular for label discretization. In [36], they first calculate an approximation of $\mathbf{Y}$ via $\mathbf{Y}_0 = \text{diag}\left(\text{diag}^{-\frac{1}{2}}(\mathbf{F}\mathbf{F}^T)\right)\mathbf{F}$, then uncover $\mathbf{Y}$ by solving the following problem:

$$\min_{\mathbf{Y}\in\text{Ind},\mathbf{R}^T\mathbf{R}=\mathbf{I}} \|\mathbf{Y} - \mathbf{Y}_0\mathbf{R}\|_F^2, \tag{4}$$

where $\mathbf{R} \in \mathbb{R}^{c\times c}$ is the rotation matrix. In [13], the authors compare K-means clustering with spectral rotations in theory and practice. Recently, [6] argues that solving $\mathbf{Y}$ based on the approximation $\mathbf{Y}_0$ affects the quality of the solution. To alleviate this problem, they propose to improve problem (4) as

$$\min_{\mathbf{Y}\in\text{Ind},\mathbf{R}^T\mathbf{R}=\mathbf{I}} \|\mathbf{D}^{\frac{1}{2}}\mathbf{Y}(\mathbf{Y}^T\mathbf{D}\mathbf{Y})^{-\frac{1}{2}} - \mathbf{H}\mathbf{R}\|_F^2. \tag{5}$$

In a word, previous works optimize problem (1) in a two-stage paradigm: 1. calculating an $n \times c$ continuous embedding $\mathbf{F}$ or $\mathbf{H}$; 2. learning $\mathbf{Y} \in \text{Ind}$ from them with different strategies. It is known that the two-stage paradigm generally decreases the quality of the solution compared to the direct paradigm. In this paper, we focus on optimizing the objective of N-Cut directly.

3 Fast Coordinate Descent N-Cut Solver

In this section, we are back to the vector-form (1) of N-Cut. Since $\mathbf{L} = \mathbf{D} - \mathbf{A}$, problem (1) is equivalent to

$$\max_{\mathbf{Y}\in\text{Ind}} \sum_{i=1}^{c} \frac{\boldsymbol{y}_i^T \mathbf{A} \boldsymbol{y}_i}{\boldsymbol{y}_i^T \mathbf{D} \boldsymbol{y}_i}. \tag{6}$$

Observing each row of $\mathbf{Y}$ being independent, we consider utilizing the coordinate descent method to directly optimize problem (6) by sequentially updating the rows. Suppose $\mathbf{Y}$ is the current solution and we are updating the m-th row with the rest $\{1, \ldots, m-1, m+1, \ldots, n\}$-th rows fixed, there are c alternative choices $\{\mathcal{Q}_c(k)\}_{k=1}^c$ where $\mathcal{Q}_c(k) \in \mathbb{R}^c$ is an one-hot row vector whose k-th element is 1, and we choose the one that maximizes Eq. (6). For convenience, $\mathcal{Q}_c(0)$ is the $\mathbb{R}^c$ vector with all elements being 0, and the matrix whose m-th row is $\mathcal{Q}_c(k)$ and other rows are same as $\mathbf{Y}$ is denoted by $\mathbf{Y}^{(k)}$. Then, the update of $\mathbf{Y}$'s m-th row can be formally expressed as

$$\hat{\boldsymbol{y}}^m = \mathcal{Q}_c \left(\arg\max_{k\in\{1,\ldots,c\}} \sum_{i=1}^{c} \frac{(\boldsymbol{y}_i^{(k)})^T \mathbf{A} \boldsymbol{y}_i^{(k)}}{(\boldsymbol{y}_i^{(k)})^T \mathbf{D} \boldsymbol{y}_i^{(k)}} \right), \tag{7}$$

where $\boldsymbol{y}_i^{(k)}$ is the i-th column of $\mathbf{Y}^{(k)}$, and $\hat{\boldsymbol{y}}^m$ stands for updated $\boldsymbol{y}^m$. However, directly calculating Eq. (7) is expensive because the time complexities of calculating $\{(\boldsymbol{y}_i^{(k)})^T \mathbf{A} \boldsymbol{y}_i^{(k)}\}_{k=1}^c$ and $\{(\boldsymbol{y}_i^{(k)})^T \mathbf{D} \boldsymbol{y}_i^{(k)}\}_{k=1}^c$ are $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$. Thus, the time complexity of updating all n rows adds up to $\mathcal{O}(n^3)$, which is the same as EVD.

3.1 Our Fast Implementation

To reduce the overhead, we design several strategies to accelerate the calculation of Eq. (7). At first, we observe that the calculations for $(\boldsymbol{y}_i^{(k)})^T \mathbf{A} \boldsymbol{y}_i^{(k)}$ and $(\boldsymbol{y}_i^{(k)})^T \mathbf{D} \boldsymbol{y}_i^{(k)}$ with different k are redundant since $\{\mathbf{Y}^{(k)}\}_{k=1}^c$ only differ in m-th row. Hence, we consider transforming Eq. (6) as follows:

$$\begin{aligned}
&\max_{k\in\{1,\ldots,c\}} \sum_{i=1}^{c} \frac{(\boldsymbol{y}_i^{(k)})^T \mathbf{A} \boldsymbol{y}_i^{(k)}}{(\boldsymbol{y}_i^{(k)})^T \mathbf{D} \boldsymbol{y}_i^{(k)}} \\
\overset{①}{\Longleftrightarrow} &\max_{k\in\{1,\ldots,c\}} \sum_{i=1}^{c} \left[\frac{(\boldsymbol{y}_i^{(k)})^T \mathbf{A} \boldsymbol{y}_i^{(k)}}{(\boldsymbol{y}_i^{(k)})^T \mathbf{D} \boldsymbol{y}_i^{(k)}} - \frac{(\boldsymbol{y}_i^{(0)})^T \mathbf{A} \boldsymbol{y}_i^{(0)}}{(\boldsymbol{y}_i^{(0)})^T \mathbf{D} \boldsymbol{y}_i^{(0)}} \right] \\
\overset{②}{\Longleftrightarrow} &\max_{k\in\{1,\ldots,c\}} \mathcal{L}(k) = \frac{(\boldsymbol{y}_k^{(k)})^T \mathbf{A} \boldsymbol{y}_k^{(k)}}{(\boldsymbol{y}_k^{(k)})^T \mathbf{D} \boldsymbol{y}_k^{(k)}} - \frac{(\boldsymbol{y}_k^{(0)})^T \mathbf{A} \boldsymbol{y}_k^{(0)}}{(\boldsymbol{y}_k^{(0)})^T \mathbf{D} \boldsymbol{y}_k^{(0)}},
\end{aligned} \tag{8}$$

