Chapter 10

Big Data Analytics for Social Networks

Brian Baingana, Panagiotis Traganitis, Georgios Giannakis

University of Minnesota, U.S.A.

Gonzalo Mateos

University of Rochester, U.S.A.

10.1 Introduction

The last few years have witnessed a dramatic upswing in the volume, variety, and acquisition rate of data from disparate sources. Among other factors, this "data deluge" has been fueled by the ubiquity of the web, pervasive deployment of low-cost sensors, cheaper storage memory, and growing awareness that data is a key asset from which valuable insights can be unlocked. Social networks lie at the forefront of this revolution, presenting ample opportunities to address several challenges associated with big data. Examples include micro-blogging services (e.g., Twitter), web-based friendship networks (e.g., Facebook), and online product reviews (e.g., Yelp).

Exploiting the immense big data opportunities comes at the cost of overcoming significant challenges. First, traditional analytics are ill-equipped to cope with the sheer volume of data. Many distributed platforms (e.g., Hadoop/MapReduce and GraphLab) have emerged to tame the scale of data. Nevertheless, only a subset of algorithms are readily implementable on these platforms, and development of more versatile architectures is an active research area.

In addition, most social data are high-dimensional with many features. In order to finesse the curse of dimensionality, parsimonious models must be devised for feature subset selection. Since most big data are acquired sequentially as streaming inputs, batch learning and optimization approaches are impractical. For example, securities traders interested in capturing market sentiment from real-time tweets are driven by the need for split-second buy/sell decisions. Furthermore, big data acquisition pipelines are plagued by measurement inaccuracies, misses, and incompleteness due to, e.g., privacy concerns.

10.1.1 Signal Processing for Big Data

Signal processing (SP) provides a principled framework within which big data challenges can be readily addressed [SGM14]. Advances in SP have formalized parsimonious models that exploit key properties inherent to big data, e.g., sparsity, low-rank, and manifold structures. For example, several contem-

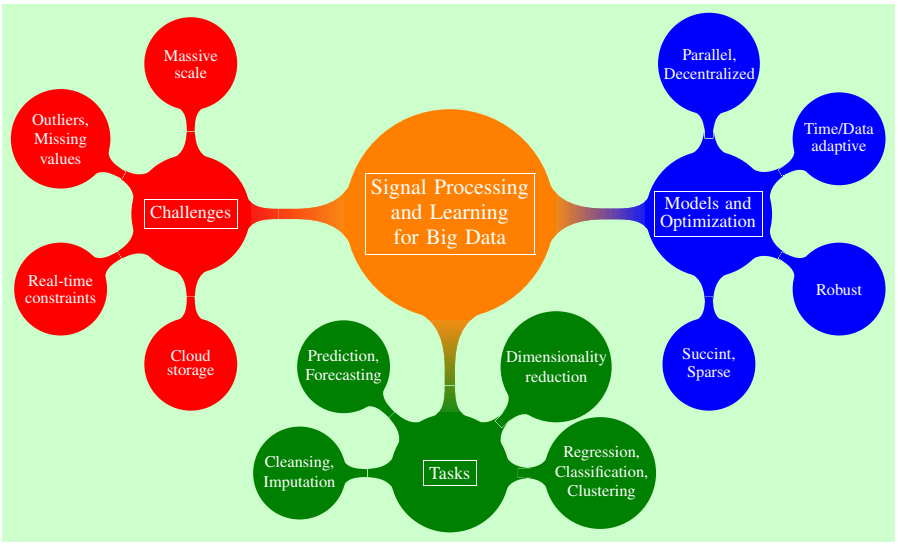


FIGURE 10.1.1: Signal processing in the context of Big Data analytics [SGM14].

porary models jointly capture low rank and sparsity (see, e.g., [MMG13c]), subsuming several learning paradigms such as principal component analysis (PCA) [HTF09], dictionary learning (DL) [OF97], and compressive sampling (CS) [CW08].

Scaling to very large problem instances, while attaining real-time operation are noteworthy themes that have shaped the direction of contemporary research efforts. The *alternating direction method of multipliers (ADMM)* has enjoyed growing popularity in decentralized learning and optimization algorithms, especially in settings where data resides over a network of computing nodes [MBG10, BPC$^+$11, MMG13a].

For streaming data, *online learning* has emerged as a powerful framework for real-time analytics [MBPS10, KST11, MMG13b]. Popular online algorithms including online mirror descent, and online gradient descent have been studied and deployed in practical social network settings; see e.g., [SS11] for a comprehensive review. Figure 10.1.1 depicts various themes for which SP and learning offer a comprehensive framework.

In addition to decentralized computation, significant improvements in running time have been realized though random sampling and random projection algorithms [Mah11]. These methods operate on a randomized sketch of an input data matrix by either: i) sampling a small subset of rows of the matrix that are most informative (random sampling); or ii) linearly combining a small number of rows of the matrix (random projection). Under reasonable

conditions, these randomized approaches can in theory attain asymptotically faster worst-case running times than deterministic alternatives.

10.1.2 Social Network Analytics Problems

Initial social network studies were exploratory in nature and unveiled surprising properties common to large networks e.g., existence of power laws, small-world properties, and preferential attachment strategies for network growth [New10, EK10]. The exponential growth of the web coupled with ground-breaking advances in fields as diverse as computational biology, and epidemiology have led to a dramatic change in the scope and size of problems studied. A few examples include ranking of web pages [BP98, CDK+99], discovery of causal interactions in gene regulatory networks [CBG13], and prediction of the spread of infectious diseases [VR07, Jac10].

Looking at network science through the lens of statistical learning, several contemporary problems boil down to (non-)parametric regression, dimensionality reduction, clustering, or (semi-)supervised classification. "Work-horse" dimensionality reduction approaches such as multidimensional scaling (MDS) have been advocated for network visualization tasks [KK89, BP11, BG13]. Topology inference problems from network processes have been the focus of several recent works [RLS10, BMG14]. Another line of research has concentrated on virus (also information, buying patterns) propagation models over complex networks [Het00, EK10]. Other interesting topics include community discovery [GN02, For10], prediction of network processes from partial observations [Kol09, FRG14], and detection of anomalies in social networks [MG12b]. This chapter is representative of a subset of problems, for which the identified big data challenges are addressed by leveraging statistical learning advances.

Section 10.2 focuses on scalable visualization of social networks via nonlinear dimensionality reduction. Inference and imputation of corrupted signals over social graphs is the topic of Section 10.3. This is followed by Section 10.4, which presents algorithms for community discovery in big social networks. Finally, Section 10.5 presents recent algorithms for tracking topologies of dynamic social networks that facilitate diffusion of network processes. Admittedly, these approaches represent a small fraction of the gamut of social network analytics methods. Nevertheless, they are representative of a broader class of contemporary tools that are relevant to big data analytics in social networks.

10.2 Visualizing and reducing dimension in social nets

Network visualization is often accomplished through graph embedding, which entails mapping each node to a point in Euclidean space. Due to the data

deluge spawned by modern network-based phenomena, the rising complexity and sheer volume of networks present new opportunities and challenges for graph embedding tools that capture global patterns and use visual metaphors to convey meaningful structural information e.g., hierarchy, node similarity, and natural communities [HL12].

Most traditional visualization algorithms trade off the clarity of structural characteristics of the underlying network for aesthetic requirements like minimal edge crossing and fixed edge lengths; e.g., [KK89, PT98, DBD13]. Although efficient for small graphs (hundreds of nodes), embeddings for larger graphs generated by classical approaches are seldom structurally informative. To this end, several approaches have been developed for embedding graphs while preserving specific structural properties. Pioneering methods (e.g., [KK89]) have resorted to *multidimensional scaling (MDS)*, which seeks a low-dimensional representation of high-dimensional data so that pairwise dissimilarities are preserved through Euclidean distances in the embedding [BG05, BG13]. In this case, the vertex dissimilarity structure is preserved through pairwise distance metrics in the embedding. Spectral embeddings whose coordinates consist of entries in the leading principal components of the network adjacency matrix are advocated in [LWH03]. The structure-preserving embedding algorithm solves a semidefinite program with linear topology constraints that emphasize reconstructability of the graph through neighborhood methods [SJ09]. Other visualization methods emphasize community structure [YLZ+13], while concentric layouts emphasize node hierarchy by placing the highest ranked nodes at the center of the embedding [AHDBV06, BP11, BG13].

Despite the rich history associated with graph drawing methods, development of visualization techniques that effectively capture hierarchical structure and other global patterns remains a challenging and active area of research. This section showcases a recently developed kernel-based visualization approach that leverages *local linear embedding (LLE)*, a popular manifold learning technique [BG15]. Similar to recent works on graph embedding, node importance is captured through *centrality* constraints [BP11, BG13]. In general, centrality measures provide a means to assign a level of importance to each node in a network [Sab66, Fre77]. For instance, betweenness centrality describes the extent to which information is routed through a specific node by measuring the fraction of all shortest paths traversing it; see e.g., [Kol09, p. 89]. Other measures include closeness and eigenvalue centrality.

10.2.1 Kernel-based graph embedding

Consider a network represented by an undirected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{E}$ denotes the set of edges, and $\mathcal{V}$ the set of vertices with cardinality $|\mathcal{V}| = N$. Suppose the structure of G is captured by its so-termed adjacency matrix $\mathbf{A}$ whose (i, j)-th entry (hereafter denoted by a_{ij}) is zero only if edge $(i, j) \notin \mathcal{E}$, otherwise it denotes the weight of (i, j). Given G and a prescribed embedding

dimension p (typically $p \in \{2, 3\}$), the graph embedding task is tantamount to searching for the set of $p \times 1$ vectors $\mathcal{X} := \{\mathbf{x}_i\}_{i=1}^N$ which "effectively" capture the underlying local graph structure.

