Learning Distilled Graph for Large-Scale Social Network Data Clustering

Wenhe Liu, Dong Gong, Mingkui Tan, Javen Qinfeng Shi, Yi Yang, and Alexander G. Hauptmann

Abstract—Spectral analysis is critical in social network analysis. As a vital step of the spectral analysis, the graph construction in many existing works utilizes content data only. Unfortunately, the content data often consists of noisy, sparse, and redundant features, which makes the resulting graph unstable and unreliable. In practice, besides the content data, social network data also contain link information, which provides additional information for graph construction. Some of previous works utilize the link data. However, the link data is often incomplete, which makes the resulting graph incomplete. To address these issues, we propose a novel Distilled Graph Clustering (DGC) method. It pursuits a *distilled graph* based on both the content data and the link data. The proposed algorithm alternates between two steps: in the feature selection step, it finds the most representative feature subset w.r.t. an intermediate graph initialized with link data; in graph distillation step, the proposed method updates and refines the graph based on only the selected features. The final resulting graph, which is referred to as the distilled graph, is then utilized for spectral clustering on the large-scale social network data. Extensive experiments demonstrate the superiority of the proposed method.

Index Terms—Feature evaluation and selection, clustering, spectral analysis, social network

1 Introduction

In the last decade, we have witnessed the explosive growth of social networks. Every second, millions of photos, videos and texts are posted on social networking websites such as *Facebook* and *Twitter*, and we can also share these contents among a large number of users. Therefore, the social network analysis becomes more difficult when given large-scale data. In social network analysis, an important task is to discover the underlying communities [1] or the users' mutual interests [1]. This is often achieved by social network data clustering on the content data of the users (e.g., post tags of the users), based on the fact that users in the same community often have similar or correlated content information [2], [3]. Spectral analysis is an efficient solution to complete the social network data clustering, which analyzes social networks as graphs. Specifically, the users are treated as nodes, and the weighted edges in graphs are used to measure the relationships between users. As a result, we can obtain the affinity (similarity) information among nodes. In clustering, similar nodes will share similar labels. Such structure is consistent to the social influence theory, which indicates that users from the same group are more likely to have similar topics [4]. Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}^{\mathsf{T}} \in \mathbb{R}^{n \times d}$ be the content data of users. In spectral analysis, the similarity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$ can be computed via a Gaussian similarity function:

$$\mathbf{S}(i,j) = \begin{cases} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right), & \text{if } \mathbf{x}_i \in \mathcal{N}_k(\mathbf{x}_j) \text{ and } \mathbf{x}_j \in \mathcal{N}_k(\mathbf{x}_i) \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

where $\mathcal{N}_k(\mathbf{x}_j)$ denotes the set of k-nearest neighbors of $\mathbf{x}$, and σ is a scale parameter.

In most of the previous studies, the graph is often constructed using all features on content data, and will not be updated or improved after its construction. Unfortunately, in social network, the content data often contain a large number of unhelpful features. The complex activities of a large number of users may generate a large number of noisy, sparse, and redundant features (e.g., irrelevant tags for photos). As a result, graphs constructed based on all the features may cause significant bias on the true relationships of users. More critically, when the number of features becomes huge, the euclidean distance $d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|^2$ becomes meaningless. In this case, all the rest points tend to be the nearest neighbors of one point, which results in the "curse-of-dimensionality" issue [5]. To alleviate this issue, many previous methods resort to feature selection before clustering [6], [7], [8], [9], [10], [11]. However, in these methods, the feature selection step is independent to the graph construction step, which may neglect important information about the true structure.

- *W. Liu and Y. Yang are with the UTS-SUSTech Joint Centre of Computational Intelligence Systems, Southern University of Science and Technology, No.1088, Xueyuan Avenue, Shenzhen 518055, China, and also with the Centre for Artificial Intelligence, University of Technology Sydney, 15 Broadway, Ultimo, NSW 2007, Australia. E-mail: {allenlwh, yee.i.yang}@gmail.com.*
- *D. Gong and J.Q. Shi are with the School of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia. E-mail: edgong01@gmail.com, qinfeng.shi@ieee.org.*
- *M. Tan is with the School of Software Engineering, South China University of Technology, 381 Wushan Road, Tianhe District, Guangzhou 510630, P.R.China. E-mail: tanmingkui@gmail.com.*
- *A.G. Hauptmann is with the School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213. E-mail: alex+@cs.cmu.edu.*

Manuscript received 6 July 2017; revised 1 Dec. 2018; accepted 3 Mar. 2019. Date of publication 8 Mar. 2019; date of current version 3 June 2020. (Corresponding author: Mingkui Tan.) Recommended for acceptance by Y. Zhang. Digital Object Identifier no. 10.1109/TKDE.2019.2904068

Besides the content information, social network data also have link information (such as the "follows" or friend-connections). As mentioned in [4], linked users in a social network are more likely to have similar topics. Therefore, it is intuitively beneficial for community discovery. However, the link data is often incomplete or inaccurate. Given an unimaginably large number of users in a social network, one human user has no time to browse all the other users' website in the social network to determine their relationships precisely. Even worse, the link data may not always indicate the identifications of the community of persons. For example, a machine learning researcher may follow the blog of a physician, but they belong to different communities.

In this paper, we propose to use both the link data and the content data to construct the graph for social network data clustering. In the proposed method, we distill the graph iteratively. In each iteration, we identify the most representative features and update the graph in an alternating manner. First, we update the graph by using only the selected features. Then, given a constructed graph, we further seek the most representative features subset w.r.t. the graph. The finally obtained graph, referred to as *distilled graph*, is then used in spectral clustering. We denote the proposed algorithm as Distilled Graph Clustering (DGC).

In summary, we make the following contributions.

- We propose an iterative scheme to learn *distilled* graphs for large-scale social network data by exploiting both link information and content information. In our algorithm, distilled graph discovery and feature selection are conducted simultaneously. The proposed method can be further extended to learn graphs for semi-supervised and supervised learning settings.
- Given a fixed graph, we propose a novel algorithm to find the most representative feature subset. Instead of solving the problem with $\ell_{2,1}$-norm regularization, the proposed method addresses transfer the problem to a convex optimization problem with guaranteed performance and provides excellent convenience to control the number of selected features.
- Our algorithm is scalable for real-world large-scale social networks. We validate the proposed algorithm by comparing with other state-of-the-art algorithms on real-world data. Our method demonstrates competitive or better performances in the experiments and demonstrates outstanding efficiency, especially when the number of both links and/or features increasing.

2 Related Work

In this section, we review some of the most related works in the social network analysis, including feature selection and spectral analysis methods.

Feature selection technique has been widely used in social network analysis [12], [13]. In supervised feature selection, Tang et al. [1] extract different social relations to enhance the feature selection on link data. In semi-supervised feature selection [9], [10], there is a co-clustering framework presented to discover the communities of users in [9]. In unsupervised feature selection, there are methods directly utilizing content data for feature selection [6], [7], [11], and methods utilizing link data to enhance the feature selection performance [6], [8]. For instance, in [6], it introduces a social dimension structure learning method on link data to boost feature selection.

Spectral analysis algorithms aim to discover the local geometric structure of data via graph embedding [14], [15], [16], [17], [18]. In spectral analysis, similarity graph is constructed to capture the geometric relationship among instances. In the graph, instances within the same clusters are linked by edges with high similarity weights. The edges between instances from different clusters are assigned low similarity weights. Specifically, in spectral analysis for social network data clustering, different approaches have been applied for building the similarity graph. In several works, such as in [19], similarity graph is constructed with only content data. However, the content data may contain redundant and noisy features [8], leading to the unreliable and contaminated similarity measures. In other works [20], [21], [22], [23], the graph is constructed with only link data. In these works, link data is used to construct a pre-calculated graph for clustering. As a result, such methods are highly depend on the link information between every instances, which makes them not suitable for the real-world social network, where link information is not complete.

Spectral analysis has also be embedded with feature selection algorithms [24], [25], [26]. To preserve the data structure, some approaches [18], [27] proposed to learn the graph embedding along with feature selection. The algorithm in [28] performs subspace learning and feature selection jointly. In [29], with the pre-calculated similarity graph, discriminative features can be selected to preserve the local structure in low dimensional space. Method presented in [30] utilizes the content data via a linear regression formulation and adopts the link data through a spectral graph regularization term. Overall, all of these works treat the spectral graph construction and feature selection as two isolated steps. It will cause inaccurate results when link data and content data becomes more complex in large-scale scenario.

3 Spectral Analysis for Social Network Data

Notation. Let the superscript $^{\mathsf{T}}$ denote the transpose of a vector/matrix, superscript † denote the pseudo inverse of a matrix, $\mathbf{0}$ be a vector/matrix with all zeros, $\mathbf{I}$ be an identity matrix, $\mathrm{diag}(\mathbf{v})$ be a diagonal matrix with diagonal elements equal to $\mathbf{v}$, $\langle \mathbf{A}, \mathbf{B} \rangle = \mathrm{tr}(\mathbf{A}\mathbf{B}^{\mathsf{T}})$ be the inner product of $\mathbf{A}$ and $\mathbf{B}$, and $\|\mathbf{v}\|_p$ be the ℓ_p-norm of a vector $\mathbf{v}$. The Frobenius norm of $\mathbf{X}$ is defined as $\|\mathbf{A}\|_F = \sqrt{\langle \mathbf{A}, \mathbf{A} \rangle}$. For a sparse vector $\mathbf{x}$, let $\mathrm{support}(\mathbf{x}) = \{i | x_i \neq 0\} \in \{1, ..., m\}$ be its support. Let $\mathbf{A} \odot \mathbf{B}$ represent the element-wise product of two matrices $\mathbf{A}$ and $\mathbf{B}$. Lastly, for any convex function $\Omega(\mathbf{A})$, let $\partial\Omega(\mathbf{A})$ denote its subdifferential at $\mathbf{A}$. We define $\mathcal{G} = \{\mathcal{X}, \mathbf{A}\}$ as a weighted graph with a vertex set $\mathcal{X}$ and an affinity matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ constructed on $\mathcal{X}$. The (unnormalized) Laplacian matrix associated with $\mathcal{G}$ is defined as

$$\mathbf{L}_{\mathbf{A}} = \mathbf{D}_{\mathbf{A}} - \mathbf{A},$$

where $\mathbf{D_A}$ is a diagonal matrix with $\mathbf{D_A}(i,i) = \sum_j \mathbf{A}(i,j)$. To avoid the scale issue, in this paper, hereafter we use the normalized graph Laplacian matrix:

$$\mathbf{L_A} := \mathbf{D_A}^{-1/2}\mathbf{L_A}\mathbf{D_A}^{-1/2}. \tag{2}$$

Let $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}^\mathsf{T} \in \mathbb{R}^{n\times d}$ be the content data set with d features and n instances. In this work, we assume $\mathbf{X}$ is centered observation, i.e., $E(\mathbf{X}^\mathsf{T}) = \mathbf{0}$. In the social network scenario, each instance of $\mathbf{X}$ corresponds to a user. In this paper, the data are assumed in high dimensions, i.e., d is very large.