where ① holds because $\frac{(\boldsymbol{y}_i^{(0)})^T \mathbf{A} \boldsymbol{y}_i^{(0)}}{(\boldsymbol{y}_i^{(0)})^T \mathbf{D} \boldsymbol{y}_i^{(0)}}$ is a constant, and ② holds because $\frac{(\boldsymbol{y}_i^{(k)})^T \mathbf{A} \boldsymbol{y}_i^{(k)}}{(\boldsymbol{y}_i^{(k)})^T \mathbf{D} \boldsymbol{y}_i^{(k)}} = \frac{(\boldsymbol{y}_i^{(0)})^T \mathbf{A} \boldsymbol{y}_i^{(0)}}{(\boldsymbol{y}_i^{(0)})^T \mathbf{D} \boldsymbol{y}_i^{(0)}}, \forall i \neq k$. After the above transformations, we avoided calculating $(\boldsymbol{y}_i^{(k)})^T \mathbf{A} \boldsymbol{y}_i^{(k)}$ and $(\boldsymbol{y}_i^{(k)})^T \mathbf{D} \boldsymbol{y}_i^{(k)}, \forall i \neq k$, and problem (7) is simplified as

$$\hat{\boldsymbol{y}}^m = \mathcal{Q}_c \left(\arg\max_{k\in\{1,\ldots,c\}} \frac{(\boldsymbol{y}_k^{(k)})^T \mathbf{A} \boldsymbol{y}_k^{(k)}}{(\boldsymbol{y}_k^{(k)})^T \mathbf{D} \boldsymbol{y}_k^{(k)}} - \frac{(\boldsymbol{y}_k^{(0)})^T \mathbf{A} \boldsymbol{y}_k^{(0)}}{(\boldsymbol{y}_k^{(0)})^T \mathbf{D} \boldsymbol{y}_k^{(0)}} \right). \tag{9}$$

The next issue is how to efficiently calculate $\{(\boldsymbol{y}_k^{(k)})^T \mathbf{A} \boldsymbol{y}_k^{(k)}\}_{k=1}^c$ and $\{(\boldsymbol{y}_k^{(k)})^T \mathbf{D} \boldsymbol{y}_k^{(k)}\}_{k=1}^c$. To this goal, we first calculate $\boldsymbol{y}_k^T \mathbf{A} \boldsymbol{y}_k$, $\boldsymbol{y}_k^T \mathbf{D} \boldsymbol{y}_k$ and $\boldsymbol{y}_k^T \boldsymbol{a}_m$. Then, $(\boldsymbol{y}_k^{(k)})^T \mathbf{A} \boldsymbol{y}_k^{(k)}$ and $(\boldsymbol{y}_k^{(k)})^T \mathbf{D} \boldsymbol{y}_k^{(k)}$ can be inferred from them as the following two cases supposing the p-th element of $\boldsymbol{y}^m$ is 1:

1) If $k = p$, it is easy to observe that $\boldsymbol{y}_k^{(k)} = \boldsymbol{y}_k$ and $\boldsymbol{y}_k^{(0)} = \boldsymbol{y}_k - \mathcal{Q}_n(m)$, where $\mathcal{Q}_n(m) \in \mathbb{R}^n$ is an one-hot column vector whose m-th element is 1. Then the terms about $\boldsymbol{y}_k^{(k)}$ in Eq. (9) can be simply calculated as

$$\frac{(\boldsymbol{y}_k^{(k)})^T \mathbf{A} \boldsymbol{y}_k^{(k)}}{(\boldsymbol{y}_k^{(k)})^T \mathbf{D} \boldsymbol{y}_k^{(k)}} = \frac{\boldsymbol{y}_k^T \mathbf{A} \boldsymbol{y}_k}{\boldsymbol{y}_k^T \mathbf{D} \boldsymbol{y}_k}. \tag{10}$$

And the term about $\boldsymbol{y}_k^{(0)}$ in Eq. (9) can be calculated as

$$\begin{aligned} &\frac{(\boldsymbol{y}_k^{(0)})^T \mathbf{A} \boldsymbol{y}_k^{(0)}}{(\boldsymbol{y}_k^{(0)})^T \mathbf{D} \boldsymbol{y}_k^{(0)}} = \frac{(\boldsymbol{y}_k - \mathcal{Q}_n(m))^T \mathbf{A} (\boldsymbol{y}_k - \mathcal{Q}_n(m))}{(\boldsymbol{y}_k - \mathcal{Q}_n(m))^T \mathbf{D} (\boldsymbol{y}_k - \mathcal{Q}_n(m))} \\ =&\frac{\boldsymbol{y}_k^T \mathbf{A} \boldsymbol{y}_k - 2\boldsymbol{y}_k^T \boldsymbol{a}_m + a_{mm}}{\boldsymbol{y}_k^T \mathbf{D} \boldsymbol{y}_k - 2\boldsymbol{y}_k^T \boldsymbol{d}_m + d_{mm}} = \frac{\boldsymbol{y}_k^T \mathbf{A} \boldsymbol{y}_k - 2\boldsymbol{y}_k^T \boldsymbol{a}_m}{\boldsymbol{y}_k^T \mathbf{D} \boldsymbol{y}_k - d_{mm}}, \end{aligned} \tag{11}$$

where we eliminate a_{mm} in the last step because the diagonal elements of $\mathbf{A}$ are zeros. Hence, the objective value becomes

$$\mathcal{L}(k \mid k = p) = \frac{\boldsymbol{y}_k^T \mathbf{A} \boldsymbol{y}_k}{\boldsymbol{y}_k^T \mathbf{D} \boldsymbol{y}_k} - \frac{\boldsymbol{y}_k^T \mathbf{A} \boldsymbol{y}_k - 2\boldsymbol{y}_k^T \boldsymbol{a}_m}{\boldsymbol{y}_k^T \mathbf{D} \boldsymbol{y}_k - d_{mm}}. \tag{12}$$