Suppose $\{\mathbf{y}_i \in \mathbb{R}^q\}_{i=1}^N$ are data points sampled from a nonlinear manifold. LLE seeks the low-dimensional vectors $\{\mathbf{x}_i \in \mathbb{R}^p\}_{i=1}^N$ ($p \ll q$) that preserve the local neighborhood structure on the manifold. First, the neighborhoods $\{\mathcal{N}_i\}_{i=1}^N$ are constructed per datum by selecting the K-nearest neighbors of i, or setting $\mathcal{N}_i := \{\mathbf{y}_j \in \mathbb{R}^q : \|\mathbf{y}_i - \mathbf{y}_j\|_2 \leq \epsilon, \epsilon > 0, j = 1, \ldots, N\}$. Assuming $|\mathcal{N}_i| = K$, each point is then fit to a linear combination of its neighbors by solving the following constrained least-squares (LS) optimization problem

$$\underset{\left\{\sum_{j\in\mathcal{N}_i} w_{ij}=1\right\}}{\arg\min} \left\| \mathbf{y}_i - \sum_{j\in\mathcal{N}_i} w_{ij}\mathbf{y}_j \right\|_2^2 \quad i = 1, \ldots, N, \tag{10.2.1}$$

where $\{w_{ij}\}_{j=1}^K$ are the reconstruction weights for point i, while the constraint enforces shift invariance. Setting $w_{ij} = 0$ for $j \notin \mathcal{N}_i$, the final step determines $\{\mathbf{x}_i \in \mathbb{R}^p\}_{i=1}^N$ so that reconstruction weights are preserved by solving:

$$\underset{\left\{\begin{smallmatrix}\sum_{i=1}^N \mathbf{x}_i=\mathbf{0}\\ \frac{1}{N}\sum_{i=1}^N \mathbf{x}_i\mathbf{x}_i^T=\mathbf{I}\end{smallmatrix}\right\}}{\arg\min} \sum_{i=1}^N \left\| \mathbf{x}_i - \sum_{j=1}^N w_{ij}\mathbf{x}_j \right\|_2^2. \tag{10.2.2}$$

The equality constraints are included to eliminate the trivial all-zero solution, and also to eliminate shift and rotation ambiguities.

In order to tailor LLE for graph embedding where only weights $\{a_{ij}\}$ are available, one must contend with the general non-existence of high dimensional vectors $\{\mathbf{y}_i\}_{i=1}^N$ defined per node. Fortunately, the optimization problem (10.2.1) can be cast in terms of the inner products $\mathbf{y}_i^T\mathbf{y}_j$ for all $i, j \in \{1, \ldots, N\}$. This brings to bear the merits of kernel methods which entail computations on inner products of transformed feature vectors, $\phi(\mathbf{y})$, namely $k_{ij}(\mathbf{y}_i, \mathbf{y}_j) = \phi^T(\mathbf{y}_i)\phi(\mathbf{y}_j)$. However, this flexibility comes at the challenge of selecting the best kernel matrix $\mathbf{K} \in \mathbb{R}^{N\times N}$ where $[\mathbf{K}]_{ij} := k_{ij}(\mathbf{y}_i, \mathbf{y}_j)$. A few choices of $\mathbf{K}$ include:

i. The doubly-centered dissimilarity matrix $\mathbf{K} = -(1/2)\mathbf{J}\boldsymbol{\Delta}^{(2)}\mathbf{J}$, where $[\boldsymbol{\Delta}^{(2)}]_{ij}$ denotes the squared geodesic distance between nodes i and j, or any other dissimilarity metric on the graph, and $\mathbf{J} := \mathbf{I} - N^{-1}\mathbf{1}\mathbf{1}^T$ denotes the centering operator ($\mathbf{I}$ is the identity matrix and $\mathbf{1}$ is the all-one column vector) [BG05]. In this case, $\mathbf{K}$ is reminiscent of the kernel adopted by classical MDS.

ii. The Penrose-Moore pseudoinverse of the graph Laplacian; that is $\mathbf{K} = \mathbf{L}^\dagger$, where $\mathbf{L} := \mathbf{A} - \mathbf{D}$, $\mathbf{A} \in \{0, 1\}^{N\times N}$ denotes the binary graph adjacency

matrix, and $\mathbf{D} := \text{diag}(\mathbf{A1})$. It turns out that $\mathbf{L}^\dagger$ admits an intuitive interpretation as a similarity matrix based on random walk distances on graphs [FPRS07].

iii. Matrix $\mathbf{K} = \mathbf{A}\mathbf{A}^T$, where $\mathbf{A} \in \{0,1\}^{N\times N}$, and $[\mathbf{A}\mathbf{A}^T]_{ij}$ counts the number of single-hop neighbors shared by nodes i and j.

Neighborhood selection in traditional LLE entails $\mathcal{O}\left(qN^2\right)$ complexity with $q \gg$. This bottleneck can be overcome by setting $\mathcal{N}_i$ to the single-hop neighbors per node. Let $\mathbf{Y}_i := [\phi(\mathbf{y}_1^i), \ldots, \phi(\mathbf{y}_{d_i}^i)]$ collect the "virtual" transformed vectors associated per $\mathcal{N}_i$, where d_i denotes the degree of node i. Letting $\mathbf{w}_i := [w_{i1}, \ldots, w_{id_i}]^T$, the constrained LS fit (10.2.1) can be written as:

$$\mathbf{w}_i = \underset{\{\mathbf{w}:\, \mathbf{1}^T\mathbf{w}=1\}}{\arg\min}\ \mathbf{w}^T\mathbf{K}_i\mathbf{w} - 2\mathbf{w}^T\mathbf{k}_i, \tag{10.2.3}$$

where $\mathbf{K}_i := \mathbf{Y}_i^T\mathbf{Y}_i$ and $\mathbf{k}_i := \mathbf{Y}_i^T\phi(\mathbf{y}_i)$ are submatrices of $\mathbf{K}$ indexed by elements of $\mathcal{N}_i$. Resorting to Lagrange multiplier theory, one can readily solve for $\mathbf{w}_i$ in (10.2.3) in closed form. Moreover for large-scale graph embedding, (10.2.3) can be easily parallelized over clusters of computing nodes. Each subproblem entails $\mathcal{O}(d_i^3)$ complexity, which is manageable because typically $d_i \ll N$.

The low-dimensional graph embedding can be evaluated from the reconstruction weights via (10.2.2), by solving for:

$$\underset{\left\{\begin{smallmatrix}\mathbf{X}\\ \sum_{i=1}^N \mathbf{x}_i=\mathbf{0}\\ \frac{1}{N}\sum_{i=1}^N \mathbf{x}_i\mathbf{x}_i^T=\mathbf{I}\end{smallmatrix}\right\}}{\arg\min}\ \text{Tr}\left[\mathbf{X}^T(\mathbf{I}-\mathbf{W})^T(\mathbf{I}-\mathbf{W})\mathbf{X}\right], \tag{10.2.4}$$

where $\mathbf{W}^T := [\tilde{\mathbf{w}}_1, \ldots, \tilde{\mathbf{w}}_N]$, $\tilde{w}_{ij} = w_{ij}$ if $j \in \mathcal{N}_i$ otherwise $\tilde{w}_{ij} = 0$, $\mathbf{X}^T := [\mathbf{x}_1, \ldots, \mathbf{x}_N]$, and Tr(.) denotes matrix trace. The solution comprises the 2nd to the $(p+1)$st least dominant eigenvectors of $(\mathbf{I}-\mathbf{W})^T(\mathbf{I}-\mathbf{W})$. For large graphs, $\mathbf{X}$ can be efficiently computed via orthogonal iterations which are amenable to decentralization, entailing $\mathcal{O}(pN^2)$ complexity; see e.g., [KM08].

Although this approach preserves the local graph topology defined by single-hop neighbors, the spectral decomposition is generally not scalable for very large networks. Moreover for large network visualization tasks, one is often more interested in conveying global properties such as node hierarchy. To this end, the next subsection modifies the embedding step to enforce centrality constraints.

10.2.2 Centrality-constraints

Large-scale settings call for emphasis on structural properties such as node hierarchy over aesthetics. Centrality measures impose a hierarchical ordering among nodes by quantifying the relative importance of nodes over their peers

e.g., degree, closeness, and betweenness centralities [Kol09]. As a result, graph embedding under centrality constraints yields very informative network visualizations. Let $\mathcal{C}(\mathcal{G}) := \{c_i\}_{i=1}^N$ denote the set of centralities of $\mathcal{G}$, with c_i representing the centrality measure of node i. The goal is to determine the embedding $\{\mathbf{x}_i\}_{i=1}^N$ that effectively "preserves" the centrality ordering in $\mathcal{C}(\mathcal{G})$.

Modifying the final step in (10.2.2) to incorporate centrality constraints yields the following optimization problem:

$$\begin{aligned} \underset{\mathbf{x}_1,\ldots,\mathbf{x}_N}{\arg\min} \quad & \sum_{i=1}^N \left\| \mathbf{x}_i - \sum_{j=1}^N w_{ij}\mathbf{x}_j \right\|_2^2 \\ \text{s. t.} \quad & \|\mathbf{x}_i\|_2^2 = f^2(c_i),\ i = 1,\ldots,N, \end{aligned} \tag{10.2.5}$$

where $f(c_i)$ is a monotone decreasing function of c_i ensuring that more central nodes are placed closer to the center. The dropped $\mathbf{0}$-mean constraint can be compensated for by a post-processing centering operation upon determination of $\{\hat{\mathbf{x}}_i\}_{i=1}^N$.