3.1 Affinity Information for Social Network Data

For social network data, beside the content information data $\mathbf{X}$, we may have additional link information. For example, we may have an additional link data matrix $\mathbf{R} \in \mathbb{R}^{n\times n}$ indicating the link states among users, where $\mathbf{R}(i,j) = 1$ if u_i and u_j are linked; otherwise 0. In this work, we focus on symmetrical relationships between users. We assume the relations between the users are symmetrical, i.e., $\mathbf{R} = \mathbf{R}^\mathsf{T}$. By default, we assume $\mathbf{R} = \mathbf{0}$ if no link information is provided. Let $\mathbf{L_S}$ and $\mathbf{L_R}$ be the Laplacian matrices built on $\mathbf{S}$ and $\mathbf{R}$ using Eq. (2). Considering both similarities based on content information $\mathbf{R}$ and link information $\mathbf{S}$, we simply construct the final Laplacian matrix as a liner combination of $\mathbf{L_S}$ and $\mathbf{L_R}$:

$$\mathbf{L} = \mathbf{L_S} + \alpha\mathbf{L_R}, \tag{3}$$

where α is a positive leverage parameter. The magnitudes of $\mathbf{L_S}$ and $\mathbf{L_R}$ are naturally comparable because they both describe the relations of the same user community.

3.2 Spectral Clustering

Once obtaining $\mathbf{L}$, we can apply the spectral clustering algorithm to partition the instances into c clusters. Let $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}^\mathsf{T} \in \mathbb{R}^{n\times c}$ be the cluster indicator vector of the c classes $\{1, \ldots, c\}$ for n instances, where $\mathbf{Y}(i,j) = 1$ if $\mathbf{x}_i$ is labeled as the jth cluster; otherwise 0. Note that, $\mathbf{Y}$ is a matrix with discrete values, which makes the optimization problem NP-hard [31]. We relax the indicator matrix by a scaled cluster indicator matrix $\mathbf{F} = \mathbf{Y}(\mathbf{Y}^\mathsf{T}\mathbf{Y})^{-\frac{1}{2}}$ to allow its entries to take on any real values. It is easy to verify that $\mathbf{F}^\mathsf{T}\mathbf{F} = \mathbf{I}$. Accordingly, the standard spectral clustering problem can be addressed by solving the following minimization problem:

$$\min_{\mathbf{F}} \operatorname{tr}(\mathbf{F}^\mathsf{T}\mathbf{L}\mathbf{F}), \quad \text{s.t.} \quad \mathbf{F}^\mathsf{T}\mathbf{F} = \mathbf{I}. \tag{4}$$

In problem Eq. (4), the optimal indicator matrix $\mathbf{F}$ is obtained to assign similar values to the instances that are close to each other on the graph represented by $\mathbf{L}$. This problem can be addressed by eigen-decomposition of $\mathbf{L}$ [14].

Algorithm 1. Spectral Clustering for Social Network Data

Input: Content matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, link matrix $\mathbf{R} \in \mathbb{R}^{n\times n}$, and leverage parameter α.

1: Compute the normalized matrix $\mathbf{L}$ according to Eq. (3).
2: Compute the first c eigenvectors (denoted by $\mathbf{U} \in \mathbb{R}^{n\times c}$) of $\mathbf{L}$.
3: Let $\mathbf{u}_i \in \mathbb{R}^c$ be the vector corresponding to the ith row of $\mathbf{U}$. Cluster the points $\{\mathbf{u}_i\}_{i=1}^n$ into clusters $\{1, ..., c\}$ via the k-means algorithm.

4 Learning Distilled Graph for Spectral Analysis

Previous works and the model in Eq. (4) calculating the the similarities using only the content information from scratch, in which the graph is usually constructed using all features. Once the graph is constructed, it will not be changed or improved. Unfortunately, in practice, the content information often contains many noisy, sparse, and redundant features. Using all the features may cause significant bias of the true linkages of users. To avoid these issues, we have to find a *distilled* graph that is constructed using only the true representative features.

4.1 General Method

Suppose we know the true representative features and index them by a 0-1 vector $\boldsymbol{\tau} \in \{0,1\}^d$. Here, $\tau_j = 1$ means the jth feature is a true feature; while $\tau_j = 0$ means the jth feature is a redundant/noisy feature. Then the ideal similarity matrix (or graph) can be constructed by

$$\mathbf{S}_{\boldsymbol{\tau}}(i,j) = \begin{cases} \exp\Big(-\frac{\|\boldsymbol{\rho}\odot(\mathbf{x}_i-\mathbf{x}_j)\|^2}{\sigma^2}\Big), & \text{if } \boldsymbol{\tau}\odot\mathbf{x}_i \in \mathcal{N}_k(\boldsymbol{\tau}\odot\mathbf{x}_j) \\ & \text{and } \boldsymbol{\tau}\odot\mathbf{x}_j \in \mathcal{N}_k(\boldsymbol{\tau}\odot\mathbf{x}_i) \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

and the ideal Laplacian matrix $\mathbf{L}(\boldsymbol{\tau})$ can be computed by

$$\mathbf{L}(\boldsymbol{\tau}) = \mathbf{L}_{\mathbf{S}_{\boldsymbol{\tau}}} + \alpha\mathbf{L_R}. \tag{6}$$

As shown in Eq. (6), once $\boldsymbol{\tau}$ is known, the graph is fixed. In practice, the ground truth of $\boldsymbol{\tau}$ is unknown. Suppose the number of true representative features satisfies $||\boldsymbol{\tau}||_0 \le k$, where k is some number representing the knowledge of the number of true features. An intuitive way to learn $\boldsymbol{\tau}$ is to address the following optimization problem:

$$\min_{\mathbf{F},\boldsymbol{\tau}} \operatorname{tr}(\mathbf{F}^\mathsf{T}\mathbf{L}(\boldsymbol{\tau})\mathbf{F}), \text{ s.t. } \mathbf{F}^\mathsf{T}\mathbf{F} = \mathbf{I}, \ ||\boldsymbol{\tau}||_0 \le k. \tag{7}$$

However, the above problem is non-convex and NP-hard, since there are exponential number of elements in $\boldsymbol{\tau}$, and the values of the elements in $bftau$ are discrete. In Eq. (7), $\boldsymbol{\tau}$ can also been seen as an indicator for selecting useful features. Meanwhile, given a fixed $\boldsymbol{\tau}$, it is possible to find the most representative graph w.r.t. $\mathbf{L}(\boldsymbol{\tau})$. Therefore, We propose to solve the problem by alternately applying graph construction and feature selection.

Overall, starting from a reasonable initial guess of $\boldsymbol{\tau}$, the proposed algorithm gradually optimizes the graph distillation and the feature selection task in an alternating manner to obtain a distilled graph. Let Υ be the index sets of the features selected on graph with parameter $\boldsymbol{\tau}$, i.e., $\Upsilon = \operatorname{support}(\boldsymbol{\tau})$ or vice-versa. In each iteration, the algorithm first seeks the most representative features w.r.t. $\mathbf{L}(\boldsymbol{\tau})$ and records them into a feature index set $\mathcal{S}$. Second, the algorithm updates the feature set Υ by adding the new features $\mathcal{S}$ as $\Upsilon := \Upsilon \cup \mathcal{S}$. Third, the algorithm updates $\boldsymbol{\tau}$ and $\mathbf{L}(\boldsymbol{\tau})$ with the updated feature set Υ. The whole procedure of the proposed algorithm is summarized in Algorithm 2.

Algorithm 2 is very similar to the EM algorithm. First, the initialization of $\boldsymbol{\tau}$ is very important. Without loss of generality,

we can initialize $\boldsymbol{\tau} = \mathbf{0}$ and $\Upsilon = \emptyset$ since no feature is selected at the beginning. The graph is initialized as a fully-connected graph with $\mathbf{S}_{\boldsymbol{\tau}}(i,j) = 1$. This initialization is reasonable since it is very related to the principle component analysis when $\mathbf{R} = \mathbf{0}$. Second, it is easy to obtain the combined Laplacian matrix via Eq. (6). Note that, if $\mathbf{R} \neq \mathbf{0}$, we may start by assuming $\mathbf{S}_{\boldsymbol{\tau}}(i,j) = 0$ and $\mathbf{L}_{\mathbf{S}_{\boldsymbol{\tau}}} = \mathbf{0}$. In this case, we have $\mathbf{L}(\boldsymbol{\tau}) = \alpha\mathbf{R}$. The leverage parameter α can be optimized by grid research. Third, the remaining issue is how to find the most representative features w.r.t. a given $\mathbf{L}(\boldsymbol{\tau})$. In this stage, we aim to find the most representative features based on the graph. We transfer the spectral clustering task to a regularized least square problem. Then, we propose to solve the problem through a relaxed convex optimization method with a worst-case analysis strategy to gradually select the most representative features [32], [33], [34]. Here, the number of features k is introduced in the algorithm. In practice, the number of true features is unknown; thus k is unknown. We further relax the sparsity constraint $||\tau||_0 \leq k$ as $||\rho||_0 \leq \varrho < k$, where ϱ is a small integer which reflects a rough knowledge of k. In practice, ϱ can be very small (e.g., $\varrho = 1$). Due to the optimization scheme, we do not have to estimate k but approach it iteratively. We will discuss the details of our algorithm in the following sections.