The time complexity of calculating Eq. (12) is $\mathcal{O}(1)$ because all the terms on the right-hand side have been calculated before.

2) If $k \neq p$, we can observe that $\boldsymbol{y}_k^{(0)} = \boldsymbol{y}_k$ and $\boldsymbol{y}_k^{(k)} = \boldsymbol{y}_k + \mathcal{Q}_n(m)$. Then the terms about $\boldsymbol{y}_k^{(0)}$ in Eq. (9) can be simply calculated as

$$\frac{(\boldsymbol{y}_k^{(0)})^T \mathbf{A} \boldsymbol{y}_k^{(0)}}{(\boldsymbol{y}_k^{(0)})^T \mathbf{D} \boldsymbol{y}_k^{(0)}} = \frac{\boldsymbol{y}_k^T \mathbf{A} \boldsymbol{y}_k}{\boldsymbol{y}_k^T \mathbf{D} \boldsymbol{y}_k}. \tag{13}$$

And the term about $\boldsymbol{y}_k^{(k)}$ in Eq. (9) can be calculated as

$$\begin{aligned} &\frac{(\boldsymbol{y}_k^{(k)})^T \mathbf{A} \boldsymbol{y}_k^{(k)}}{(\boldsymbol{y}_k^{(k)})^T \mathbf{D} \boldsymbol{y}_k^{(k)}} = \frac{(\boldsymbol{y}_k + \mathcal{Q}_n(m))^T \mathbf{A} (\boldsymbol{y}_k + \mathcal{Q}_n(m))}{(\boldsymbol{y}_k + \mathcal{Q}_n(m))^T \mathbf{D} (\boldsymbol{y}_k + \mathcal{Q}_n(m))} \\ =&\frac{\boldsymbol{y}_k^T \mathbf{A} \boldsymbol{y}_k + 2\boldsymbol{y}_k^T \boldsymbol{a}_m + a_{mm}}{\boldsymbol{y}_k^T \mathbf{D} \boldsymbol{y}_k + 2\boldsymbol{y}_k^T \boldsymbol{d}_m + d_{mm}} = \frac{\boldsymbol{y}_k^T \mathbf{A} \boldsymbol{y}_k + 2\boldsymbol{y}_k^T \boldsymbol{a}_m}{\boldsymbol{y}_k^T \mathbf{D} \boldsymbol{y}_k + d_{mm}}. \end{aligned} \tag{14}$$

Then the objective value of Eq. (9) w.r.t. $k \neq p$ becomes

$$\mathcal{L}(k \mid k \neq p) = \frac{\boldsymbol{y}_k^T \mathbf{A} \boldsymbol{y}_k + 2\boldsymbol{y}_k^T \boldsymbol{a}_m}{\boldsymbol{y}_k^T \mathbf{D} \boldsymbol{y}_k + d_{mm}} - \frac{\boldsymbol{y}_k^T \mathbf{A} \boldsymbol{y}_k}{\boldsymbol{y}_k^T \mathbf{D} \boldsymbol{y}_k}. \tag{15}$$

The time complexity of calculating Eq. (15) is $\mathcal{O}(1)$ because all the terms on the right-hand side have been calculated before. Traversing k from 1 to c, the total time complexity is $\mathcal{O}(c)$. After obtaining all $\{\mathcal{L}(k)\}_{k=1}^c$, we update $\hat{\boldsymbol{y}}^m$ to the one that maximizes it.

Since Eqs. (12) and (15) are efficiently calculated based on previously known $\boldsymbol{y}_k^T \mathbf{A} \boldsymbol{y}_k$, $\boldsymbol{y}_k^T \mathbf{D} \boldsymbol{y}_k$ and $\boldsymbol{y}_k^T \boldsymbol{a}_m$, we need to refresh them before updating the next row of $\mathbf{Y}$. Suppose $r = \arg\max_k \mathcal{L}(k)$, the solution remains unchanged if $r = p$ (recall that p stands for the index of 1 prior to updating $\boldsymbol{y}^m$), so there's no need to update them. Otherwise, we consider the following two additional cases:

1) For variables involving $\boldsymbol{y}_r$, we have $\hat{\boldsymbol{y}}_r = \boldsymbol{y}_r^{(r)}$ so they can be calculated as

$$\begin{aligned} &\hat{\boldsymbol{y}}_r^T \mathbf{A} \hat{\boldsymbol{y}}_r = (\boldsymbol{y}_r^{(r)})^T \mathbf{A} \boldsymbol{y}_r^{(r)}, \quad \hat{\boldsymbol{y}}_r^T \mathbf{D} \hat{\boldsymbol{y}}_r = (\boldsymbol{y}_r^{(r)})^T \mathbf{D} \boldsymbol{y}_r^{(r)}, \\ &\hat{\boldsymbol{y}}_r^T \mathbf{A} = \boldsymbol{y}_r^T \mathbf{A} + \boldsymbol{a}^m. \end{aligned} \tag{16}$$

Algorithm 1 Fast-CD solver for N-Cut problem (8)

Input: $\mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{Y} \in \mathbb{R}^{n\times c}$