Problem (10.2.5) is non-convex without global optimality guarantees. Fortunately, the problem decouples over vectors $\{\mathbf{x}_i\}_{i=1}^N$ motivating a block coordinate descent (BCD) approach [Ber99]. The optimization variables can now be partitioned into N blocks with $\mathbf{x}_i$ corresponding to block i. Consequently, during iteration r one cycles through all blocks by solving:

$$\begin{aligned} \mathbf{x}_i^r = \underset{\mathbf{x}}{\arg\min} \quad & \left\| \mathbf{x} - \sum_{j<r} w_{ij}\mathbf{x}_j^r - \sum_{j>r} w_{ij}\mathbf{x}_j^{r-1} \right\|_2^2 \\ \text{s. t.} \quad & \|\mathbf{x}\|_2^2 = f^2(c_i). \end{aligned} \tag{10.2.6}$$

Letting $\mathbf{v}_i^r := \sum_{j<r} w_{ij}\mathbf{x}_j^r + \sum_{j>r} w_{ij}\mathbf{x}_j^{r-1}$ and $\lambda \geq 0$ denote a Lagrange multiplier, the solution:

$$\mathbf{x}_i^r = \underset{\mathbf{x}}{\arg\min} \quad \|\mathbf{x} - \mathbf{v}_i^r\|_2^2 + \lambda(\|\mathbf{x}\|_2^2 - f^2(c_i)) \tag{10.2.7}$$

is given by:

$$\mathbf{x}_i^r = \frac{\mathbf{v}_i^r}{1+\lambda}. \tag{10.2.8}$$

Upon substitution of (10.2.8) into the equality constraint in (10.2.6), one obtains the closed-form per-iteration update:

$$\mathbf{x}_i^r = \begin{cases} \frac{\mathbf{v}_i^r}{\|\mathbf{v}_i^r\|_2} f(c_i), & \text{if } \|\mathbf{v}_i^r\|_2 > 0 \\ \mathbf{x}_i^{r-1}, & \text{otherwise.} \end{cases} \tag{10.2.9}$$

If $\mathbf{X}^r$ denotes the embedding matrix after r BCD iterations, the operation $\mathbf{X} = (\mathbf{I} - N^{-1}\mathbf{1}\mathbf{1}^T)\mathbf{X}^r$ centers $\{\mathbf{x}_i^r\}_{i=1}^N$ to the origin in order to satisfy the shift invariance property of the embedding. Algorithm 10.2.1 summarizes the steps

outlined for the centrality-constrained graph embedding scheme. The only inputs to the algorithm are the graph topology $\mathcal{G}$, the centrality measures, $\{c_i\}_{i=1}^N$, the graph embedding dimension p, and the kernel matrix $\mathbf{L}$. Note that Algorithm 10.2.1 scales well to big data settings since both steps can be computed in parallel. Moreover, (10.2.3) entails $\mathcal{O}(d_i^3)$ complexity, with $d_i \ll N$, and the dimensionality of (10.2.6) is $p \in \{2, 3\}$.

Algorithm 10.2.1: Centrality-constrained graph embedding

Input: $\mathcal{G}$, $\mathcal{C}(\mathcal{G})$, $\mathbf{K}$, ϵ, p
for $i = 1 \dots N$ (in parallel) **do**
  Set $\mathcal{N}_i$ to single-hop neighbors of i
  Extract $\mathbf{K}_i$ and $\mathbf{k}_i$ from $\mathbf{K}$
  Solve $\mathbf{w}_i = \arg\min_{\mathbf{w}} \ \mathbf{w}^T\mathbf{K}_i\mathbf{w} - 2\mathbf{w}^T\mathbf{k}_i$ s. t. $\mathbf{1}^T\mathbf{w} = 1$
  Set $w_{ij} = 0$ for $j \notin \mathcal{N}_i$
end for
Initialize $\mathbf{X}^0$, $r = 0$
repeat
  $r = r + 1$
  for $i = 1 \dots N$ (in parallel) **do**
    Compute $\mathbf{x}_i^r$ according to (10.2.9)
    $\mathbf{X}^r(i,:) = (\mathbf{x}_i^r)^T$
  end for
until $\|\mathbf{X}^r - \mathbf{X}^{r-1}\|_F \leq \epsilon$
$\mathbf{X} = (\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T)\mathbf{X}^r$

10.2.3 Numerical tests

This section presents numerical tests conducted on a synthetic small-world network generated by the Watts-Strogatz model [WS98] and Gnutella, a real-world file-sharing network. Given the number of nodes N, average degree $\bar{d}$, and $\beta \in [0, 1]$, the Watts-Strogatz model constructs a $\bar{d}$-regular ring lattice and rewires each edge with probability β. The synthetic graph was generated with $N = 2\times10^3$, $\bar{d} = 4$, and $\beta = 0.3$. Several centrality measures are available with emphasis on different importance criteria. For instance, *closeness centrality* captures the extent to which a particular node is close to all other nodes in the network, and it is commonly defined as $c_i := 1/(\sum_{j\in\mathcal{V}} d_{ij})$, where d_{ij} denotes the geodesic distance between nodes i and j. For the experiments, closeness centralities were transformed as follows:

$$f(c_i) = \left(\frac{c_{\max} - c_i}{c_{\max} - c_{\min}}\right), \tag{10.2.10}$$

where $c_{\max} := \max_i \ c_i$, and $c_{\min} := \min_i \ c_i \ \ \forall i = 1, \dots, N$.

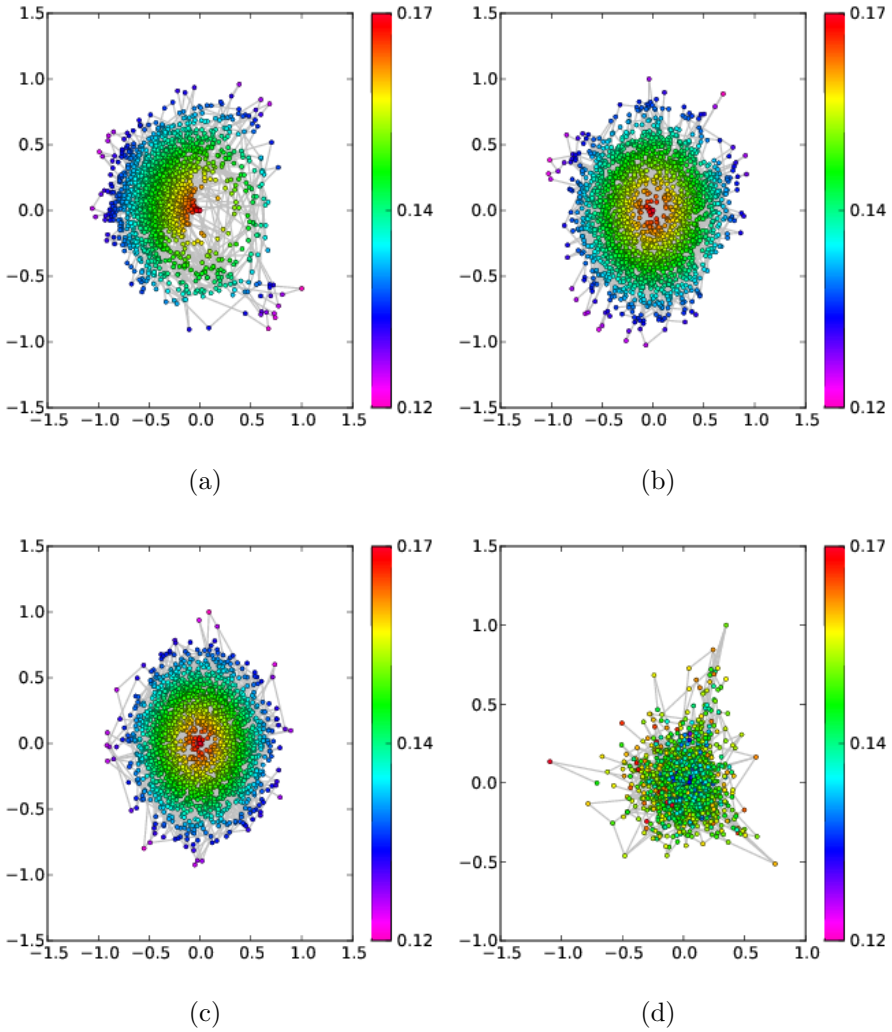


FIGURE 10.2.1: Graph embeddings for a Watts-Strogatz graph with $N = 2,000$: (a) Centrality-constrained embedding with $\mathbf{K}_1 = (-1/2)\mathbf{J}\boldsymbol{\Delta}^{(2)}\mathbf{J}$; (b) Centrality-constrained embedding with $\mathbf{K}_2 = \mathbf{A}\mathbf{A}^T$; (c) Centrality-constrained embedding with $\mathbf{K}_3 = \mathbf{L}^\dagger$; and (d) Centrality-agnostic embedding based on kernel matrix $\mathbf{K}_1$. The color bar maps node colors to varying centrality values.

Figure 10.2.1 depicts visualizations of the Watts-Strogatz network obtained by setting $N = 2,000$. In Figure 10.2.1 (a), (b), and (c), centrality-constrained embeddings are plotted with kernel matrices $\mathbf{K}_1 = (-1/2)\mathbf{J}\boldsymbol{\Delta}^{(2)}\mathbf{J}$,

$\mathbf{K}_2 = \mathbf{A}\mathbf{A}^T$, and $\mathbf{K}_3 = \mathbf{L}^\dagger$, respectively. The appeal for centrality-constrained embeddings is clear when compared with Figure 10.2.1(d), which depicts a "centrality-agnostic" graph embedding using $\mathbf{K}_1$. Here, the final weight preservation step entailed spectral decomposition of $(\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W})$, consistent with the original LLE algorithm. It is clear that even for a moderately sized synthetic graph, little meaningful information can be conveyed visually from the centrality-agnostic embedding. For instance, it is not obvious how an analyst would discern which nodes are most accessible to peers, or those whose removal would compromise the rate of information propagation over the network.