Algorithm 2. Learning Distilled Graph for Spectral Clustering

Input: Content matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, link matrix $\mathbf{R} \in \mathbb{R}^{n\times n}$, k and leverage parameter α.
1: Initialize $\boldsymbol{\tau} = \mathbf{0}$ and selected feature subset $\Upsilon = \emptyset$.
2: Compute $\mathbf{L}(\boldsymbol{\tau})$ according to Eq. (6).
3: Find the most representative features $\mathcal{S}$ given $\mathbf{L}(\boldsymbol{\tau})$.
4: Update $\Upsilon := \Upsilon \cup \mathcal{S}$, and update $\boldsymbol{\tau}$ and $\mathbf{L}(\boldsymbol{\tau})$ accordingly.
5: Repeat Step 2-4 until k features are selected.

4.2 Finding the Most Representative Features w.r.t. $\mathbf{L}(\boldsymbol{\tau})$

To find the most the representative features w.r.t. $\mathbf{L}(\boldsymbol{\tau})$, we need some transformations. First, we assume that there is a feature mapping matrix $\mathbf{W} \in \mathbb{R}^{d\times c}$ that maps $\mathbf{X}$ to the pseudo-label $\mathbf{F}$, i.e., $\mathbf{F} = \mathbf{XW}$. Then, problem Eq. (4) can be rewritten as the following problem

$$\min_{\mathbf{W}} \ \mathrm{tr}(\mathbf{W}^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{L}(\boldsymbol{\tau})\mathbf{XW}), \quad \text{s.t.} \quad \mathbf{W}^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{XW} = \mathbf{I}. \tag{8}$$

Suppose $\mathbf{W}^*$ is an optimal solution to the above problem. In general, $\mathbf{W}^*$ is not sparse. However, a feature item with larger $||\mathbf{W}^*_{i\bullet}||_2$ is more important, where $\mathbf{W}^*_{i\bullet}$ denotes the ith row of $\mathbf{W}$. Problem Eq. (8) can be addressed via generalized eigenvalue decomposition [35], which, however, can be very expensive for large-scale and high dimensional problems. To reduce the computational burden for large-scale problems, Sun et al. [35] show that $\mathbf{W}^*$ in fact can be obtained by solving a least square regression problem $\min_{\mathbf{W}} \ \frac{1}{2}\|\mathbf{XW} - \mathbf{T}\|_F^2$, where $\mathbf{T}$ is the regression target that can be computed by Algorithm 3 (See more details in [35]).

In practice, to avoid the possible over-fitting and improve the robustness to noise, we introduce a regularizer $\lambda\|\mathbf{W}\|_F^2$ and obtain the following optimization problem

$$\min_{\mathbf{W}} \frac{1}{2}\|\mathbf{XW} - \mathbf{T}\|_F^2 + \lambda\|\mathbf{W}\|_F^2, \tag{9}$$

where $\lambda > 0$ is a regularization parameter. However, the regularizer $\lambda\|\mathbf{W}\|_F^2$ in Eq. (9) does not encourage sparse solution. To induce sparsity to $\mathbf{W}$ and identify the most representative features, we introduce a binary indicator vector $\boldsymbol{\rho} \in \{0,1\}^d$, whose entries are 1 for the selected feature and 0 otherwise. Let

$$\Pi = \{\boldsymbol{\rho} : \boldsymbol{\rho} \in \{0,1\}^d, \|\boldsymbol{\rho}\|_0 \leq \varrho < k\}$$

be the domain of $\boldsymbol{\rho}$, where integer $\varrho \ll d$ represents the least number of features to be selected. Finally, we formulate our objective function for learning the most representative features as follows

$$\min_{\boldsymbol{\rho}\in\Pi} \min_{\mathbf{W}} \frac{1}{2}\|\mathbf{E}\|_F^2 + \frac{\lambda}{2}\|\mathbf{W}\|_F^2 \ \text{s.t.} \ \mathbf{E} = \mathbf{T} - \mathbf{X}\mathrm{diag}(\boldsymbol{\rho})\mathbf{W}, \tag{10}$$

where $\mathrm{diag}(\boldsymbol{\rho})$ is a diagonal matrix whose diagonal is $\boldsymbol{\rho}$ and $\mathbf{E}$ is an auxiliary variable $\mathbf{E} = \mathbf{T} - \mathbf{X}\mathrm{diag}(\boldsymbol{\rho})\mathbf{W}$. In Eq. (10), there are $\sum_{i=1}^{\varrho}\binom{d}{i}$ feasible $\boldsymbol{\rho}$'s in Π (e.g., $|\Pi| = \sum_{i=1}^{\varrho}\binom{d}{i}$), and the feature selection task can be cast as finding the best $\boldsymbol{\rho}$ from Π and solving the inner minimization problem w.r.t. $\mathbf{W}$.

Algorithm 3. Computation of Regression Target $\mathbf{T}$

Input: Symmetric positive and semi-definite matrix $\mathbf{L}(\boldsymbol{\tau})$.
1: Decompose $\mathbf{L}(\boldsymbol{\tau}) = \mathbf{HH}^T$, where $\mathbf{H} \in \mathbb{R}^{n\times r}$.
2: Do QR-decomposition $\mathbf{HP} = \mathbf{QR}$ with a permutation matrix $\mathbf{P}$.
3: Do singular value decomposition $\mathbf{R} = \mathbf{U}_{\mathbf{R}}\boldsymbol{\Sigma}_{\mathbf{R}}\mathbf{V}_{\mathbf{R}}^{\mathsf{T}}$.
4: Compute $\mathbf{T} = \mathbf{QU}_{\mathbf{R}}$.
Output: Regression target $\mathbf{T}$.

Problem Eq. (10) is non-convex, and NP-hard to solve. However, following [36], [37], we can make a convex relaxation, and transform it into the following convex quadratically constrained linear programming (QCLP) problem.

4.3 Relaxation of the Inner Problem (10)

For any fixed $\boldsymbol{\rho} \in \Pi$, the inner problem of Eq. (10) can be represented as

$$\min_{\mathbf{W}} \ \frac{1}{2}\|\mathbf{E}\|_F^2 + \frac{\lambda}{2}\|\mathbf{W}\|_F^2 \ \text{ s.t. } \ \mathbf{E} = \mathbf{T} - \mathbf{X}\mathrm{diag}(\boldsymbol{\rho})\mathbf{W}, \tag{11}$$

which can be solved in its dual form. By introducing the dual variable $\boldsymbol{\Lambda} \in \mathbb{R}^{n\times r}$, we can obtain the Lagrangian function as

$$\begin{aligned}\mathcal{L}(\mathbf{W},\mathbf{E},\boldsymbol{\Lambda}) = &\frac{1}{2}\|\mathbf{E}\|_F^2 + \frac{\lambda}{2}\|\mathbf{W}\|_F^2 \\ &- \langle\ \boldsymbol{\Lambda},\ \mathbf{E} - \mathbf{T} + \mathbf{X}\mathrm{diag}(\boldsymbol{\rho})\mathbf{W}\ \rangle.\end{aligned} \tag{12}$$

By setting the derivatives of $\mathcal{L}(\mathbf{W},\mathbf{E},\boldsymbol{\Lambda})$ w.r.t. $\mathbf{W}$ and $\mathbf{E}$ to $\mathbf{0}$ respectively, we get

$$\mathbf{W} = \frac{1}{\lambda}\mathrm{diag}(\boldsymbol{\rho})\mathbf{X}^{\mathsf{T}}\boldsymbol{\Lambda}, \text{and } \mathbf{E} = \boldsymbol{\Lambda}. \tag{13}$$

Substituting Eq. (13) into Eq. (12), we obtain the dual form of Eq. (11) as

$$\max_{\mathbf{\Lambda}\in\mathbb{R}^{n\times r}} -\frac{1}{2}tr\left(\mathbf{\Lambda}^{\mathsf{T}}\left(\frac{1}{\lambda}\mathbf{X}\mathrm{diag}(\boldsymbol{\rho})\mathbf{X}^{\mathsf{T}}+\mathbf{I}\right)\mathbf{\Lambda}\right)+tr(\mathbf{\Lambda}^{\mathsf{T}}\mathbf{T}).$$

For convenience, we define

$$f(\mathbf{\Lambda},\boldsymbol{\rho})=\frac{1}{2}tr\left(\mathbf{\Lambda}^{\mathsf{T}}\left(\frac{1}{\lambda}\mathbf{X}\mathrm{diag}(\boldsymbol{\rho})\mathbf{X}^{\mathsf{T}}+\mathbf{I}\right)\mathbf{\Lambda}\right)-tr(\mathbf{\Lambda}^{\mathsf{T}}\mathbf{T}),$$

which is a convex function. Following this, problem Eq. (10) can be equivalently reformulated as

$$\min_{\boldsymbol{\rho}\in\Pi}\max_{\mathbf{\Lambda}\in\mathbb{R}^{n\times r}} -f(\mathbf{\Lambda},\boldsymbol{\rho}). \tag{14}$$

Problem Eq. (14) is still a non-convex problem since it is a mixed integer programming problem. According to minimax inequality (5.46) in [38], we have:

$$\min_{\boldsymbol{\rho}\in\Pi}\max_{\mathbf{\Lambda}\in\mathbb{R}^{n\times r}} -f(\mathbf{\Lambda},\boldsymbol{\rho})\geq\max_{\mathbf{\Lambda}\in\mathbb{R}^{n\times r}}\min_{\boldsymbol{\rho}\in\Pi} -f(\mathbf{\Lambda},\boldsymbol{\rho}). \tag{15}$$

Accordingly, problem $\max_{\mathbf{\Lambda}\in\mathbb{R}^{n\times r}}\min_{\boldsymbol{\rho}\in\Pi}-f(\mathbf{\Lambda},\boldsymbol{\rho})$ is a convex lower bound of problem Eq. (14).

Recall that each $\boldsymbol{\rho}\in\Pi$ defines a quadratic constraint w.r.t. $\mathbf{\Lambda}$. Thus problem Eq. (15) has exponentially many constraints as there are $|\Pi|=\sum_{i=1}^{\varrho}\binom{d}{i}$, making it hard to address directly.

4.4 General Optimization for Problem (15)

Although there are exponentially many constraints in Eq. (15), only a few of them are active at the optimal solution, due to an assumption that the number of relevant features w.r.t $\mathbf{T}$ are small. Motivated by this observation and the cutting-plane method for handling problems with many constraints [32], [33], [39], we propose Algorithm 4 for problem Eq. (15).