Calculate and store $\boldsymbol{y}_k^T\mathbf{A}$, $\mathbf{D}$, $\boldsymbol{y}_k^T\mathbf{A}\boldsymbol{y}_k$, $\boldsymbol{y}_k^T\mathbf{D}\boldsymbol{y}_k$, and $n_k = \boldsymbol{y}_k^T\mathbf{1}, \forall k \in \{1,\ldots,c\}$ **for** $t \leftarrow 1$ **to** T **do** `// outer iteration`

  for $m \leftarrow 1$ **to** n **do** `// inner iteration`

    $p \leftarrow$ the index of element 1 in $\boldsymbol{y}^m$ **if** $n_p = 1$ **then** `// Avoid empty cluster`

      continue

    Calculate $\mathcal{L}(k), \forall k$ via Eq. (12) or Eq. (15) $r \leftarrow \arg\max_{k\in\{1,\ldots,c\}} \mathcal{L}(k)$ **if** $r \neq p$ **then** `// Better solution found`

      Update $\boldsymbol{y}_r^T\mathbf{A}\boldsymbol{y}_r, \boldsymbol{y}_r^T\mathbf{D}\boldsymbol{y}_r, \boldsymbol{y}_r^T\mathbf{A}$ via Eq. (16) Update $\boldsymbol{y}_p^T\mathbf{A}\boldsymbol{y}_p, \boldsymbol{y}_p^T\mathbf{D}\boldsymbol{y}_p, \boldsymbol{y}_p^T\mathbf{A}$ via Eq. (17) $y_{mr} \leftarrow 1, y_{mp} \leftarrow 0, n_r \leftarrow n_r + 1, n_p \leftarrow n_p - 1$

Output: $\mathbf{Y}$

2) For variables involving $\boldsymbol{y}_p$, we have $\hat{\boldsymbol{y}}_p = \boldsymbol{y}_p^{(0)}$ then they can be calculated as

$$\begin{aligned} \hat{\boldsymbol{y}}_p^T\mathbf{A}\hat{\boldsymbol{y}}_p &= (\boldsymbol{y}_p^{(0)})^T\mathbf{A}\boldsymbol{y}_p^{(0)}, \quad \hat{\boldsymbol{y}}_p^T\mathbf{D}\hat{\boldsymbol{y}}_p = (\boldsymbol{y}_p^{(0)})^T\mathbf{D}\boldsymbol{y}_p^{(0)}, \\ \hat{\boldsymbol{y}}_p^T\mathbf{A} &= \boldsymbol{y}_p^T\mathbf{A} - \boldsymbol{a}^m. \end{aligned} \tag{17}$$

Obviously, calculating $\{\hat{\boldsymbol{y}}_k^T\mathbf{A}\hat{\boldsymbol{y}}_k, \hat{\boldsymbol{y}}_k^T\mathbf{D}\hat{\boldsymbol{y}}_k \mid k \in \{r,p\}\}$ does not introduce extra complexity since all the variables have been calculated in previous steps. Calculating $\{\hat{\boldsymbol{y}}_k^T\mathbf{A} \mid k \in \{r,p\}\}$ is only necessary after actually updating $\mathbf{Y}$ (*i.e.*, $r \neq p$). The whole procedure is summarized in Algorithm 1. We break the outer iteration when the increasing rate of objective value is lower than a threshold, which is 10^{-9} throughout the experiments.

4 Nearest Neighbor Hierarchical Initialization

As introduced in Section 2, K-means and spectral rotation are common discretization strategies for N-Cut. However, a good initialization is crucial for both of them to obtain satisfactory performance. To be specific, it's well-known that Lloyd's algorithm for K-means is sensitive to the initial guesses for the clustering centroids and often gets stuck in bad local minima. There's also no easy way to initialize the rotation matrix $\mathbf{R}$ for spectral rotation and the initial labels $\mathbf{Y}$ for our Fast-CD solver. A workaround is to replicate the process multiple times with distinct initial guesses and pick the one with the best objective value, but the shortcoming is obvious: it's computationally expensive and brings uncertainties to clustering.

4.1 Our Methodology

The nearest neighbor method is one of the most simple yet effective methods for classification. Each data sample is assigned to the class of its closest neighbor. The idea is also adopted by the assignment step of K-means clustering, where each data sample is assigned to its nearest cluster centroid. Recently, Sarfraz *et al*. proposed an efficient parameter-free method for data clustering using first neighbor relations [27] that iteratively merges data samples with their 1-nearest neighbor (1-nn) and generates a cluster hierarchy. Motivated by the simpleness and effectiveness of nearest neighbor methods, we propose a nearest neighbor hierarchical initialization (N^{2}HI) method for graph data. N^{2}HI is efficient and able to produce semantically meaningful