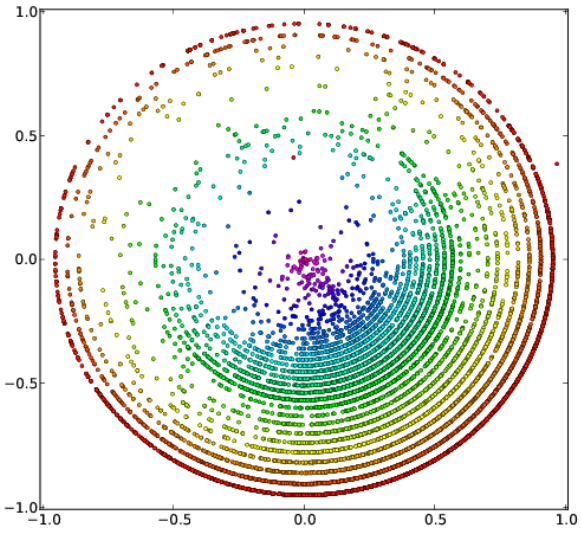
Figure 10.2.2 depicts visualizations of snapshots of the Gnutella peer-to-peer file-sharing network [LKF07]. Directed edges represent connections between hosts. The snapshots were captured on Aug. $4, 2012$ ($N = 10,876$, $|\mathcal{E}| = 39,994$) and Aug. $24, 2012$ ($N = 26,518$, $|\mathcal{E}| = 65,369$), respectively. The adjacency matrices were symmetrized to obtain undirected versions of the network. In this case $\mathbf{K} = \mathbf{A}\mathbf{A}^T$ and $\mathcal{C}$ was set to the node degrees. It is clear from the network drawings that despite the dramatic growth in the number of edges over a 20 day span, most new nodes had low degree, and are located far from the center.

10.2.4 Visualization of dynamic social networks

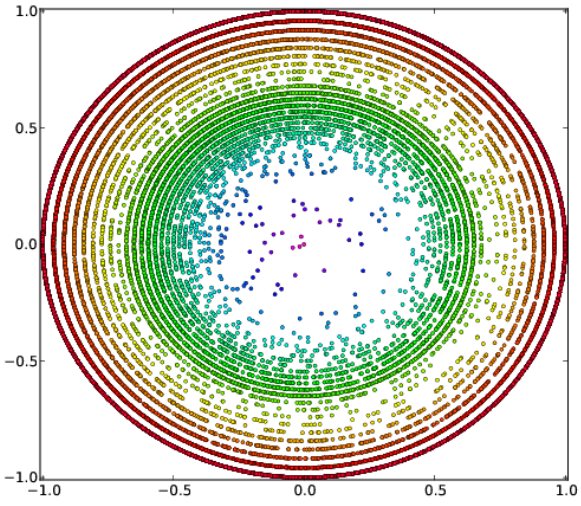
This section has so far focused on addressing issues concerning structurally-informative visualization of large static networks. However, these issues are exacerbated in settings involving dynamic networks whose topologies evolve over time [BW97, BW98, BC03, MMBd05, MMBd06, LS08, FT08]. Typically, one is given a time series of graphs $\{G_t\}_{t=1}^T$, representing static snapshots indexed by time intervals $t = 1, \ldots, T$. Although it is possible to compute a sequence of static embeddings, this approach does not scale to big data where graph snapshots may encode millions of nodes, and may be acquired in a streaming fashion.

Moreover, the sequence of embeddings must "respect" the viewer's *mental map*, which captures a sense of stability of the underlying structure of the network across time intervals. Sequential embeddings with drastic changes for a significant number of node positions violate a viewer's expectations, rendering mental reconciliation of temporal network changes hard. Pioneering works have advocated a Bayesian framework that facilitates modeling dependencies between consecutive graph layouts [BW97, BW98]. A more recent visualization framework advocates *regularized graph layouts*, which involve optimizing a cost function augmented with a penalty that enforces stability across timeslots [XKI12].

Instead of generating a sequence of stable embeddings, a recent approach promotes static displays that capture temporal variations using non-negative matrix factorization for dimensionality reduction [MM13]. It tacitly assumes that the number of nodes remains fixed but edge connections vary with time.



(a) Gnutella-04 (08/04/2012)



(b) Gnutella-24 (08/24/2012)

FIGURE 10.2.2: Visualization of two snapshots of the large-scale file-sharing network Gnutella [LKF07] based on degree centrality and $\mathbf{K} = \mathbf{A}\mathbf{A}^T$.

Static visual plots of rank-one matrix factors capture the temporal evolution of node importance. Despite numerous efforts, most contemporary approaches are not tailored for big networks with streaming (and possibly missing) inputs, a setup of growing research activity.

10.3 Inference and imputation on social graphs

Inference over graphs is a rich subject, and includes the typical (non-) parametric regression, classification, and clustering tasks. Here, we will focus on anomaly detection, interpolation (a.k.a. imputation), and extrapolation (a.k.a. prediction).

10.3.1 Distributed anomaly detection for social graphs

This section deals with graph (network) anomaly identification. On the one hand, existing approaches have dealt with the pursuit of abnormal behavior exhibited by processes that evolve over graphs, such as Internet traffic flows [LCD04, ZGGR05, MMG13b, MMG13c, MR13], or spatiotemporal energy consumption profiles [CLL$^+$10, MG12a, MG13], to name a couple. But also of great interest for spam, fraud and network intrusion detection is to consider the structure of the underlying graphs, and determine whether e.g., nodes, edges, or egonets are anomalous in the sense that they deviate from postulated network models; see e.g., [NC03, SQCF05, EH07, MT08, TL11, AMF12, MAB13] for noteworthy contributions. In the sequel, a novel approach to social graph anomaly detection is outlined, which is based on contemporary low-rank plus sparse matrix decompositions. A distributed algorithm is then developed leveraging the alternating-directions method of multipliers (ADMM); see e.g., [BT99, BPC$^+$11].

10.3.1.1 Anomaly detection via sparse plus low-rank decomposition

The idea here is to consider the egonets in a social graph (i.e., each node's induced single-hop subgraph), and for each of them evaluate structural features or graph invariants such as, average degree, number of nodes, principal eigenvalue and average clustering coefficient, to name a few. It is thus possible to collect all these structural quantities in a feature×egonet matrix $\mathbf{Y}$. Specifically, let the $D \times 1$ vector $\mathbf{y}_n := [y_{1,n}, \ldots, y_{D,n}]^T$ collect D different graph features, calculated for each egonet $n \in [1, N]$ in a graph of N vertices. Consider the $D \times N$ graph feature matrix $\mathbf{Y} := [\mathbf{y}_1, \ldots, \mathbf{y}_N]$. The d-th row $\mathbf{y}^T(d)$ of $\mathbf{Y}$ is the networkwide sequence corresponding to feature d, measured for all egonets in the graph. It may be useful to explicitly account for missing data, which

could arise because not all features can be calculated for all egonets, or when a subsampling strategy is implemented to reduce the computational complexity in forming $\mathbf{Y}$. To this end, consider the set $\Omega \subseteq \{1,\ldots,D\} \times \{1,\ldots,N\}$ of index pairs (d,n) defining a sampling pattern (or mask) of the entries of $\mathbf{Y}$. Introducing the matrix sampling operator $\mathcal{P}_\Omega(\cdot)$, which sets the entries of its matrix argument not indexed by Ω to zero and leaves the rest unchanged, the (possibly) incomplete matrix of egonet features in the presence of outliers can be modeled as:

$$\mathcal{P}_\Omega(\mathbf{Y}) = \mathcal{P}_\Omega(\mathbf{X} + \mathbf{O} + \mathbf{E}), \tag{10.3.1}$$

where $\mathbf{X}$, $\mathbf{O}$, and $\mathbf{E}$ denote the nominal feature matrix, the outliers (i.e., anomalies), and small approximation errors, respectively. For nominal egonet features $y_{d,n} = x_{d,n} + e_{d,n}$, one has no anomaly; that is $o_{d,n} = 0$.

The model is inherently under-determined, since even for the (most favorable) case of full data, i.e., $\Omega \equiv \{1,\ldots,D\} \times \{1,\ldots,N\}$, there are twice as many unknowns in $\mathbf{X}$ and $\mathbf{O}$ as there is data in $\mathbf{Y}$. Estimating $\mathbf{X}$ and $\mathbf{O}$ becomes even more challenging when data are missing, since the number of unknowns remains the same, but the amount of data is reduced. In any case, estimation of $\{\mathbf{X},\mathbf{O}\}$ from $\mathcal{P}_\Omega(\mathbf{Y})$ is an ill-posed problem unless one introduces extra structural assumptions on the model components to reduce the effective degrees of freedom. To this end, two cardinal properties of $\mathbf{X}$ and $\mathbf{O}$ will prove instrumental. First, a key observation is that for "nominal" complex networks most of these features obey power laws [FFF99, Kol09, EK10], and hence $\mathbf{Y}$ (or its entrywise logarithm) will be approximately *low rank*. Second, anomalies (or outliers) only occur sporadically across egonets and features, yielding a *sparse* matrix $\mathbf{O}$.

An estimator matching nicely the specifications of the graph anomaly detection problem stated, is the so-termed "robust" (stable) principal components pursuit (PCP, also know as RPCA for robust principal component analysis) [ZLW$^+$10, CLMW11, CSPW11, MG12b], that will be outlined here for completeness. PCP seeks estimates $\{\hat{\mathbf{X}},\hat{\mathbf{O}}\}$ as the minimizers of:

$$\text{(P1)} \qquad \min_{\{\mathbf{X},\mathbf{O}\}} \|\mathcal{P}_\Omega(\mathbf{Y}-\mathbf{X}-\mathbf{O})\|_F^2 + \lambda_* \|\mathbf{X}\|_* + \lambda_1 \|\mathbf{O}\|_1 ,$$

where the ℓ_1-norm $\|\mathbf{O}\|_1 := \sum_{d,n} |o_{d,n}|$ and the nuclear norm $\|\mathbf{X}\|_* := \sum_i \sigma_i(\mathbf{X})$ ($\sigma_i(\mathbf{X})$ denotes the i-th singular value of $\mathbf{X}$) are utilized to promote sparsity in the number of outliers (nonzero entries) in $\mathbf{O}$, and the low rank of $\mathbf{X}$, respectively. The nuclear and ℓ_1-norms are the closest convex surrogates to the rank and cardinality functions, which albeit the most natural criteria they are in general NP-hard to optimize [CG84, Nat95]. The tuning parameters $\lambda_1, \lambda_* \geq 0$ control the tradeoff between fitting error, rank, and sparsity level of the solution. When an estimate $\hat{\sigma}_v^2$ of the noise variance is available, guidelines for selecting λ_* and λ_1 have been proposed in [ZLW$^+$10].