Algorithm 4. Learning the Most Representative Features

Input: Input data $\mathbf{X}\in\mathbb{R}^{n\times d}$ and $\mathbf{L}(\boldsymbol{\tau})$.
1: Compute regression target $\mathbf{T}$ according to Algorithm 3.
2: Initialize $\Pi^0=\emptyset$, $\mathbf{\Lambda}^0=\mathbf{T}$, iteration index $t=1$.
3: Worst-case analysis: find the most violated $\boldsymbol{\rho}^t$ by solving Eq. (16).
4: Set $\Pi^t=\Pi^{t-1}\cup\{\boldsymbol{\rho}^t\}$.
5: Update $\mathbf{\Lambda}^t$ by solving the subproblem in Eq. (17) w.r.t. Π^t.
6: Let $t=t+1$. Repeat Step 3-5 until the stopping conditions are achieved.
Output: A set $\mathcal{S}$ indexing the selected features indicated by Π and recovered $\mathbf{W}$ (from solving the subproblem).

In Algorithm 4, we iteratively find the most-violated constraint and add to an active set, and then solve a subproblem associated with the active constraints. At the tth iteration, we find the most-violated constraint through the *worst-case analysis* in Step 2. Let $\mathbf{A}=\mathbf{X}^{\mathsf{T}}\mathbf{\Lambda}^{t-1}\in\mathbb{R}^{d\times r}$ where $\mathbf{\Lambda}^{t-1}=\mathbf{T}-\mathbf{X}\mathrm{diag}(\boldsymbol{\rho}^{t-1})\mathbf{W}^{t-1}$, and define $s_i=\sum_{j=1}^{r}(\mathbf{A}_{i,j})^2$. The worst-case analysis can be achieved by solving the following optimization problem:

$$\boldsymbol{\rho}^t=\arg\max_{\boldsymbol{\rho}\in\Pi} f(\mathbf{\Lambda}^{t-1},\boldsymbol{\rho})=\arg\max_{\boldsymbol{\rho}\in\Pi}\sum_{i=1}^{d}s_i\rho_i. \tag{16}$$

Problem Eq. (16) can be solved by first finding the ϱ largest s_i's, and then setting the corresponding ρ_i to 1 and the rests to 0.

After obtaining a new $\boldsymbol{\rho}^t$, we will add it into the active set $\Pi^t=\Pi^{t-1}\cup\{\boldsymbol{\rho}^t\}$, which is initialized as an empty set $\emptyset$, and then solve for $\mathbf{\Lambda}^t$ by addressing a subproblem w.r.t. constraints defined by Π^t:

$$\min_{\mathbf{\Lambda}\in\mathbb{R}^{n\times r},\,\theta\in\mathbb{R}}\quad\theta,\quad\text{s.t.}\;\; f(\mathbf{\Lambda},\boldsymbol{\rho})\leq\theta,\;\forall\boldsymbol{\rho}\in\Pi^t, \tag{17}$$

which can be efficiently solved in primal form.

4.5 Optimization for Subproblem (17)

Let $H=|\Pi^t|$ be the number of active constraints at tth iteration in Algorithm 4. Before achieving the stopping conditions, we solve the subproblem Eq. (17) w.r.t. $\mathbf{\Lambda}$ under the H constraints in Π^t. Although H is much smaller than $|\Pi|$ (i.e., the number of constrains) in problem Eq. (15), solving Eq. (17) is computationally expensive when n is vary large. Recall that after t iterations, Π^t contains at most $H\varrho$ features, where $H\varrho\ll n$. Motivated by this observation, we propose to solve Eq. (17) in the primal form w.r.t. $\mathbf{W}$.

Note that instead of maintaining all elements in $\mathbf{W}$ explicitly, we only need to handle the elements in $\mathbf{W}$ regarding the elements regarding the features recorded in Π^t. For each $\boldsymbol{\rho}_h\in\Pi^t$, where $h\in[H]$, we let $\mathbf{X}^h\in\mathbb{R}^{n\times\varrho}$ be the data set regarding features indicated by $\boldsymbol{\rho}_h$, $\mathbf{\Omega}_h\in\mathbb{R}^{\varrho\times r}$ >denote the weight matrix w.r.t. $\mathbf{X}^h$, and $\mathbf{\Omega}=[\mathbf{\Omega}_h^{\mathsf{T}}]_{h=1}^{H\mathsf{T}}\in\mathbb{R}^{\varrho H\times r}$ be a staked suppermatrix of all $\mathbf{\Omega}_h$'s. Then we have the following proposition.

Proposition 1. *For any $t>0$, the subproblem Eq. (17) can be equivalently addressed by solving its primal form:*

$$\min_{\mathbf{\Omega}}\;\frac{\lambda}{2}\left(\sum_{h=1}^{H}\|\mathbf{\Omega}_h\|_F\right)^2+\frac{1}{2}\|\mathbf{E}\|_F^2, \tag{18}$$

where $\mathbf{E}=\mathbf{T}-\sum_{h=1}^{H}\mathbf{X}^h\mathbf{\Omega}_h$ denotes the regression residual. Moreover, the dual variable $\mathbf{\Lambda}$ in Eq. (17) can be recovered by $\mathbf{\Lambda}=\mathbf{E}$ for the the worst-case analysis.

Proof. Let $g(\mathbf{\Omega})=\frac{\lambda}{2}(\sum_{h=1}^{H}\|\mathbf{\Omega}_h\|_F)^2$. Define a convex cone $\mathcal{Q}_\varrho=\{(\mathbf{\Omega},v)\in(\mathbb{R}^{\varrho\times r},\mathbb{R})\mid\|\mathbf{\Omega}\|_F\leq v\}$. Let $z_h=\|\mathbf{\Omega}_h\|_F$ and $z=\sum_{h=1}^{H}z_h$, we have $g(\mathbf{\Omega})=\frac{\lambda}{2}z^2$, where $z_h\geq0$ and $z\geq0$. Problem Eq. (17) can be reformulated as

$$\begin{aligned}\min_{\mathbf{\Omega},\mathbf{E},z}\;&\frac{\lambda}{2}z^2+\frac{1}{2}\|\mathbf{E}\|_F^2\\ \text{s.t. }&\mathbf{E}=\mathbf{T}-\sum_{h=1}^{H}\mathbf{X}^h\mathbf{\Omega}_h,\sum_{h=1}^{H}z_h\leq z,\;(\mathbf{\Omega}_h,z_h)\in\mathcal{Q}_\varrho,\forall h.\end{aligned} \tag{19}$$

Following [30], we obtain the dual problem:

$$\max_{\mathbf{\Lambda},\alpha}\;\;tr(\mathbf{\Lambda}^{\mathsf{T}}\mathbf{T})-\frac{1}{2}tr(\mathbf{\Lambda}^{\mathsf{T}}\mathbf{\Lambda})-\frac{1}{2\lambda}\alpha^2 \tag{20}$$

$$\text{s.t.}\;\;\|\mathbf{X}^{h\mathsf{T}}\mathbf{\Lambda}\|_F\leq\alpha,\;\;h=1,...,H. \tag{21}$$

Let $\theta=-tr(\mathbf{\Lambda}^{\mathsf{T}}\mathbf{T})+\frac{1}{2}tr(\mathbf{\Lambda}^{\mathsf{T}}\mathbf{\Lambda})+\frac{1}{2\lambda}\alpha^2$ and $f(\mathbf{\Lambda},\boldsymbol{\rho}_h)=\frac{1}{2\lambda}\|\mathbf{X}^{h\mathsf{T}}\mathbf{\Lambda}\|_F+\frac{1}{2}tr(\mathbf{\Lambda}^{\mathsf{T}}\mathbf{\Lambda})-tr(\mathbf{\Lambda}^{\mathsf{T}}\mathbf{T})$. Problem Eq. (20) becomes $\max_{\mathbf{\Lambda}\in\mathbb{R}^{n\times r},\theta\in\mathbb{R}}-\theta,\text{s.t.}\;\;f(\mathbf{\Lambda},\boldsymbol{\rho}_h)\leq\theta,h=1,...,H$, which is equivalent to problem Eq. (17). This completes the proof. □

In Eq. (18), the regularization term $\frac{\lambda}{2}(\sum_{h=1}^{H}\|\Omega_h\|_F)^2$ is non-smooth. We propose to solve it via an Accelerated Proximal Gradient (APG) method [40]. For convenience, we define $g(\Omega)=\frac{\lambda}{2}(\sum_{h=1}^{H}\|\Omega_h\|_F)^2$, $p(\Omega)=\frac{1}{2}\|\mathbf{T}-\sum_{h=1}^{H}\mathbf{X}^h\Omega_h\|_F^2$ and $\phi(\Omega)=g(\Omega)+p(\Omega)$. Given a point $\mathbf{V}\in\mathbb{R}^{\varrho H\times r}$, the APG method iteratively minimizes a quadratic approximation to $g(\Omega)+p(\Omega)$ w.r.t. Ω as following problem: $q(\Omega,\mathbf{V})=g(\Omega)+p(\mathbf{V})+\langle\nabla p(\mathbf{V}),(\Omega-\mathbf{V})\rangle+\frac{\tau}{2}\|\Omega-\mathbf{V}\|_F^2+p(\Omega)=\frac{\tau}{2}\|\Omega-\mathbf{G}\|_F^2+g(\Omega)+p(\mathbf{V})-\frac{1}{2\tau}\|\nabla p(\mathbf{V})\|_F^2$, where $\nabla p(\mathbf{V})$ denote the gradient of $p(\cdot)$ at point $\mathbf{V}$, $\tau>0$ denotes a guess of Lipschitz constant of $p(\cdot)$, and $\mathbf{G}=\mathbf{V}-\frac{1}{\tau}\nabla p(\mathbf{V})=[\mathbf{G}_h^{\mathsf{T}}]_{h=1}^{H\times\mathsf{T}}$, where $\mathbf{G}_h$ is the corresponding component to Ω_h. To minimize $q(\Omega,\mathbf{V})$ w.r.t. Ω, the problem reduces to the following Moreau projection problem [41]: $\min_\Omega\ \frac{\tau}{2}\|\Omega-\mathbf{G}\|_F^2+g(\Omega)$. Martins has shown that this problem has an unique closed-form solution, which is summarized in Proposition 2. The optimal solution can be obtained via Algorithm 5.