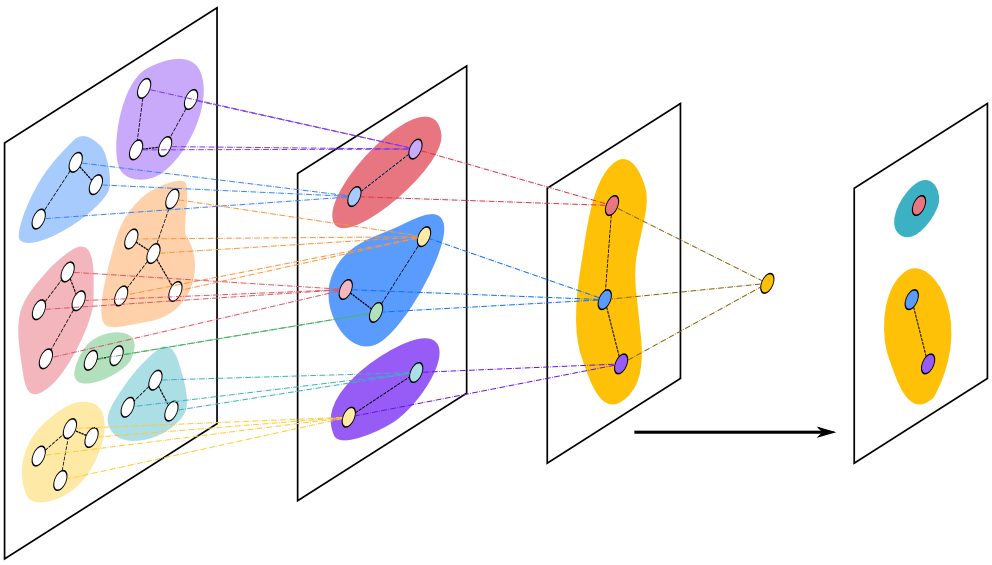


Figure 1. The overview of N^{2}HI. Black dashed lines within each level illustrate 1-nn relationships. Colored dashed lines across neighboring levels illustrate graph coarsening processes. A hierarchy of 7–3–1 clusters is first generated based on 1-nearest neighbors. The 3-cluster partition is later refined to obtain a 2-cluster partition.

cluster assignments, well coupled with our Fast-CD solver, and always gives deterministic outputs thus effectively avoiding uncertainties in clustering tasks. Concretely, N^{2}HI is composed of three steps as illustrated in Fig. 1.

a) Clustering by first-neighbor relations: We denote the input similarity graph at layer ℓ as $\mathbf{A}^{(\ell)} \in \mathbb{R}^{n_\ell \times n_\ell}$, the 1-nn of each data sample can be obtained by

$$u_i^{(\ell)} = \arg\max_i \boldsymbol{a}_i^{(\ell)}, \forall i \in \{1, \ldots, n_\ell\}. \tag{18}$$

Then, we obtain the partition at layer ℓ by assigning the i- and j-th samples to the same cluster if $i = u_j^{(\ell)}$ or $j = u_i^{(\ell)}$, *i.e.*, one is the 1-nn of the other. All data samples and their 1-nn form a bipartite graph, so the cluster assignments can be obtained by Tarjan's algorithm [30] in linear time to find its connected components. The input graph at the first layer is $\mathbf{A}^{(1)} = \mathbf{A}$.

b) Graph coarsening: After computing the cluster assignments, we merge data samples within the same cluster to obtain the input of the next layer. However, we have no access to the raw data samples in the context of graph-based clustering, so it's impossible to directly average them to obtain the representation after merging to compute new similarities. Instead, we propose to approximate the inter-cluster similarities of the next layer as

$$\mathbf{A}^{(\ell+1)} = \frac{1}{(\mathbf{Y}^{(\ell)})^T \mathbf{1}\mathbf{1}^T \mathbf{Y}^{(\ell)}} \circ (\mathbf{Y}^{(\ell)})^T \mathbf{A}^{(\ell)} \mathbf{Y}^{(\ell)}, \tag{19}$$

where $\mathbf{Y}^{(\ell)}$ denotes the cluster indicator matrix obtained by partitioning $\mathbf{A}^{(\ell)}$, $\mathbf{1}$ is an all-one vector with proper dimension, $\circ$ denotes the Hadamard product. We provide a 2-cluster example to facilitate interpreting how Eq. (19) works. Given two clusters $\mathcal{C}_1$, $\mathcal{C}_2$ associated with a similarity matrix $\mathbf{A}$ and cluster indicator matrix $\mathbf{Y}$, we propose to express the similarity between a sample $i \in \mathcal{C}_1$ and the whole $\mathcal{C}_2$ by the averaged similarity

between i and all $j \in \mathcal{C}_2$:

$$\hat{a}_{i,\mathcal{C}_2} = \frac{1}{|\mathcal{C}_2|} \sum_{j \in \mathcal{C}_2} a_{ij}, \tag{20}$$

and vice-versa for $\mathcal{C}_1$ and any $j \in \mathcal{C}_2$. Thus, the inter-cluster similarity between $\mathcal{C}_1$ and $\mathcal{C}_2$ is the averaged sum of pairwise similarities of all samples from the two clusters:

$$\hat{a}_{\mathcal{C}_1,\mathcal{C}_2} = \frac{1}{|\mathcal{C}_1| \cdot |\mathcal{C}_2|} \sum_{i \in \mathcal{C}_1, j \in \mathcal{C}_2} a_{ij} = \frac{\boldsymbol{y}_1^T \mathbf{A} \boldsymbol{y}_2}{\boldsymbol{y}_1^T \mathbf{1} \cdot \boldsymbol{y}_2^T \mathbf{1}}, \tag{21}$$

which is indeed the individual elements of Eq. (19). After obtaining $\mathbf{A}^{(\ell+1)}$, it's feed into Eq. (18) and the process is repeated until there's only one cluster. The output is a cluster hierarchy $\mathcal{T} = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_L \mid |\Gamma_\ell| > |\Gamma_{\ell+1}|, \forall \ell\}$, in which $\Gamma_\ell = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{|\Gamma_\ell|}\}$ are the sets of clusters.

c) Refinement: In order to obtain the clustering result with exact c clusters, we provide a mechanism to refine the hierarchy. If $\exists \Gamma_\ell \in \mathcal{T}, |\Gamma_\ell| = c$, the result is directly obtained without any subsequent steps. Otherwise, we find the Γ_ℓ from $\mathcal{T}$ that satisfying $|\Gamma_\ell| > c > |\Gamma_{\ell+1}|$, and refine it (essentially we can pick any Γ_ℓ as long as $|\Gamma_\ell| > c$). The refinement is composed of $|\Gamma_\ell| - c$ iterations, in which we subsequently merge one pair (u, v) with the largest similarity at a time and update the similarities by $a_{iu} \leftarrow (a_{iu} + a_{iv})/2, \forall i \notin \{u, v\}$. Obviously, c cannot be larger than the first partition $|\Gamma_1|$, but we empirically find that $|\Gamma_1|$ is *always* larger than the ground-truth c we desire.