The location of nonzero entries in $\hat{\mathbf{O}}$ reveals "anomalies" across both features and egonets, while their amplitudes quantify the magnitude of deviation.

Clearly, it does not make sense to flag outliers in data that has not been observed, namely for $(d, n) \notin \Omega$. In those cases (P1) yields $\hat{o}_{d,n} = 0$ since both the Frobenius and ℓ_1-norms are separable across the entries of their matrix arguments.

A numerical test on the arXiv General Relativity and Quantum Cosmology collaboration social graph [Les11] is depicted in Figure 10.3.1. The graph has $5,242$ vertices (authors) and $14,496$ edges (indicating that two authors collaborated on at least one paper), and there is no missing data. The red egonet is flagged as anomalous, and it is apparent that edge density (number of collaborations) is markedly larger than e.g., the other highlighted peers (purple, green and magenta egonets). Beyond egonets, it is also possible to devise algorithms to unveil anomalous graphs at a macro level. To this end, the relevant graph invariants (rows of $\mathbf{Y}$) should be evaluated for the whole network using e.g., the scalable graph minining package Pegasus [KTF09], and across networks (columns of $\mathbf{Y}$) for all those graphs of interest in the analysis.

Being convex (P1) is computationally appealing, and it has been shown to attain good performance in theory and practice. For instance, in the absence of noise and when there is no missing data, identifiability and exact recovery conditions were reported in [CLMW11] and [CSPW11]. Even when data are missing, it is possible to recover the low-rank component under some technical assumptions [CLMW11]. Theoretical performance guarantees in the presence of noise are also available [ZLW+10]. Regarding *batch centralized* algorithms, a PCP solver based on the accelerated proximal gradient method was put forth in [LGW+11, MMG13c], while the ADMM was employed in [YY13, MMG13c]. For a single but *dynamic* network, detection of structural changes in time can be naturally accommodated if the feature vectors (now time-indexed columns of $\mathbf{Y}$) are recalculated per time slot, and processed on-the-fly using online graph algorithms for streaming data; see also [PPY13, MAB13, MMG13b, WTPP14].

10.3.1.2 In-network processing algorithm

Increasingly-large graphs and computational challenges arising with big data motivate well devising *fully-distributed* iterative algorithms for unveiling anomalies in social graphs. In a nutshell, per iteration $k = 1, 2, \ldots$ nodes n (i.e., graph vertices) carry out simple computational tasks locally, relying on their own local feature vectors $\mathbf{y}_n$. Subsequently, local estimates are refined after exchanging messages only with directly connected neighbors in the vertex set $\mathcal{N}_n$, which facilitates percolation of local information to the whole network. The end goal is for each node to form local estimates $\mathbf{x}_n[k]$ and $\mathbf{o}_n[k]$ that coincide with the n-th columns of $\hat{\mathbf{X}}$ and $\hat{\mathbf{O}}$ as $k \to \infty$, where $\{\hat{\mathbf{X}}, \hat{\mathbf{O}}\}$ is the solution of (P1) obtained when all data $\mathcal{P}_\Omega(\mathbf{Y})$ are centrally available.

In its present form (P1) is not amenable for distributed implementation due to the non-separable nuclear norm present in the cost function. If an upper bound $\text{rank}(\hat{\mathbf{X}}) \leq \rho$ is a priori available, (P1)'s search space is effectively

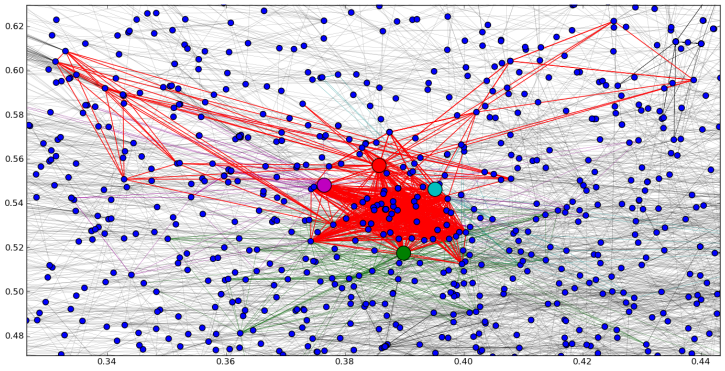


FIGURE 10.3.1: An anomalous (depicted in red) egonet for the arXiv General Relativity and Quantum Cosmology collaboration social graph [Les11]. The red egonet is flagged as anomalous by the proposed low-rank plus sparse matrix decomposition method.

reduced and one can factorize the decision variable as $\mathbf{X} = \mathbf{P}\mathbf{Q}^T$, where $\mathbf{P}$ and $\mathbf{Q}$ are $D \times \rho$ and $N \times \rho$ matrices, respectively. Next, consider the following alternative characterization of the nuclear norm (see e.g. [SS05, SRJ04, RR13]):

$$\begin{aligned} \|\mathbf{X}\|_* := \min_{\{\mathbf{P},\mathbf{Q}\}} \quad & \frac{1}{2}\left(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2\right) \\ \text{s. t.} \quad & \mathbf{X} = \mathbf{P}\mathbf{Q}^T, \end{aligned} \tag{10.3.2}$$

where the optimization is over all possible bilinear factorizations of $\mathbf{X}$, so that the number of columns ρ of $\mathbf{P}$ and $\mathbf{Q}$ is also a variable. Leveraging (10.3.2), the following equivalent reformulation of (P1) provides an important first step towards obtaining a distributed algorithm for graph anomaly detection

$$\min_{\{\mathbf{P},\mathbf{Q},\mathbf{A}\}} \sum_{n=1}^{N} \left[\|\mathcal{P}_{\Omega_n}(\mathbf{y}_n - \mathbf{P}\mathbf{q}_n - \mathbf{o}_n)\|^2 + \frac{\lambda_*}{2N}\left(\|\mathbf{P}\|_F^2 + N\|\mathbf{q}_n\|^2\right) + \lambda_1\|\mathbf{o}_n\|_1\right], \tag{10.3.3}$$

which is non-convex due to the bilinear terms $\mathbf{x}_n = \mathbf{P}\mathbf{q}_n$, and where $\mathbf{Q}^T := [\mathbf{q}_1, \ldots, \mathbf{q}_N]$. Adopting the separable Frobenius-norm regularization in (10.3.3) comes with no loss of optimality relative to (P1), provided $\text{rank}(\hat{\mathbf{X}}) \leq \rho$. By finding the global minimum of (10.3.3) [which could have considerably less variables than (P1)], one can recover the optimal solution of (P1). But since (10.3.3) is non-convex, it may have stationary points which need not be globally optimum. Interestingly, as asserted in [MMG13a, Prop. 1] if a stationary point $\{\bar{\mathbf{P}}, \bar{\mathbf{Q}}, \bar{\mathbf{O}}\}$ of (10.3.3) satisfies $\|\mathcal{P}_\Omega(\mathbf{Y} - \bar{\mathbf{P}}\bar{\mathbf{Q}}^T - \bar{\mathbf{O}})\| < \lambda_*$, then $\{\hat{\mathbf{X}} := \bar{\mathbf{P}}\bar{\mathbf{Q}}^T, \hat{\mathbf{O}} := \bar{\mathbf{O}}\}$ is the globally optimal solution of (P1).

To decompose the cost in (10.3.3), in which summands inside the square brackets are coupled through the global variable $\mathbf{P}$, introduce auxiliary copies $\{\mathbf{P}_n\}_{n=1}^N$ representing local estimates of $\mathbf{P}$, one per node n. These local copies along with *consensus* constraints yield the distributed estimator:

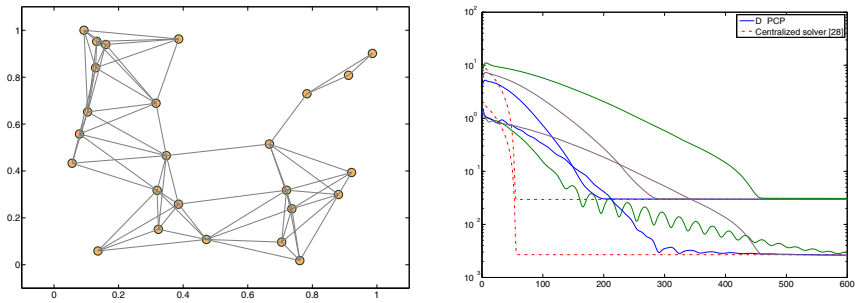
$$\begin{aligned}\min_{\{\mathbf{P}_n,\mathbf{q}_n,\mathbf{o}_n\}} \sum_{n=1}^{N} \big[&\|\mathcal{P}_{\Omega_n}(\mathbf{y}_n - \mathbf{P}_n\mathbf{q}_n - \mathbf{o}_n)\|^2 \\ &+ \frac{\lambda_*}{2N}\left(\|\mathbf{P}_n\|_F^2 + N\|\mathbf{q}_n\|^2\right) + \lambda_1\|\mathbf{o}_n\|_1\big] \\ \text{s. t.}\quad &\mathbf{P}_n = \mathbf{P}_m,\ \ m \text{ linked with } n \in \mathcal{N},\end{aligned} \tag{10.3.4}$$

which is equivalent to (10.3.3) provided the network topology graph is connected. Even though consensus is a fortiori imposed within neighborhoods, it extends to the whole (connected) network, and local estimates agree on the global solution of (10.3.3). Exploiting the separable structure of (10.3.4), a general framework for in-network sparsity-regularized rank minimization was put forth in [MMG13a], whereas a distributed algorithm for PCP (D-PCP) can be found in [MG13]. Specifically, distributed iterations were obtained after adopting the ADMM, an iterative Lagrangian method well-suited for parallel processing [BT99, BPC+11]. In a nutshell, local tasks per iteration $k = 1, 2, \ldots$ entail solving small unconstrained quadratic programs to refine the projections $\mathbf{q}_n[k]$ on the nominal feature subspace $\mathbf{P}_n[k]$, in addition to soft-thresholding operations to update the egonet anomaly vectors $\mathbf{o}_n[k]$ per node; see [MG13] for further details. Per iteration, graph nodes exchange their subspace estimates $\mathbf{P}_n[k]$ only with directly connected neighbors. This way the communication overhead stays affordable, and independent of the network size N.