Proposition 2. *Let $\mathcal{S}_\tau(\mathbf{G},\mathbf{V})$ be the optimal solution to Moreau projection problem at $\mathbf{V}$, then $\mathcal{S}_\tau(\mathbf{G},\mathbf{V})$ is unique and can be calculated as follows:*

$$[\mathcal{S}_\tau(\mathbf{G},\mathbf{V})]_h=\begin{cases}\frac{o_h}{\|\mathbf{G}_h\|_F}\mathbf{G}_h, & if\ o_h>0,\\ \mathbf{0}, & otherwise,\end{cases}\tag{22}$$

where $[\mathcal{S}_\tau(\mathbf{G},\mathbf{V})]_h\in\mathbb{R}^{H\times r}$ denote the corresponding component w.r.t. $\mathbf{G}_h$ and $\mathbf{o}\in\mathbb{R}^H$ be an intermediate variable. Let $\widehat{\mathbf{o}}=[\|\mathbf{G}_h\|_F]_{h=1}^H\in\mathbb{R}^H$, the intermediate vector $\mathbf{o}$ can be obtained via a soft-threshold operator $o_h=[soft(\widehat{\mathbf{o}},\sigma)]_h=\begin{cases}\widehat{\mathbf{o}}-\sigma, & if\ \widehat{\mathbf{o}}>\sigma,\\ \mathbf{0}, & otherwise,\end{cases}$. Here the threshold σ is calculated in Step 4 of Algorithm 5.

Algorithm 5. Moreau Projection for $\mathcal{S}_\tau(\mathbf{G},\mathbf{V})$

Input: Matrix $\mathbf{G}$ and parameter $s=1/\tau>0$.
1: Calculate $\widehat{o}_h=\|\mathbf{G}_h\|_F$ for all $h\in[H]$.
2: Sort $\widehat{\mathbf{o}}$ into $\bar{\mathbf{o}}$ such that $\bar{o}_1>...>\bar{o}_h$.
3: Find $p=\max\{h|\bar{o}_h-\frac{s}{1+hs}\sum_{i=1}^{h}\hat{o}_i,h\in[H]\}$.
4: Calculate the threshold value $\sigma=\frac{s}{1+ps}\sum_{i=1}^{p}\hat{o}_i$.
5: Compute $\mathbf{o}=soft(\widehat{o},\sigma)$.
Output: $\mathcal{S}_\tau(\mathbf{G},\mathbf{V})$ via Eq. (22).

With the Moreau Projection in Algorithm 5, we present the APG method for problem Eq. (18) in Algorithm 6. In Algorithm 6 L_t is the Lipschitz constant w.r.t. $p(\cdot)$ in the tth iteration in Algorithm 4, which is adjusted by the line search. When a new set of features is added into the subproblem optimization in Algorithm 4, we can use the optimal solution of the last subproblem (Ω_{t-1}) as an initial guess to the corresponding components of the next subproblem.

4.6 Stopping Conditions of Algorithm 4

The stopping condition of Algorithm 4 is critical in our method. Since the intermediate graph is often not very accurate and the content data are often noisy, the target matrix $\mathbf{T}$ is not very accurate. Therefore, we should not solve the problem Eq. (15) very exactly; otherwise, the over-fitting problem will happen. To avoid this, we can stop the algorithm early with a relatively loose stopping condition:

$$\frac{|\theta^{t+1}-\theta^t|}{|\varrho\theta^0|}\le\varepsilon,\tag{23}$$

where ε is set to 0.1 in practice. Note that one may use the $\ell_{2,1}$-norm regularization to encourage sparsity, in which one can only choose a large regularization parameter to choose fewer features. However, if the regularization parameter is too large, bias issue would happen, which indicates the superiority of the proposed method.

Algorithm 6. APG for Solving Problem (18)

Input: .
1: Initialization: Initialize the Lipschitz constant $L_t=L_{t-1}$, set $\Omega^0=\mathbf{V}^1=[\Omega_t^{\mathsf{T}},\mathbf{0}^{\mathsf{T}}]^{\mathsf{T}}$ by warm start, $\tau_0=L_t$, $\eta\in(0,1)$, parameter $\varsigma^1=1$ and $k=1$.
2: Set $\tau=\eta\tau_{k-1}$.
For $j=0,1,...$
Set $\mathbf{G}=\mathbf{V}^k-\frac{1}{\tau}\nabla p(\mathbf{V}^k)$, and compute $\mathcal{S}_\tau(\mathbf{G},\mathbf{V}^k)$.
If $\phi(\mathcal{S}_\tau(\mathbf{G},\mathbf{V}^k))\le q(\mathcal{S}_\tau(\mathbf{G},\mathbf{V}^k),\mathbf{V}^k)$, Set $\tau_k=\tau$, stop;
Else $\tau=\min\{\eta^{-1}\tau,L_t\}$
End
End
3: Set $\Omega^k=\mathcal{S}_{\tau_k}(\mathbf{G},\mathbf{V}^k)$ and $L_k=\tau_k$. Go to Step 7 if the stopping condition achieves.
4: Set $\varsigma^k=\frac{1+\sqrt{1+4(\varsigma)^2}}{2}$.
5: Set $\mathbf{V}^{k+1}=\Omega^k+\frac{\varsigma^k-1}{\varsigma^{k+1}}(\Omega^k-\Omega^{k-1})$.
6: Let $k=k+1$ and go to Step 1.
7: Return and output $\Omega_t=\Omega^k$, and $L_t=\eta\tau_k$.

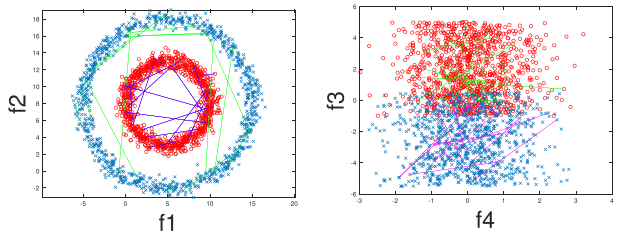
4.7 Computational Complexity

In Algorithm 1, let the input data be $\mathbf{X}\in\mathbb{R}^{n\times d}$, where n is the number of instances and d is the dimensionality of data. Overall, without the pursuit of distilled graph, the complexity of Algorithm 1 is dominated by the cost of building the Laplacian matrix in step 1, which is $\mathcal{O}(n^2d)$ [42].

Next, we analyze the complexity of our Algorithm 2.

- In step 3, we apply Algorithm 4 to find the most representative features. Let the matrix $\mathbf{L}(\boldsymbol{\tau})$ be with s significant singular values, where $s\le n$. The complexities of key operations in Algorithm 4 are as follows: 1) We apply Algorithm 3 to pursuit the regression target $\mathbf{T}$. In this subproblem, the complexities of decomposing $\mathbf{L}(\boldsymbol{\tau})=\mathbf{H}\mathbf{H}^T$ and QR-decomposition $\mathbf{HP}=\mathbf{QR}$ are both $\mathcal{O}(ns^2)$. The complexity of SVD of $\mathbf{R}\in\mathbb{R}^{s\times s}$ is $\mathcal{O}(s^3)$. 2) The complexity of worst-case analysis is $\mathcal{O}(nds+ds+m\log d)$. 3) We invite APG algorithm in Algorithm 6 to solve the subproblem in Eq. (17). The complexity of a standard APG algorithm is $\mathcal{O}(1/\sqrt{\eta})$ [40], with an η-optimal solution.
- In step 4, the complexity of computation of Laplacian matrix $\mathbf{L}(\boldsymbol{\tau})$ is $\mathcal{O}(n^2\rho^t)$, where ρ^t is the number of selected features in the tth iteration.

Overall, the complexity of Algorithm 2 is $\mathcal{O}(\sum^T(n^2\rho^t+nds))$, where T is the total iteration times. In social network analysis, on a high-dimensional data, we can evaluate $\rho^t\ll d$. Moreover, in both synthetic and real-world data experiments, the algorithm is stopped very early in ten iterations, which



(a) f1 w.r.t. f2 (b) f3 w.r.t. f4

Fig. 1. Illustration of synthetic samples (solid points) with positive links (solid lines) on different views of features.

implies T is small. Such that, Algorithm 2 is highly efficient compared with the conventional spectral analysis algorithms.

5 Experiments

5.1 Datasets

We evaluate the performance of various methods on both synthetic dataset and a collection of real-world datasets.

Synthetic Dataset. The synthetic dataset is a binary clustering dataset with three informative features $\mathbf{f}_1$, $\mathbf{f}_2$ and $\mathbf{f}_3$, namely $\mathbf{X}$. Inspired by [36], we draw the features as the following. As shown in Fig. 1, we draw the first two features $\mathbf{f}_1$ and $\mathbf{f}_2$ as composite features from a uniform distributed 2-circle dataset. We construct $\mathbf{f}_3$ as the most informative single feature [43]. Feature $\mathbf{f}_4$ and other features are noisy features drawn from a uniformly distribution. Ideally, a good feature selection algorithm should successfully identify the features $\mathbf{f}_1$ and $\mathbf{f}_2$ as a group, which is referred to as *ideal features* [36].

If the algorithm is forced to select only one feature, feature $\mathbf{f}_3$ should be selected. Clearly, if only considering one-feature spaces, neither $\mathbf{f}_1$-space nor $\mathbf{f}_2$-space is helpful for the clustering. However, if we can select more than one feature, the feature group of $\mathbf{f}_1$ and $\mathbf{f}_2$ is the best choice. As shown in Fig. 1b, clusters have a number of overlapped instances on the dimension of $\mathbf{f}_3$. As shown in Fig. 1a, clusters can be distinctly separated on the joint space of $\mathbf{f}_1$ and $\mathbf{f}_2$.

In our experiment, we additionally simulate a link matrix $\mathbf{R}$ as input data. Some sample pairs are selected as the linked samples within each cluster with the corresponding similarity assigned to 1 in $\mathbf{R}$. They are referred to as *positive links*, which reflect the correct geometric structures of the clusters (solid lines in Fig. 1. We also draw some sample pairs with samples from different clusters as linked, which are referred to as *noise links*.

BlogCatalog[1]: BlogCatalog is a social website of blogs. Each user of BlogCatalog is considered as a sample and tags added to blogs is considered as features. Link information is provided by "follows" relations between users. A user can also register under predefined categories, which are used as ground truth.

Flickr [2]: Flickr is a website for people to share their images. Similar as BlogCatalog, users are considered as samples, tags generated by users to their photos are used as features, "follows" relations are used as link information, pre-specified categories for photo collation are used as ground truth. The details of the real-world datasets are shown in Table 1.