5 Implementation details

5.1 Comparing with the released source codes

There is no released source code on github.We reproduced the code of the method in matlab according to the paper, reproduced most of the trials in the experiment, and achieved results close to those of the original paper.The self-tuning code used for constructing the similarity graph was obtained from a web search.The code for calculating a series of evaluation metrics is also referenced from web-based implementations.In addition to the experiments covered in the paper, we also compared with many classical or recent methods on a large number of other datasets.

5.2 Experimental environment setup

we conduct extensive experiments on different tasks to demonstrate the superiority of Fast-CD compared with other N-Cut solvers. In addition, we compare N-Cut with other popular graph-based clustering methods and discuss the trade-off of choosing the appropriate model for particular tasks. In order to validate the effectiveness of the proposed method, we conducted a number of experiments on real datasets.All experiments were conducted on a personal work station with Intel Xeon W-2275 CPU, 125G RAM.

If one of the methods in the experiment requires the construction of a similarity graph, we uniformly use the self-tuning method and set the number of k-nearest neighbors to 10.For high-dimensional datasets (greater than 100 dimensions),we traverse from 2 to the smaller of the number ofsamples and dimensions, and take half of this square off as the step size.For low-dimensional data, we project directly from 2 to dimension traversal

Table 1. Dataset statistics

Name	# Samples	# Features	# Classes
German	1000	20	2
GTdb	750	21600	50
MNIST10	6996	784	10
MSRA50	1799	1024	12
Segment	2310	19	7
STL-10	13000	2048	10
UMist	575	644	20
Waveform-21	2746	21	3

for each value.For the k-means algorithm in each method,we use randomly initialized clustering centers and set the number of iterations to 100.

Apart from the N-Cut objective value, we also employ three widely adopted metrics to quantify the clustering performance, including clustering accuracy (ACC), normalized mutual information (NMI) [29] and adjusted Rand index (ARI) [14].

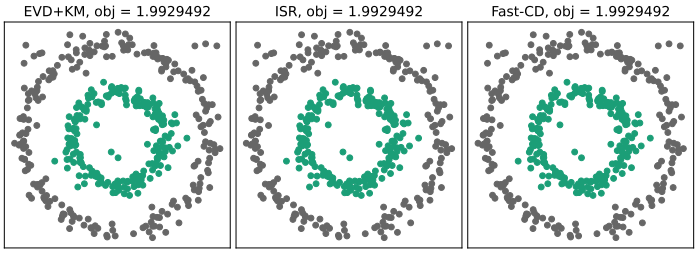
5.3 Dataset for Clustering

Table 2. Dataset for extra experiment

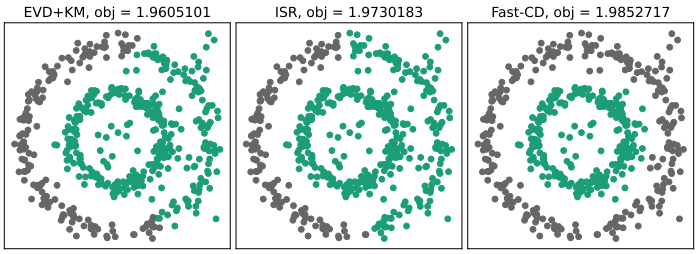
Dataset	**# Samples**	**# Features**	**# Classes**
Mpeg7_uni	1400	6000	70
COIL100	7200	1024	100
10kTrain	10000	784	10
binalpha	1404	320	36
ORL	400	2116	40
Jaffe	213	676	10
penbased	10992	16	10
optdigits	5620	64	10
texture	5500	40	11
semeion	1593	256	10
gas_sensor_array_drift	13910	128	6
Wholesale	440	6	2
Minist2k2k	4000	784	10
optdigit	5620	64	10
PalmData25	2000	256	100
facesOlivetti	400	10304	40

Userknowledge	403	5	4
uspst2007n	2007	256	10
pendigits	10992	16	10

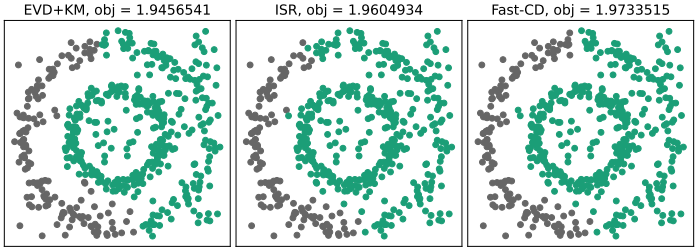
For classical graph-based clustering method, we employ 8 real-world benchmark datasets of varying numbers of samples and features that cover a wide range to evaluate the clustering performance. `German`, `Segment` and `Waveform-21` are grabbed from the UCI Machine Learning Repository [1]. `MNIST10` is a subset of the famous handwritten digit dataset [15]. `STL-10` is an object dataset [7] and we use the features extracted by a ResNet-50 [10] convolution neural network. Their statistics characters are summarized in Table 1. We employ the self-tuning spectral clustering strategy [37] to construct the initial graphs.Additionally, a series of experiments were conducted under a large dataset to demonstrate the general superiority of the approach presented in the paper.All these datasets' info are showen in Table 2.



(a) 50 noise points. All competitors correctly cluster two circles and converge to the same solution.



(b) 100 noise points. EVD+KM and ISR fail to cluster the outer circle, Fast-CD correctly clusters two circles and achieves the best N-Cut objective.



(c) 150 noise points. All competitors fail to cluster the outer circle, but Fast-CD still achieves the best N-Cut objective.

Figure 2. Clustering results for noisy 2-circle data. Each circle contains 200 points. Different numbers of extra uniform noise points are added.

Table 3. Objective values and clustering performance of different N-Cut solvers on 8 real-world benchmark datasets, the best results are in bold.