When employed to solve non-convex problems such as (10.3.4), so far ADMM offers no convergence guarantees. However, there is ample experimental evidence in the literature that supports empirical convergence of ADMM, especially when the non-convex problem at hand exhibits "favorable" structure. For instance, (10.3.4) is a linearly constrained bi-convex problem with potentially good convergence properties – extensive numerical tests in [MG13, MMG13a] demonstrate that this is indeed the case. While establishing convergence remains an open problem, one can still prove that upon convergence the distributed iterations attain consensus and global optimality, offering the desirable centralized performance guarantees [MMG13a].

10.3.1.3 Numerical tests

A test social network of $N = 25$ nodes is generated as a realization of the random geometric graph model, meaning nodes are randomly placed on the unit square and two nodes communicate with each other if their Euclidean distance is less than a prescribed communication range of 0.4; see Figure 10.3.2 (a). The number of egonet features is $D = 20$. Entries of $\mathbf{E}$ are independent and identically distributed (i.i.d.), zero-mean, Gaussian with variance $\sigma^2 = 10^{-3}$;



(a) (b)

FIGURE 10.3.2: (a) A simulated small social network graph with $N = 25$ nodes. (b) Convergence of the D-PCP algorithm for different network sizes. D-PCP attains the same estimation error as the centralized solver.

i.e., $e_{d,n} \sim \mathcal{N}(0, \sigma^2)$. A simulated nominal egonet feature matrix with rank $r = 3$ is generated from the bilinear factorization model $\mathbf{X} = \mathbf{W}\mathbf{Z}^T$, where $\mathbf{W}$ and $\mathbf{Z}$ are $D \times r$ and $N \times r$ matrices with i.i.d. entries drawn from Gaussian distributions $\mathcal{N}(0, 100/D)$ and $\mathcal{N}(0, 100/N)$, respectively. Every entry of $\mathbf{O}$ is randomly drawn from the set $\{-1, 0, 1\}$ with $\Pr(o_{d,n} = -1) = \Pr(o_{d,n} = 1) = 5 \times 10^{-2}$. To simulate missing data, a sampling matrix $\mathbf{\Omega} \in \{0, 1\}^{D \times N}$ is generated with i.i.d. Bernoulli distributed entries $o_{d,n} \sim \text{Ber}(0.7)$ (30% missing data on average). Finally, measurements are generated as $\mathcal{P}_\Omega(\mathbf{Y}) = \mathbf{\Omega} \odot (\mathbf{X} + \mathbf{O} + \mathbf{E})$ [cf. (10.3.1)], and node n has available the n-th column $\mathbf{y}(n)$ of $\mathcal{P}_\Omega(\mathbf{Y})$.

To experimentally corroborate the convergence and optimality of the D-PCP algorithm for graph anomaly detection, the distributed iterations are run and compared with the centralized benchmark (P1), obtained using the solver in [YY13]. Parameters $\lambda_1 = 0.0141$ and $\lambda_* = 0.346$ are chosen as suggested in [ZLW+10]. For both schemes, Figure 10.3.2b shows the evolution of the global estimation errors $e_X[k] := \|\mathbf{X}[k] - \mathbf{X}\|_F / \|\mathbf{X}\|_F$ and $e_O[k] := \|\mathbf{O}[k] - \mathbf{O}\|_F / \|\mathbf{O}\|_F$. It is apparent that the D-PCP algorithm converges to the centralized estimator, and as expected convergence slows down due to the delay associated with the information flow throughout the network. The test is also repeated for network sizes of $N = 15$ and 35, to illustrate that the time till convergence scales gracefully as the network size increases.

10.3.2 Prediction from partially-observed network processes

Understanding the influence of topology on network processes has relied on measuring and monitoring the network itself. In practice however, gathering network-wide measurements scales poorly with the network size and may thus be impractical for various networks of interest. For instance, large social network surveys also pose a major logistic issue due to, for example, limited

availability of individuals included in the survey. Tools such as `tcpdump` provide detailed packet-level information in Internet protocol (IP) networks, but collecting all these data can demand excessive power and bandwidth. Moreover, errors due to measurement and data-handling are more likely to emerge as the amount of data collected increases. A similar challenge arises when capturing spatial and temporal structures in big data. There, one may be forced to rely on partial (random) observations of the data so that inference algorithms remain operational while coping with the data deluge. With these motivating challenges in mind, this section surveys a recent joint topology- and data-driven algorithm to enable network-wide prediction of dynamical processes based on partial network observations, that is, measurements collected only at a subset of network nodes [FRG14]. The known (graph-induced) network structure and historical data are leveraged to design a dictionary for representing the network process. The novel approach draws from semi-supervised learning to enable learning the dictionary with only partial network observations. Once the dictionary is learned, network-wide prediction becomes possible via a regularized LS estimate which exploits the parsimony encapsulated in the design of the dictionary.

Consider an undirected weighted graph $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the vertex set with cardinality $N = |\mathcal{V}|$ and $\mathcal{E}$ is the edge set. The connectivity and edge strenghts of G are characterized by the weighted adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, where the entry $a_{i,j} := [\mathbf{A}]_{i,j} > 0$ if nodes v_i and v_j are connected, and $a_{i,j} = 0$ otherwise. At time instant $t \in \mathbb{N}$, corresponding to each vertex $v_n \in \mathcal{V}$ there is a scalar variable $x_{n,t} \in \mathbb{R}$, which represents the network-wide dynamical process of interest. All node variables are collected in a single vector $\mathbf{x}_t := [x_{1,t} \dots x_{N,t}]^T \in \mathbb{R}^N$. To account for missing data, it is assumed that $M < N$ vertices are measured at any given time. For simplicity in exposition, the number of observed vertices M is assumed fixed. However, expressions and algorithms derived in the subsequent sections can be readily modified to allow for time-varying M. Let $\mathbf{M}_t \in \mathbb{R}^{M \times N}$ denote a binary measurement matrix with $0-1$ entries selecting the measured components of $\mathbf{x}_t$. Each row of $\mathbf{M}_t$ corresponds to a vector of the canonical basis for $\mathbb{R}^N$, i.e., each row has only one nonzero entry, which takes the value of 1, while all other entries are set to 0. The $M \times 1$ measurement vector at time t is modeled as:

$$\mathbf{y}_t = \mathbf{M}_t \mathbf{x}_t + \boldsymbol{\epsilon}_t, \quad t = 1, 2, \dots, \tag{10.3.5}$$

where $\boldsymbol{\epsilon}_t$ is a random error term capturing measurement imperfections.

Recently, a network process prediction algorithm was put forth in [FRG14], where missing entries of $\mathbf{x}_t$ are estimated from historical measurements in $\mathcal{T}_M := \{\mathbf{y}_t\}_{t=1}^T$ by leveraging the structural regularity of $\mathbf{x}_t$ (induced by the underlying graph) through a semi-supervised dictionary learning (DL) approach. Under the DL framework, *data-driven* dictionaries for *sparse* signal representation are adopted as a versatile means of capturing parsimonious signal structures; see e.g., [TF10] for a tutorial treatment. Propelled by the success of compressive sampling (CS) [Don06], sparse signal model-

ing has led to major advances in several machine learning, audio and image processing tasks [HTF09, TF10]. Motivated by these ideas, it is postulated in [FRG14] that graph signals can be represented as a linear combination $\mathbf{x}_t = \mathbf{B}\mathbf{s}_t$ of a few ($\ll Q$) columns of an over-complete dictionary (basis) matrix $\mathbf{B} := [\mathbf{b}_1, \ldots, \mathbf{b}_Q] \in \mathbb{R}^{N \times Q}$, where $\mathbf{s}_t \in \mathbb{R}^Q$ is a *sparse* vector of expansion coefficients. Many signals including speech and natural images admit sparse representations even under generic predefined dictionaries, such as those based on the Fourier and the wavelet bases, respectively [TF10]. Like audio and natural images, vertex variables can exhibit strong correlations induced from the structure of the underlying graph. For instance, Internet traffic volumes on two links are highly correlated if they both carry common end-to-end flows, as indicated by the corresponding routing matrix. DL schemes are attractive due to their flexibility, since they utilize training data to *learn* an appropriate over-complete basis customized for the data at hand. However, the use of DL for modeling network data is well motivated but so far relatively unexplored.

10.3.2.1 Semi-supervised prediction of network processes

Suppose for now that either a learnt, or, a suitable pre-specified dictionary $\mathbf{B}$ is available, and consider predicting the process on the unobserved vertices. Data-driven learning of dictionaries from historical data will be addressed in the ensuing subsection. To cope with the absence of some entries of $\mathbf{x}_t$ not present in $\mathbf{y}_t$, the idea here is to capitalize on the topology of G. To that end, suppose $w_{i,j}$ represents a similarity weight between the time-dependent variables associated with nodes v_i and v_j; e.g., the correlation between $x_{i,t}$ and $x_{j,t}$. The topology of G, and thus the spatial correlation of the process, is captured by its Laplacian matrix $\mathbf{L} := \text{diag}(\mathbf{A}\mathbf{1}_N) - \mathbf{A}$. Given $\mathbf{B}$, $\mathbf{L}$ and the measurements $\mathbf{y}_t$, contemporary tools developed in the area of CS and semi-supervised learning can be used to form $\hat{\mathbf{x}}_t$, which includes estimates for the missing $N - M$ vertex observations [Don06, BNS06, HTF09].