1. https://www.blocatalog.com/
2. https://www.flickr.com/

TABLE 1
Information of the Real-World Datasets

	BlogCatalog	Flickr
# of Users	5,196	7,575
# of Features	8,189	12,047
# of Links	171,743	239,738
# of Ave Degree	66.11	63.30
# of Classes	6	9

5.2 General Experimental Settings

Two common evaluation metrics for clustering are used in our experiments, i.e., *accuracy* (ACC) and *normalized mutual information* (NMI) [8], [15], [44]. For both of the two measurements, the higher values indicate better performance.

NMI Let $\mathbf{C}$ denote the clustering labels from ground truth, and $\mathbf{C}'$ denote the predicted labels. The mutual information between $\mathbf{C}$ and $\mathbf{C}'$ is $MI(\mathbf{C},\mathbf{C}') = \sum_{c_i\in\mathbf{C},c'_j\in\mathbf{C}'} p(c_i,c'_j) log\frac{p(c_i,c'_j)}{p(c_i)p(c'_j)}$, where $p(c_i)$ and $p(c'_j)$ are the probabilities of instances in cluster c_i and c'_j, respectively, and $p(c_i,c'_j)$ denotes the probability of instances in cluster c_i and c'_j at the same time. The normalized mutual information is thus defined as:

$$NMI(\mathbf{C},\mathbf{C}') = \frac{MI(\mathbf{C},\mathbf{C}')}{\max(H(\mathbf{C}),H(\mathbf{C}'))}, \tag{24}$$

where $H(\mathbf{C})$ and $H(\mathbf{C}')$ represent the entropies of clusterings indicated by label $\mathbf{C}$ and $\mathbf{C}'$.

ACC Let p_i and q_j be the clustering label from the predicted result and ground truth for instance $\mathbf{x}_i$ respectively. The accuracy is thus defined as:

$$ACC = \frac{1}{n}\sum_{i=1}^{n}\delta(q_i, map(p_i)), \tag{25}$$

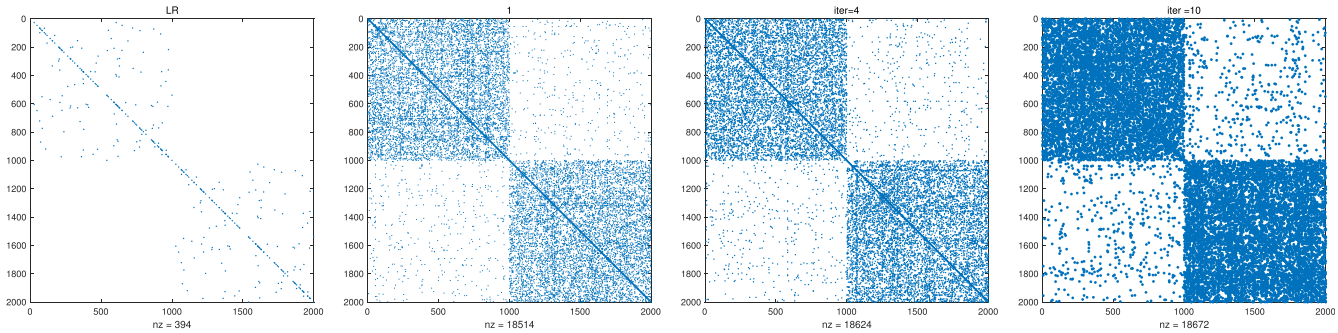
where n is the total number of instance and $\delta(\cdot)$ denotes an indicator function such that $\delta(x,y)=1$ if $x=y$ while $\delta(x,y)=0$ otherwise. $map(p)$ is a permute function that maps the predicted cluster label to match the ground truth label as much as possible.

Sparsity To study the sparsity property, we defined a sparsity ratio as $\rho(\mathbf{w}) = 1 - \frac{Card(\mathbf{w})}{m}$ where $card(\mathbf{w})$ is the number of non zeros of $\mathbf{w}$. For methods without a sparsity regularization we present an alternate definition as $card(\mathbf{w}) = count\ (|w_i|/\max_j(|w_j|) \geq 10^{-4})$, i.e., the number of weights with large relative magnitude [45].

Parameters In the experiments, we tune our k in a set $\{5,10,15\}$ and determine the regularization parameters and the leverage parameters in a set $\{10^{-3},10^{-2},10^{-1},0,10^{1},10^{2},10^{3}\}$ and report the best result.

5.3 Baseline Algorithms

Four state-of-the-art unsupervised feature selection algorithms are compared with ours as baseline algorithms. Specifically, we compare with a feature selection algorithm utilizing both link and content data, *NetFS* [8], which shows state-of-the-art performance in social network analysis. Moreover, three classical feature selection methods, i.e., *LapScore* [46] and *SPEC* [24] and *NDFS* [26], are also compared as baselines, which perform feature selection using the content data only. In the experiments, we refer our algorithm



(a) $\mathbf{R}$ (Init.) (b) iter. 1 (c) iter. 4 (d) iter. 10

Fig. 2. Distilled graph learning. Evolution of the similarity matrix w.r.t. iterations. In the initialization step, $\mathbf{R}$ is very sparse, which implies that we are only given limited link information of the data. After the 4th update of $\mathbf{L}$, the similarity matrix becomes denser. After that, more noisy links are identified in iteration 10.

as *DGC*. For the compared algorithms, we first apply each algorithm to select features and then perform clustering on the datums with the selected features. We report the average result of 20 independent experiments.

5.4 Experiments on Synthetic Dataset

We first test our algorithm on the synthetic dataset. First, we test the performance of distilled graph learning especially for our algorithm and evaluate the performances using the number of nonzero items in the similarity matrix. Additionally, in order to test the scalability of our algorithms, we increase the number of links and the number of features in the experiments and report the ACC, computation time and sparsity.

Distilled Graph Learning. We run our algorithm with ten iterations despite the stop condition reached at iteration 4. The test data is produced with 1,000 instances, 2,000 features and 10 percent positive links. For the convenience of visualization, we reorder the samples by their clusters. The sparse representation of $2,000 \times 2,000$ similarity matrix is illustrated in Fig. 2. Each point in the figure is the similarity evaluated between samples. As we can observe in Fig. 2a, in initialization step, $\mathbf{R}$ is very sparse. This indicates that we are only given limited link information of data. After the 4th update of $\mathbf{L}$, the similarity matrix becomes denser. After that, more noisy links are identified in iteration 10.

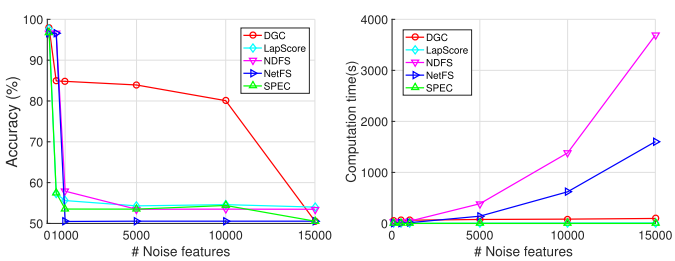
Influence of the Number of Noisy Features. We fix the number of instances to 1000, with 10 percent positive links and 10 percent noise links. As shown in Fig. 3a, clustering accuracies of all algorithms except ours witness a dramatically falling to around 50 percent when the number of noise feature increases to 1,000 and more. Our algorithm is the only one stays at 85 percent until it falls after giving 10,000 noise feature. Moreover, Fig. 3b shows that the computation time of our algorithm is not sensitive to the number of features, even with a large amount of 15,000. However, NDFS and NetFS consume much time when there are more then 5,000 noisy features.

Influence of the Number of Positive Links. Among all methods, only NetFS and the proposed DGC utilize both link and content data. The testing data contains 1,000 instances with 2,000 features. The number of positive links increase from 200 to 150,000. As shown in Fig. 4a, the performances of both NetFS and DGC are getting better as long as the number of the positive link increase. Our algorithm can get more benefits from the positive link information, which obtains a significant increase of the accuracy when there are more than 3,000 positive links. It implies that the propose method can use the link information better. As shown in 4b, the computation time of our algorithm is not sensitive to the number of links, while NetFS witnesses a growth of computation time after 50,000.

Sparsity. As shown in Fig. 5, for sparsity study, our algorithm performs the highest sparsity ratios in all the experiments around 98 percent. By introducing an $\ell_{2,1}$-norm as regularization term, NDFS algorithm also achieves a sparsity ratio around only 80 percent. NetFS witness a gradually increasing of sparsity while the number of noise feature growth. However, it is not sensitive to the number of positive links, as positive links are only used to construct pre-calculated latent low-dimension space for feature selection.

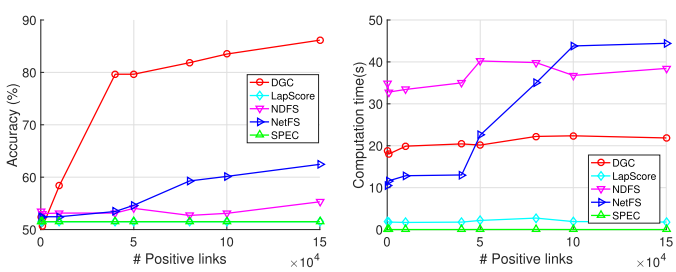
5.5 Experiments on Real-World Datasets

We test our algorithm on two real-world datasets BlogCatalog and Flickr. First, we vary the number of selected features and investigate the performances of all compared



(a) Acc. w.r.t.# Noise feat. (b) Time(s) w.r.t. # Noise Feat.

Fig. 3. Effect of # Noise Features (*abbr.* Noise Feat.) The result of our algorithm is reported as solid red lines.



(a) Acc. w.r.t. # Positive Lnk. (b) Time(s) w.r.t. # Positive Lnk.

Fig. 4. Effect of # Positive Links (*abbr.* Positive Lnk.) Our algorithm is shown in red solid lines.