Algorithm	EVD+KM	ISR	Fast-CD	EVD+KM	ISR	Fast-CD	EVD+KM	ISR	Fast-CD	EVD+KM	ISR	Fast-CD
Dataset	German			GTdb			MNIST10			MSRA50		
obj	1.9918	1.9914	**1.9954**	15.3375	15.6509	**15.9180**	8.5232	8.5785	**8.6528**	11.6193	11.6190	**11.6259**
ACC	0.5420	0.5470	**0.6070**	0.5240	0.5347	**0.5480**	0.6954	0.5768	**0.7766**	**0.5397**	0.5331	0.5320
NMI	0.0026	0.0031	**0.0077**	0.7021	0.6873	**0.7056**	0.6506	0.5962	**0.7244**	**0.6933**	0.6805	0.6734
ARI	0.0054	0.0068	**0.0294**	**0.4060**	0.3645	0.4002	0.5346	0.4490	**0.6496**	**0.4498**	0.4362	0.4159
Dataset	Segment			STL-10			UMist			Waveform-21		
obj	6.8729	6.9204	**6.9272**	9.2781	9.2636	**9.3206**	12.4520	12.4131	**12.4825**	2.7477	2.7489	**2.7652**
ACC	**0.5043**	0.4983	0.4965	0.9312	0.8860	**0.9433**	0.4191	0.4191	**0.4730**	0.5215	0.5229	**0.5663**
NMI	**0.5088**	0.4971	0.4875	0.8783	0.8456	**0.8929**	0.6302	0.6205	**0.6765**	0.3706	0.3709	**0.3795**
ARI	**0.3738**	0.3583	0.3403	0.8574	0.7870	**0.8813**	0.3240	0.3166	**0.3668**	0.2530	0.2538	**0.2778**

6 Results and analysis

6.1 Synthetic Data

To demonstrate the importance of finding a more accurate solution, we follow [4] to evaluate the performance on the 2-circle dataset, where uniform noise is gradually added. The results are plotted in Fig. 2. We observe that Fast-CD consistently achieved the largest N-Cut objective (6) with different noisy levels. Especially, EVD+KM and ISR misclustered half of the outer circle in Fig. 2(b), while the solution with a larger objective value found by Fast-CD correctly clustered both circles.

6.2 Clustering with the Same Initialization

We run all comparison methods with the same initial cluster assignments and report the results in Table 3. The following observations are obtained from the results:

- Our proposed Fast-CD solver consistently obtained the largest objective value of N-Cut (Eq. (6)), which means that our solver is able to find a better local minimum compared with the ordinary EVD+KM solver and ISR.
- Fast-CD obtained the best clustering performance w.r.t. all three metrics on 5 out of 8 datasets. This indicates that achieving better solutions to the N-Cut problem is indeed beneficial for boosting the clustering performance.
- Fast-CD demonstrates its superb performance on `MNIST10` dataset, with **+0.07** of ACC, NMI, and ARI compared to the second-best method, which is a huge improvement.
- Despite larger objective values, both Fast-CD and ISR failed to surpass EVD+KM on `Segment` dataset. In addition, Fast-CD obtained a larger objective value than ISR but produced worse cluster assignments regarding ACC, NMI, and ARI. We argue that they belong to the rare cases where the N-Cut model

failed to capture the clustering relationships on certain datasets precisely, and thus better solutions of N-Cut lead to worse clustering performances. The issue is attributed to the design of the clustering model thus orthogonal to this work.

Table 4. Acc values compare with many method

Dataset	Kmeans	pca	lpp	TRPCA	CDKM	DBSC	DPC	FSDK	FastCD
Mpeg7_uni	42.69	53.74	**55.69**	42.88	52.41	14.43	47.82	47.96	53.5
COIL100	47.97	50.25	51.01	50.07	43.44	5.73	30.16	50.33	**68.69**
10kTrain	47.04	49.9	53.12	49.71	47.67	26.67	52.05	51.41	**59.59**
binalpha	40.5	44.05	44.93	41.29	43.75	16.03	43.14	42.66	**48.79**
ORL	60.08	66.52	64.6	56.35	68.28	25.47	68.6	62.1	**74**
Jaffe	76.15	86.01	84.18	77.51	82.3	62.96	**91.08**	87.32	**91.08**
penbased	70.22	71.71	72.37	68.55	68.21	28.79	72.48	77.39	**87.1**
optdigits	64.37	67.48	66.23	66.27	69.64	36.82	79.11	71.46	**87.33**
texture	56.39	62.56	62.8	63.17	61.1	21.25	**86.38**	70.29	84.51
semeion	55.05	63.47	63.35	59.33	59.03	37.86	65.6	60.77	**74.39**
gas_sensor_array_drift	36.04	36.99	38.14	39.97	36.98	27.11	**48.86**	40.04	48.61
Wholesale	73.14	76.95	80.11	81.16	76.14	85.41	64.07	77.16	**86.82**
Minist2k2k	44.79	47.68	46.94	47.32	46.32	30.24	**55.42**	50.2	54.17
optdigit	63.38	67.11	66.8	67.55	69.01	36.82	79.13	68.96	**87.33**
PalmData25	70.49	73.07	75.45	72.72	74.7	11.6	65.32	72.57	**89.65**
facesOlivetti	61.42	67.55	64.72	48.45	71.65	21.95	66.88	58.28	**74.5**
Userknowledge	42.26	44.86	45.58	43.33	44.29	44.27	44.42	47.89	**55.33**
uspst2007n	57.04	61.54	61.87	61.57	59.62	37.28	64.52	64.63	**75.98**
pendigits	69.28	72.7	72.41	68.12	69.1	28.28	72.52	75.53	**87.1**

6.3 Compare with Extra Clustering Methods

To compare this proposed method,several representative prototype-based clustering methods were involved, including k-means based on the coordinate descent method (CDKM) [25],discrete and balanced spectral clustering with scalability (DBSC) [33], discriminative projected clustering via Unsupervised LDA (DPC) [18], fast sparse discrim-inative k-means (FSDK) [20], a novel Normalized-Cut solver with nearest neighbor hierarchical initialization(Fast-CD) [19].To compare the effectiveness of the projection strategy in our proposed approach,we select a set of famous dimensionality reduction methods include PCA [12], truncated PCA [24], LPP [11], and ,k-means was utilized at the post stage of these methods (clustering).The average metrics for each method for ten runs of the same parameter for each dataset are collected.Bold in the table indicates the optimal accuracy under this dataset, while underlining indicates sub-optimal.