Given a snapshot of incomplete measurements $\mathbf{y}_t$ during the *operational phase* (where a suitable basis $\mathbf{B}$ is available), the sparse basis expansion coefficient vector $\mathbf{s}_t$ is estimated as:

$$\hat{\mathbf{s}}_t := \arg\min_{\mathbf{s}_t} \|\mathbf{y}_t - \mathbf{M}_t\mathbf{B}\mathbf{s}_t\|_2^2 + \lambda_s\|\mathbf{s}_t\|_1 + \lambda_w\mathbf{s}_t^T\mathbf{B}^T\mathbf{L}\mathbf{B}\mathbf{s}_t, \tag{10.3.6}$$

where λ_s and λ_w are tunable regularization parameters. The criterion in (10.3.6) consists of a LS error between the observed and postulated network measurements, along with two regularizers. The ℓ_1-norm $\|\mathbf{s}_t\|_1$ encourages sparsity in the coefficient vector $\hat{\mathbf{s}}_t$ [Don06, HTF09]. With $\mathbf{x}_t := [x_{1,t}, \ldots, x_{N,t}]^T$ given by $\mathbf{x}_t = \mathbf{B}\mathbf{s}_t$, the Laplacian regularization can be explicitly written as $\mathbf{s}_t^T\mathbf{B}^T\mathbf{L}\mathbf{B}\mathbf{s}_t = (1/2)\sum_{i=1}^N\sum_{j=1}^N a_{i,j}(x_{i,t} - x_{j,t})^2$. It is thus apparent that $\mathbf{s}_t^T\mathbf{B}^T\mathbf{L}\mathbf{B}\mathbf{s}_t$ encourages the vertex variables to be close if their corresponding weights are large. Typically adopted for semi-supervised learning, such a regularization term encourages $\mathbf{B}\mathbf{s}_t$ to lie on a smooth manifold approximated by G, which constrains how the measurements relate to

$\mathbf{x}_t$ [BNS06, RBL$^+$07]. It is also common to use normalized variants of the Laplacian instead of $\mathbf{L}$ [Kol09, p. 46].

The cost in (10.3.6) is convex but non-smooth, and customized solvers developed for ℓ_1-norm regularized optimization can be employed here as well, e.g., [HTF09, p. 92]. Once $\hat{\mathbf{s}}_t$ is available, an estimate of the full vector of network samples is readily obtained as $\hat{\mathbf{x}}_t := \mathbf{B}\hat{\mathbf{s}}_t$. It is apparent that the quality of the imputation depends on the chosen $\mathbf{B}$, and DL from historical network data in $\mathcal{T}_M$ is described next.

10.3.2.2 Data-driven dictionary learning

In its canonical form, DL seeks a (typically fat) dictionary $\mathbf{B}$ so that training data $\mathcal{T}_N := \{\mathbf{x}_t\}_{t=1}^T$ are well approximated as $\mathbf{x}_t \approx \mathbf{B}\mathbf{s}_t$, $t = 1, \ldots, T$, for some sparse vectors $\mathbf{s}_t$ of expansion coefficients [TF10]. Standard DL algorithms cannot, however, be directly applied to learn $\mathbf{B}$ since they rely on the entire vector $\mathbf{x}_t$. To learn the dictionary in the *training phase* using incomplete measurements $\mathcal{T}_M$ instead of $\mathcal{T}_N$, the idea is to capitalize on the structure in $\mathbf{x}_t$, of which G is an abstraction [FRG14]. To this end, one can adopt a similar cost function as in the operational phase [cf. (10.3.6)], yielding the data-driven basis and the corresponding sparse representation:

$$\{\hat{\mathbf{S}}, \hat{\mathbf{B}}\} := \underset{\mathbf{S},\mathbf{B}:\{\|\mathbf{b}_q\|_2 \leq 1\}_{q=1}^Q}{\arg\min} \sum_{t=1}^{T} \left[\|\mathbf{y}_t - \mathbf{M}_t\mathbf{B}\mathbf{s}_t\|_2^2 + \lambda_s\|\mathbf{s}_t\|_1 + \lambda_w \mathbf{s}_t^T\mathbf{B}^T\mathbf{L}\mathbf{B}\mathbf{s}_t\right], \tag{10.3.7}$$

where $\hat{\mathbf{S}} := [\hat{\mathbf{S}}_1, \ldots, \hat{\mathbf{S}}_T] \in \mathbb{R}^{Q\times T}$. The constraints $\{\|\mathbf{b}_q\|_2 \leq 1\}_{q=1}^Q$ remove the scaling ambiguity in the products $\mathbf{B}\mathbf{s}_t$, and prevent the entries in $\mathbf{B}$ from growing unbounded. Again, the combined regularization terms in (10.3.7) promote both sparsity in $\mathbf{s}_t$ through the ℓ_1-norm, and smoothness across the entries of $\mathbf{B}\mathbf{s}_t$ via the Laplacian $\mathbf{L}$. The regularization parameters λ_s and λ_w are typically cross-validated [HTF09, Ch. 7]. Although (10.3.7) is non-convex, a BCD solver still guarantees convergence to a stationary point [BT99]. The BCD updates involve solving for $\mathbf{B}$ and $\mathbf{S}$ in an alternating fashion, both doable efficiently via convex programming [FRG14]. Alternatively, the online DL algorithm in [MBPS10] offers enhanced scalability by sequentially processing the data in $\mathcal{T}_S$. The training and operational (prediction) phases are summarized in Figure 10.3.3, where $C_t(\mathbf{B}, \mathbf{s})$ denotes the t-th summand from the cost in (10.3.7), and $k = 1, 2, \ldots$ indicate iterations of the BCD solver employed during the training phase.

The explicit need for Laplacian regularization is apparent from (10.3.7). Indeed, if measurements from a certain vertex are not present in $\mathcal{T}_M$, the corresponding row of $\mathbf{B}$ may still be estimated with reasonable accuracy because of the third term in $C_t(\mathbf{B}, \mathbf{s})$. On top of that, it is because of Laplacian regularization that the prediction performance degrades gracefully as the number of missing entries in $\mathbf{y}_t$ increases; see also Figure 10.3.4. It is worth stressing that the time series $\{\mathbf{y}_t\}$ need not be stationary or even contiguous in time. The

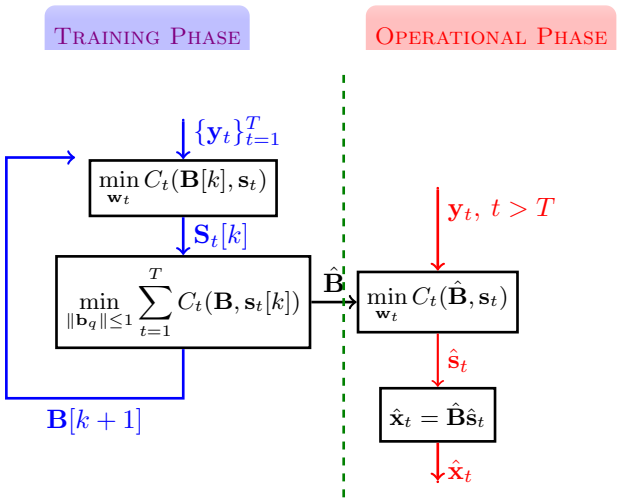


FIGURE 10.3.3: Training and operational phases of the semi-supervised DL approach for prediction of network processes that evolve over graphs [FRG14].

network-process prediction approach described so far can also be adapted to accommodate time-varying network topologies, using a time-dependent Laplacian $\mathbf{L}_t$. A word of caution is due however, since drastic changes in either $\mathbf{L}_t$ or in the statistical properties of the underlying process $\mathbf{x}_t$, will necessitate re-training $\mathbf{B}$ to attain satisfactory performance.

10.3.2.3 Numerical tests

Next, a numerical test on link count data from the Internet2 measurement archive [Int] is outlined. Consider an IP network comprising N nodes and L links, carrying the traffic of F origin-destination flows (network connections). Let $x_{l,t}$ denote the traffic volume (in bytes or packets) passing through link $l \in \{1, \ldots, L\}$ over a fixed interval of time $(t, t + \Delta t)$. Link counts across the entire network are collected in the vector $\mathbf{x}_t \in \mathbb{R}^L$, e.g., using the ubiquitous SNMP protocol. Since measured link counts are both unreliable and incomplete due to hardware or software malfunctioning, jitter, and communication errors [ZRWQ09, Rou10], they are expressed as noisy versions of a subset of $S < L$ links

$$\mathbf{y}_t = \mathbf{M}_t \mathbf{x}_t + \boldsymbol{\epsilon}_t, \quad t = 1, 2, \ldots,$$

where $\mathbf{M}_t$ is an $S \times L$ selection matrix with 0-1 entries whose rows correspond to rows of the identity matrix of size L, and $\boldsymbol{\epsilon}_t$ is an $S \times 1$ zero-mean noise term with constant variance accounting for measurement and synchronization errors. Given $\mathbf{y}_t$ the aim is to form an estimate $\hat{\mathbf{x}}_t$ of the full vector of link counts $\mathbf{x}_t$, which in this case defines the network state.