TABLE 2
Clustering Performance in ACC w.r.t. Number of Features on BlogCatalog

ACC (%)										
# Features	200	400	600	800	1000	1200	1400	1600	1800	2000
LapScore	26.75	28.88	27.85	24.52	32.91	29.66	28.98	28.84	27.27	27.31
SPEC	18.02	18.44	19.04	19.55	20.31	22.80	24.02	22.32	21.11	20.03
NDFS	23.34	30.23	34.03	42.80	34.85	32.93	32.32	30.02	30.02	28.84
NetFS	**45.61**	47.74	51.13	**52.82**	48.84	48.15	52.59	52.07	49.73	50.42
DGC	43.19	**49.92**	**51.54**	51.98	**52.44**	**53.98***	**53.12**	**53.46**	**53.37**	**52.83**
NMI										
# Features	200	400	600	800	1000	1200	1400	1600	1800	2000
LapScore	0.080	0.064	0.021	0.032	0.081	0.061	0.061	0.051	0.064	0.061
SPEC	0.081	0.061	0.021	0.021	0.021	0.021	0.080	0.021	0.061	0.011
NDFS	0.132	0.150	0.172	0.133	0.132	0.121	0.131	0.152	0.122	0.151
NetFS	**0.27**	0.29	**0.33**	0.30	0.29	0.30	0.31	0.33	0.33	0.31
DGC	0.24	**0.30**	0.31	**0.31**	**0.32**	**0.33**	**0.35***	**0.34**	**0.34**	**0.34**

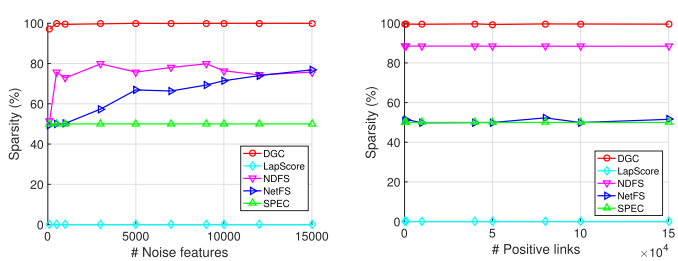
TABLE 3
Clustering Performance w.r.t. Number of Features on Flickr

ACC(%)										
# Features	200	400	600	800	1000	1200	1400	1600	1800	2000
LapScore	12.75	12.88	12.85	13.52	13.91	13.66	16.98	15.84	13.27	13.31
SPEC	11.02	12.44	11.04	13.55	14.31	14.80	14.02	15.32	14.11	13.03
NDFS	15.34	17.23	19.03	19.80	22.85	32.93	22.32	23.02	20.02	18.84
NetFS	**23.89**	27.47	30.41	35.28	38.97	43.70	45.47	47.44	**50.03**	**43.32**
DGC	16.26	**29.47**	**34.15**	**38.84**	**40.53**	**51.07***	**48.22**	**49.97**	43.65	42.95
NMI										
# Features	200	400	600	800	1000	1200	1400	1600	1800	2000
LapScore	0.080	0.064	0.021	0.032	0.081	0.061	0.061	0.051	0.064	0.061
SPEC	0.081	0.061	0.021	0.021	0.021	0.021	0.080	0.021	0.061	0.011
NDFS	0.031	0.051	0.710	0.123	0.132	0.120	0.13	0.221	0.22	0.251
NetFS	0.105	0.143	0.164	0.207	0.249	0.282	0.308	0.330	**0.355**	0.346
DGC	**0.249**	**0.270**	**0.316**	**0.319**	**0.327**	**0.343**	**0.340**	**0.342**	0.344	**0.388***

algorithms. We report the average performance of 20 independent experiments in ACC and NMI. Second, for our algorithm, we vary the parameter α w.r.t. the number of selected features to study the influence of link information. Notice in Eq. (3), the parameter α leverages the importance of the link information matrix. At last, to investigate the effect of regularization term, we vary the value of its leverage parameter λ.

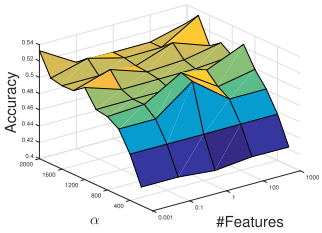
Influence of Number of Selected Features. In this experiment, we investigate the performance with the different number of selected features on two real-world datasets. We vary the number of selected features for all the algorithms on two datasets from $\{200, 400, \ldots, 1800, 2000\}$. The results are shown in Tables 3 and 2. We have the following observations:

- In general, the performances of all algorithms reach the peak when selecting around 1,200 features other than maximum 2,000 features. This demonstrates the importance of feature selection when doing spectral clustering.
- Our algorithm outperforms all algorithms in most cases on the clustering performance. In particular, by using both content and link information, our algorithm and NetFS show much better performance than others. This indicates that the utilization of link information improves the clustering performance on social network data.
- Our algorithm outperforms NetFS algorithm in overall performance (marked with the star in tables). The possible reason is that NetFS selects feature based on a low-dimensional latent space constructed by only link information. Thus it will be affected by the noisy links. On the contrary, our algorithm iteratively learns a distilled graph with the most representative feature

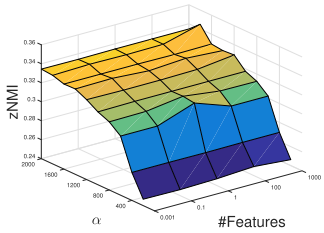


(a) Sparsity w.r.t. # Noise Feat. (b) Sparsity w.r.t. # Positive Lnk.

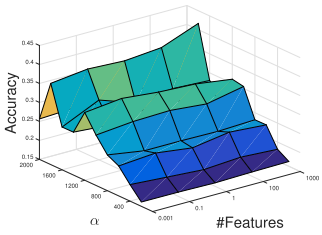
Fig. 5. Sparsity on synthetic data. Our algorithm is shown in red solid line.



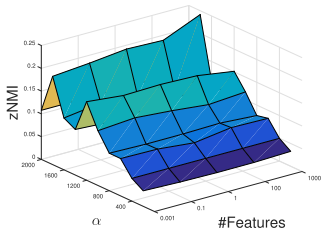
(a) Accuracy on BlogCatalog



(b) NMI on BlogCatalog

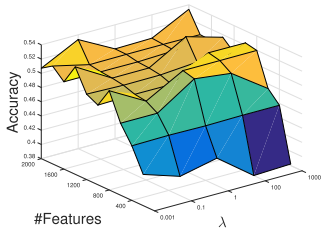


(c) Accuracy on Flickr

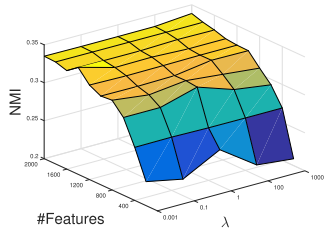


(d) NMI on Flickr

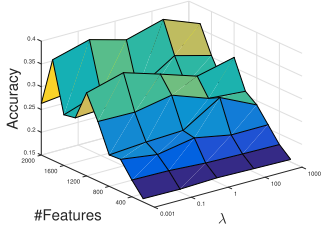
Fig. 6. Effect of α w.r.t. number of features.



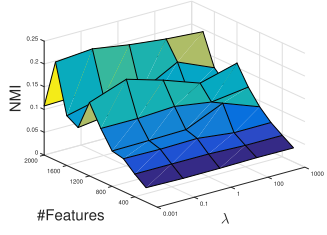
(a) Accuracy on BlogCatalog



(b) NMI on BlogCatalog



(c) Accuracy on Flickr



(d) NMI on Flickr

Fig. 7. Effect of λ w.r.t. number of features.

subset, which is less sensitive to the link data and more robust.

Influence of Link Information Matrix. We fix λ to 1 and vary α as $\{10^{-3}, 10^{-2}, 10^{-1}, 0, 10^{1}, 10^{2}, 10^{3}\}$ with different number of features from 200 to 2,000. As shown in Fig. 6, performances of our algorithm varies when the number of features changed however is not sensitive to values of α. It implies that our algorithm does not strongly rely on either content or link data, which is robust.

Influence of the Regularization Term. To study how λ affects the clustering performance, we fix α to 1 and vary λ in $\{10^{-3}, 10^{-2}, 10^{-1}, 0, 10^{1}, 10^{2}, 10^{3}\}$ with number of features varying from 200 to 2000. The results are shown in Fig. 7. As shown in Fig. 7, λ affects the performance of our algorithm as long as number of feature changing. However, there is no global optimal value for all the two data sets.

6 Conclusion and Extensions

In this paper, we proposed a novel algorithm to learn the distilled graph and select the best representative features simultaneously for spectral clustering on social network data. Specifically, we iteratively find the most representative feature subset w.r.t. the graph and then update the graph by using the selected features only. We compared our algorithm with other state-of-the-art baselines on both synthetic and real-world data sets, and the experimental results demonstrated the superior effectiveness and efficiency compared to the baselines. The proposed method can be further extended to learn the graphs in semi-supervised learning and supervised learning settings. The proposed algorithm is focused on the scenario of social network data analysis, which is a typical unsupervised learning setting. However, the proposed method can be extended to graph-based semi-supervised learning and supervised learning scenarios [10], [35], [47]. For example, the proposed method can be applied to the following settings.

$$\min_{\mathbf{W}} \ \mathrm{tr}(\mathbf{W}^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{L}(\boldsymbol{\tau})\mathbf{X}\mathbf{W}), \ \text{s.t.} \ \mathbf{W}^{\mathsf{T}}\mathbf{W} = \mathbf{I}, \tag{26}$$

or

$$\min_{\mathbf{W}} \ \mathrm{tr}(\mathbf{W}^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{L}(\boldsymbol{\tau})\mathbf{X}\mathbf{W}), \ \text{s.t.} \ \mathbf{W}^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{B}(\boldsymbol{\tau})\mathbf{X}\mathbf{W} = \mathbf{I}, \tag{27}$$

where $\mathbf{B}(\boldsymbol{\tau})$ contains the supervised information under the graph embedding framework [47]. Note that, for problem Eq. (26), if the graph is a fully-connected graph (namely $\boldsymbol{\tau} = 0$), it follows $\mathbf{L}(\boldsymbol{\tau}) = \mathbf{I} - \frac{1}{N}\mathbf{e}\mathbf{e}^{\mathsf{T}}$. Then problem Eq. (26) becomes the classical PCA problem [47].

Acknowledgments

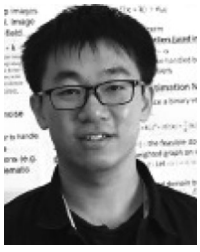
This work was supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/ Interior Business Center (DOI/IBC) contract number D17PC00340. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation/herein. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DOI/IBC, or the U.S. Government.