7 Conclusion and future work

This paper propose a novel, effective and efficient solver based on the famous coordinate descent method for Normalized-Cut, namely Fast-CD. Unlike spectral-based approaches of $\mathcal{O}(n^3)$ time complexity, the whole time complexity of our solver is just $\mathcal{O}(|E|)$. To avoid reliance on random initialization which brings uncertainties to clustering, an efficient initialization method that gives deterministic outputs is also designed. Extensive experiments strongly support the superiority of our proposal.

References

[1] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[2] Christos Boutsidis, Prabhanjan Kambadur, and Alex Gittens. Spectral clustering via the power method - provably. In *ICML*, pages 40–48, 2015.

[3] Deng Cai and Xinlei Chen. Large scale spectral clustering via landmark-based sparse representation. 45(8):1669–1680, 2015.

[4] Hong Chang and Dit-Yan Yeung. Robust path-based spectral clustering. *Pattern Recognit.*, 41(1):191–203, 2008.

[5] Bo Chen, Bin Gao, Tie-Yan Liu, Yu-Fu Chen, and Wei-Ying Ma. Fast spectral clustering of data using sequential matrix compression. In *ECML*, volume 4212, pages 590–597, 2006.

[6] Xiaojun Chen, Feiping Nie, Joshua Zhexue Huang, and Min Yang. Scalable normalized cut with improved spectral rotation. In *IJCAI*, pages 1518–1524, 2017.

[7] Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, pages 215–223, 2011.

[8] Charless C. Fowlkes, Serge J. Belongie, Fan R. K. Chung, and Jitendra Malik. Spectral grouping using the nyström method. 26(2):214–225, 2004.

[9] Lars Hagen and Andrew B Kahng. New spectral methods for ratio cut partitioning and clustering. 11(9):1074–1085, 1992.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[11] Xiaofei He and Partha Niyogi. Locality preserving projections. *Advances in neural information processing systems*, 16, 2003.

[12] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[13] Jin Huang, Feiping Nie, and Heng Huang. Spectral rotation versus k-means in spectral clustering. In *AAAI*, 2013.

[14] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. 86(11):2278–2324, 1998.

[16] Frank Lin and William W. Cohen. Power iteration clustering. In *ICML*, pages 655–662, 2010.

[17] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

[18] Feiping Nie, Xia Dong, Zhanxuan Hu, Rong Wang, and Xuelong Li. Discriminative projected clustering via unsupervised lda. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11):9466–9480, 2023.

[19] Feiping Nie, Jitao Lu, Danyang Wu, Rong Wang, and Xuelong Li. A novel normalized-cut solver with nearest neighbor hierarchical initialization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(1):659–666, 2024.

[20] Feiping Nie, Zhenyu Ma, Jingyu Wang, and Xuelong Li. Fast sparse discriminative k-means for unsupervised feature selection. *IEEE Transactions on Neural Networks and Learning Systems*, 35(7):9943–9957, 2024.

[21] Feiping Nie, Xiaoqian Wang, and Heng Huang. Clustering and projected clustering with adaptive neighbors. In *KDD*, pages 977–986, 2014.

[22] Feiping Nie, Xiaoqian Wang, Michael I. Jordan, and Heng Huang. The constrained laplacian rank algorithm for graph-based clustering. In *AAAI*, pages 1969–1976, 2016.

[23] Feiping Nie, Danyang Wu, Rong Wang, and Xuelong Li. Self-weighted clustering with adaptive neighbors. 31(9):3428–3441, 2020.

[24] Feiping Nie, Danyang Wu, Rong Wang, and Xuelong Li. Truncated robust principle component analysis with a general optimization framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(2):1081–1097, 2022.

[25] Feiping Nie, Jingjing Xue, Danyang Wu, Rong Wang, Hui Li, and Xuelong Li. Coordinate descent method for kk-means. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5):2371–2385, 2022.

[26] Ylli Sadikaj, Yllka Velaj, Sahar Behzadi, and Claudia Plant. Spectral clustering of attributed multi-relational graphs. In *KDD*, pages 1431–1440, 2021.

[27] M. Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *CVPR*, pages 8934–8943, 2019.

[28] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. 22(8):888–905, 2000.

[29] Alexander Strehl and Joydeep Ghosh. Cluster ensembles — A knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.*, 3:583–617, 2002.

[30] Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[31] Nicolas Tremblay, Gilles Puy, Rémi Gribonval, and Pierre Vandergheynst. Compressive spectral clustering. In *ICML*, pages 1002–1011, 2016.

[32] Ulrike von Luxburg. A tutorial on spectral clustering. *Stat. Comput.*, 17(4):395–416, 2007.

[33] Rong Wang, Huimin Chen, Yihang Lu, Qianrong Zhang, Feiping Nie, and Xuelong Li. Discrete and balanced spectral clustering with scalability. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(12):14321–14336, 2023.

[34] Lingfei Wu, Pin-Yu Chen, Ian En-Hsu Yen, Fangli Xu, Yinglong Xia, and Charu C. Aggarwal. Scalable spectral clustering using random binning features. In *KDD*, pages 2506–2515, 2018.

[35] Donghui Yan, Ling Huang, and Michael I. Jordan. Fast approximate spectral clustering. In *KDD*, pages 907–916, 2009.

[36] Stella X. Yu and Jianbo Shi. Multiclass spectral clustering. In *ICCV*, pages 313–319, 2003.

[37] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *NIPS*, pages 1601–1608, 2004.