References

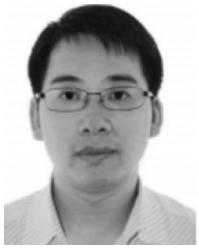
[1] J. Tang and H. Liu, "Feature selection with linked data in social media," in *Proc. SIAM Int. Conf. Data Mining*, 2012, pp. 118–128.
[2] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annu. Rev. Sociology*, pp. 415–444, 2001.
[3] P. Padungweang, C. Lursinsap, and K. Sunat, "A discrimination analysis for unsupervised feature selection via optic diffraction principle," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 10, pp. 1587–1600, Oct. 2012.
[4] P. V. Marsden and N. E. Friedkin, "Network studies of social influence," *Sociol. Methods Res.*, vol. 22, no. 1, pp. 127–151, 1993.
[5] F. Han, H. Qiu, H. Liu, and B. Caffo, "On the impact of dimension reduction on graphical structures," *arXiv:1404.7547 [stat.ME]*, Apr. 2014.
[6] J. Tang and H. Liu, "Unsupervised feature selection for linked social media data," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 904–912.
[7] J. Li, X. Hu, J. Tang, and H. Liu, "Unsupervised streaming feature selection in social media," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 1041–1050.

[8] J. Li, X. Hu, L. Wu, and H. Liu, "Robust unsupervised feature selection on networked data," in *Proc. SIAM Int. Conf. Data Mining*, 2016, pp. 387–395.
[9] X. Wang, L. Tang, H. Gao, and H. Liu, "Discovering overlapping groups in social media," in *Proc. IEEE 10th Int. Conf. Data Mining*, 2010, pp. 569–578.
[10] N. Wang, Z. Liu, and X. Li, "Graph-based semi-supervised feature selection for social media data," in *Foundations of Intelligent Systems*. New York, NY, USA: Springer, 2014, pp. 115–124.
[11] J. Tang and H. Liu, "An unsupervised feature selection framework for social media data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 12, pp. 2914–2927, Dec. 2014.
[12] X. Wei, B. Cao, and S. Y. Philip, "Unsupervised feature selection on networks: A generative view," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2215–2221.
[13] D. Papadimitriou, G. Koutrika, Y. Velegrakis, and J. Mylopoulos, "Finding related forum posts through content similarity over intention-based segmentation," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 9, pp. 1860–1873, Sep. 2017.
[14] A. Y. Ng, M. I. Jordan, Y. Weiss, et al., "On spectral clustering: Analysis and an algorithm," *Proc. 14th Int. Conf. Neural Inf. Process. Syst.: Natural Synthetic*, 2002, vol. 2, pp. 849–856.
[15] Y. Yang, H. T. Shen, F. Nie, R. Ji, and X. Zhou, "Nonnegative spectral clustering with discriminative regularization," in *Proc. 25th AAAI Conf. Artif. Intell.*, 2011, pp. 555–560.
[16] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," in *Proc. 17th Int. Conf. Neural Inf. Process. Syst.*, 2004, pp. 1601–1608.
[17] Y. Yang, F. Shen, Z. Huang, H. T. Shen, and X. Li, "Discrete nonnegative spectral clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 9, pp. 1834–1845, Sep. 2017.
[18] L. Wu, X. Wu, A. Lu, and Y. Li, "On spectral analysis of signed and dispute graphs: Application to community structure," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 7, pp. 1480–1493, Jul. 2017.
[19] Y. Song, C. Wang, M. Zhang, H. Sun, and Q. Yang, "Spectral label refinement for noisy and missing text labels," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2972–2978.
[20] K.-Y. Chiang, C.-J. Hsieh, N. Natarajan, I. S. Dhillon, and A. Tewari, "Prediction and clustering in signed networks: A local to global perspective," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1177–1213, 2014.
[21] P. Anchuri and M. Magdon-Ismail , "Communities and balance in signed networks: A spectral approach," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, 2012, pp. 235–242.
[22] K.-Y. Chiang, J. J. Whang, and I. S. Dhillon, "Scalable clustering of signed networks using balance normalized cut," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, 2012, pp. 615–624.
[23] Q. Zheng and D. Skillicorn, "Spectral embedding of signed networks," in *Proc. SIAM Int. Conf. Data Mining*, 2015, pp. 55–63.
[24] Z. Zhao and H. Liu, "Spectral feature selection for supervised and unsupervised learning," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 1151–1157.
[25] D. Cai, C. Zhang, and X. He, "Unsupervised feature selection for multi-cluster data," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 333–342.
[26] Z. Li, Y. Yang, J. Liu, X. Zhou, and H. Lu, "Unsupervised feature selection using nonnegative spectral analysis," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, pp. 1026–1032.
[27] H. Wang, P. Zhang, X. Zhu, I. W.-H. Tsang, L. Chen, C. Zhang, and X. Wu, "Incremental subgraph feature selection for graph classification," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 1, pp. 128–142, Jan. 2017.
[28] Q. Gu, Z. Li, J. Han, et al., "Joint feature selection and subspace learning," in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, vol. 22, no. 1, 2011, pp. 1294–1299.
[29] Q. Gu, M. Danilevsky, Z. Li, and J. Han, "Locality preserving feature learning," in *Proc. Artif. Intell. Statist.*, 2012, pp. 477–485.
[30] Q. Gu and J. Han, "Towards feature selection in network," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage.*, 2011, pp. 1175–1184.
[31] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
[32] M. Tan, I. W. Tsang, and L. Wang, "Towards ultrahigh dimensional feature selection for big data," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1371–1429, 2014.
[33] D. Gong, M. Tan, Q. Shi, A. van den Hengel, and Y. Zhang, "Mptv: Matching pursuit based total variation minimization for image deconvolution," *IEEE Trans. Image Process.*, vol. 28, no. 4, pp. 1851–1865, Apr. 2019.
[34] D. Gong, M. Tan, Y. Zhang, A. Van den Hengel, and Q. Shi, "Blind image deconvolution by automatic gradient activation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1827–1836.
[35] L. Sun, S. Ji, and J. Ye, "A least squares formulation for a class of generalized eigenvalue problems in machine learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 977–984.
[36] M. Tan, L. Wang, and I. W. Tsang, "Learning sparse svm for feature selection on very high dimensional datasets," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 1047–1054.
[37] D. Gong, M. Tan, Y. Zhang, A. van den Hengel, and Q. Shi, "Mpgl: An efficient matching pursuit method for generalized lasso," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1934–1940.
[38] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[39] K. Kortanek and H. No, "A central cutting plane algorithm for convex semi-infinite programming problems," *SIAM J. Optim.*, vol. 3, no. 4, pp. 901–918, 1993.
[40] K.-C. Toh and S. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems," *Pacific J. Optim.*, vol. 6, no. 615–640, 2010, Art. no. 15.
[41] A. F. Martins, M. A. Figueiredo, P. M. Aguiar, N. A. Smith, and E. P. Xing, "Online multiple kernel learning for structured prediction," *arXiv:1010.2770 [stat.ML]*, Oct. 2010.
[42] F. Nie, W. Zhu, and X. Li, "Unsupervised large graph embedding," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 2422–2428.
[43] Z. Xu, R. Jin, J. Ye, M. R. Lyu, and I. King, "Non-monotonic feature selection," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 1145–1152.
[44] S. X. Yu and J. Shi, "Multiclass spectral clustering," in *Proc. 9th IEEE Int. Conf. Comput. Vis. - Vol. 2*, 2003, pp. 313–319.
[45] A. B. Chan, N. Vasconcelos, and G. R. Lanckriet, "Direct convex relaxations of sparse svm," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 145–153.
[46] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," in *Proc. 18th Int. Conf. Neural Inf. Process. Syst.*, 2005, pp. 507–514.
[47] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 40–51, Jan. 2007.

Wenhe Liu received the MS degree in artificial intelligence from The University of Edinburgh, United Kingdom, in 2012 and the PhD degree in computer science form the University of Technology Sydney, Australia, in 2019. He is currently a post-doctoral associate with the School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania. His research interests include machine learning and its applications to multimedia and computer vision.



Dong Gong received the bachelor's and PhD degrees in computer science from the Northwestern Polytechnical University, Xian, China, in 2012 and 2018, respectively. He is currently a research fellow with the School of Computer Science, The University of Adelaide, Adelaide, SA, Australia. His current research interests include machine learning, optimization, computer vision, and image processing.



Mingkui Tan received the PhD degree in computer science from Nanyang Technological University, Singapore, in 2014. He is a professor with the School of Software Engineering, South China University of Technology. After that, he worked as a senior research associate with the School of Computer Science, University of Adelaide, Australia. His research interests include compressive sensing, big data learning, and large-scale optimization.

Qinfeng Shi is the director and founder of the Probabilistic Graphical Model Group, director of the Advanced Reasoning and Learning of Australian Institute for Machine Learning, and an associate professor with The University of Adelaide. His research interests include machine learning and computer vision, particularly probabilistic graphical models and deep learning. He has more than 3000 Google cites and H-index of 24. He has been a chief investigator (CI) for multiple Australian Research Council (ARC) grants (three as lead CI, two as co-CI). He is the first ARC Discovery Early Career Researcher Awardee (DECRA) in Machine Learning (2012-2014), and has been invited twice to Prime Ministers Science Prizes Dinner. His group develops efficient algorithms and systems that can evolve and learn from data in almost any domain ranging from computer vision, water utility, health, smart agriculture, and smart manufacturing (Industry 4.0).

Yi Yang received the PhD degree in computer science from Zhejiang University, Hangzhou, China, in 2010. He is a professor with the University of Technology Sydney, Australia. His research interests include machine learning and its applications to multimedia content analysis and computer vision, such as multimedia indexing and retrieval, surveillance video analysis, and video content understanding.

Alexander G. Hauptmann received the BA and MA degrees in psychology from Johns Hopkins University, Baltimore, Maryland, the PhD degree in computer science from the Technische Universitt Berlin, Berlin, Germany, in 1984, and the PhD degree in computer science from Carnegie Mellon University (CMU), Pittsburgh, Pennsylvania, in 1991. From 1984 to 1994, he worked on speech and machine translation, when he joined the Informedia Project for digital video analysis and retrieval, and led the development and evaluation of news-on-demand applications. He is currently with the faculty of the Department of Computer Science and the Language Technologies Institute, CMU. His current research interests include man-machine communication, natural language processing, speech understanding and synthesis, video analysis, and machine learning.