The data consists of link counts, sampled at 5 minute intervals, collected

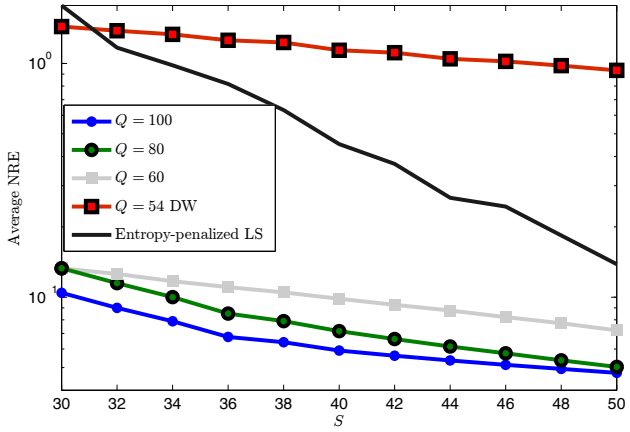


FIGURE 10.3.4: Link-traffic cartography of Internet2 data [Int]. Comparison of NRE for different values of S [FRG14].

over several weeks. For the purposes of comparison, the training phase consisted of $2,000$ time slots, with a random subset of 50 links measured (out of $L = 54$ per time slot). Performance of the learned dictionary is then assessed over the next $T_0 = 2,000$ time slots. Each test vector $\mathbf{y}_t$ is constructed by randomly selecting S entries of the full link count vector $\mathbf{x}_t$. The tuning parameters are chosen via cross-validation ($\lambda_s = 0.1$ and $\lambda_w = 10^{-5}$). Figure 10.3.4 shows the normalized reconstruction error (NRE), evaluated as $(LT_0)^{-1} \sum_{t=1}^{T_0} \|\mathbf{y}_t - \hat{\mathbf{x}}_t\|^2$ for different values of Q and S. For comparison, the prediction performance with a fixed diffusion wavelet matrix [CPR07] (instead of the data-trained dictionary), as well as that of the entropy-penalized LS method [ZRLD05] is also shown. The latter approach solves a LS problem augmented with a specific entropy-based regularizer, that encourages the traffic volumes at the source/destination pairs to be stochastically independent. The DL-based method markedly outperforms the competing approaches, especially for low values of S. Furthermore, note how performance degrades gracefully as S decreases. Remarkably, the predictions are close to the actual traffic even when using only 30 link counts during the prediction phase.

10.4 Unveiling communities in social networks

Social networks generally exhibit community structure, which is characterized by the existence of groups within which the edge density is relatively

high compared to the edge density between groups [GN02]. Communities are indicative of common functional roles or similar node behavior e.g., co-workers on Facebook, or followers of a cause on Twitter.

The community detection task has attracted significant attention from different disciplines, with several algorithms developed to tackle it. Traditional approaches resort to graph partitioning and data clustering algorithms such as k-means, hierarchical and spectral clustering [For10]. Graph partitioning algorithms such as Min-Cut and Ratio-Cut divide the graph into k parts of known size [For10]. However, these algorithms are not practical for community detection because they are limited to settings where the number and size of communities is known. On the other hand, hierarchical clustering, which is sensitive to selection of a similarity metric, is well-motivated as node similarity in social networks can be succinctly defined [HTF09].

Both k-means and spectral clustering are "workhorse" methods for (non)linearly separable data clustering. Nevertheless, spectral clustering is more appealing for community detection where the similarity graph is given. More recently, *modularity* methods have emerged, which entail optimization of a graph-clustering quality function [GN02]. Fortunato's community detection survey catalogs most of the contemporary community detection approaches [For10].

In general, most modern social networks are extremely large with millions of nodes and edges, and attempts to efficiently unveil their communities need to cope with typical big data challenges. To this end, a number of approaches based on parallelization, random sampling, and random projections have been advocated for clustering big data. Among these, parallelization is a relatively mature technology as tasks in k-means can be easily distributed over a computing cluster. However, randomized algorithms can reduce the computational load per node, and therefore require fewer nodes.

This section focuses on spectral clustering and its connections to the more general kernel k-means. In light of the aforementioned big data challenges, a discussion of recent random sampling extensions to such settings is given. It is also worth noting that the equivalence of spectral clustering with kernel k-means can prove useful to reduction of the computational load.

10.4.1 Big data spectral clustering

The emphasis of this subsection will be on big data spectral clustering. Although originally developed for data clustering, spectral clustering finds applications in community detection as it exhibits good performance in arbitrary cluster configurations. Spectral clustering exploits the properties of the similarity graph Laplacian $\mathbf{L}$ to group the vertices into a prescribed number of k clusters. Let $\mathbf{A} \in \mathbb{R}^{N \times N}$ denote the (weighted) adjacency matrix. The graph Laplacian is defined as $\mathbf{L} := \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is a diagonal matrix with $[\mathbf{D}]_{ii} = \sum_{j=1}^{N} [\mathbf{A}]_{ij}$, and $[\mathbf{A}]_{ij}$ denotes (i, j) entry of $\mathbf{A}$. The key property of $\mathbf{L}$ is the equivalence of algebraic multiplicity of the zero eigenvalue to

the number of connected components. The corresponding eigenvectors are the indicator vectors of each connected component. This can be verified by considering an eigenvalue $\lambda = 0$ of $\mathbf{L}$ and its corresponding eigenvector $\mathbf{v} \in \mathbb{R}^N$. Then:

$$\mathbf{Lv} = 0 \Rightarrow \mathbf{v}^T\mathbf{Lv} = 0 = \frac{1}{2}\sum_{i,j=1}^{N}[\mathbf{A}]_{ij}(v_i - v_j)^2, \tag{10.4.1}$$

with v_i denoting the i-th entry of $\mathbf{v}$. As all terms of the sum must vanish and $[\mathbf{A}]_{ij} \geq 0$, the entries of $\mathbf{v}$, for which $[\mathbf{A}]_{ij} \neq 0$ must be equal. Thus $\mathbf{v}$ should have constant entries corresponding to vertices of the connected component. In a network with k completely separated clusters, the graph will have k connected components, and hence $\mathbf{L}$ will have k zero eigenvalues. The corresponding eigenvectors suffice to reveal the clusters in the network. This however is not the case in social networks where the graph can be connected with communities linked to each other by a few edges. Thus, $\mathbf{L}$ will have a single all-ones ($\mathbf{1} \in \mathbb{R}^N$) eigenvector, and k eigenvalues close to zero. While the eigenvectors corresponding to these k eigenvalues will not be indicator vectors, they can still be used to separate the clusters.

Spectral clustering algorithms find the k smallest, non-zero, eigenvalues $\{\lambda_i\}_{i=1}^k$ of $\mathbf{L}$, and their corresponding eigenvectors $\{\mathbf{v}_i \in \mathbb{R}^N\}_{i=1}^k$. With $\mathbf{V} := [\mathbf{v}_1, ..., \mathbf{v}_k]$, vertex i is mapped to row i of $\mathbf{V}$. This change of representation enhances the separability of clusters in the graph, which can be recovered using simple algorithms such as k-means. The eigenvalue computation can be performed using efficient methods such as the power iteration [GL12]. Algorithm 10.4.1 depicts the unnormalized spectral clustering algorithm. In addition to the basic definition of $\mathbf{L}$, certain spectral clustering approaches leverage normalized graph Laplacians, which are equivalent to Min-Cut and Ratio-Cut [SM00, NJW+02].

Although the number of clusters is not necessarily known, one can deduce k by comparing the magnitudes of the eigenvalues of $\mathbf{L}$. Since eigenvalues corresponding to clusters are close to zero, one can assess the value of k by finding the "jump" in the spectrum of the eigenvalues.

Spectral clustering also has strong connections to kernel PCA [HTF09] and kernel k-means (Algorithm 10.4.2). Kernel k-means [DGK04] extends the classic k-means algorithm, and is able to cluster even non-linearly separable data. This is accomplished by mapping each datum to a higher-dimensional space $\mathcal{F}$, using a function $\boldsymbol{\phi} : \mathbb{R}^D \to \mathcal{F}$. The premise is that a mapping exists to render the dataset linearly separable, and hence amenable to simple and heuristic yet effective algorithms such as k-means. Even if $\mathcal{F}$ is infinite dimensional, the Representer theorem [Wah90] guarantees that inner products between data points on $\mathcal{F}$ suffice to perform clustering. Kernel k-means on a N-point dataset with k clusters aims to minimize the following objective function:

$$D = \sum_{j=1}^{k}\sum_{i=1}^{N}\|\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\mu}_{C_j}\|_2^2, \tag{10.4.2}$$

where $\boldsymbol{\phi}(\mathbf{x}_i)$ is the feature space representation of data point $\mathbf{x}_i$, and the centroid $\boldsymbol{\mu}_{C_j} := \frac{1}{|C_j|}\sum_{j\in C_j}\boldsymbol{\phi}(\mathbf{x}_j)$ is the sample mean of the points in cluster C_j. Using the Representer theorem the distance of each point in $\mathcal{F}$ from the centroid in (10.4.2) can be rewritten as:

$$\|\phi(\mathbf{x}_i) - \mu_C\|_2^2 = [\mathbf{K}]_{ii} - \frac{2}{|C|}\sum_{j\in C}[\mathbf{K}]_{ij} + \frac{1}{|C|^2}\sum_{j,l\in C}[\mathbf{K}]_{jl}, \tag{10.4.3}$$

where $\mathbf{K} \in \mathbb{R}^{N\times N}$ is the Gramian of the kernel used, and $[\mathbf{K}]_{ij}$ denotes the inner product between $\boldsymbol{\phi}(\mathbf{x}_i)$ and $\boldsymbol{\phi}(\mathbf{x}_j)$. Consequently, minimization of (10.4.2) can be written as:

$$\min_{\mathbf{U},\mathbf{C}} \ \operatorname{tr}(\mathbf{K}) - \operatorname{tr}(\mathbf{C}^{1/2}\mathbf{U}^T\mathbf{K}\mathbf{U}\mathbf{C}^{1/2}) \iff \max_{\hat{\mathbf{U}}} \ \operatorname{tr}(\hat{\mathbf{U}}^T\mathbf{K}\hat{\mathbf{U}}), \tag{10.4.4}$$

where $\mathbf{U} := [\mathbf{u}_i \dots \mathbf{u}_k]$ is a cluster membership matrix with $\mathbf{u}_i \in \{0,1\}^N$, $[\mathbf{u}_i]_j = 1$, if point j belongs to cluster i; the diagonal matrix $\mathbf{C} \in \mathbb{R}^{k\times k}$ collects inverses of cluster cardinalities, $\mathbf{C} = \operatorname{diag}(\frac{1}{|C_1|}, ..., \frac{1}{|C_k|})$; and, $\hat{\mathbf{U}} := \mathbf{C}^{1/2}\mathbf{U}$. The optimization problem in (10.4.4) is non-convex as $\hat{\mathbf{U}}$ is binary. By relaxing the binary constraint, and requiring $\hat{\mathbf{U}}^T\hat{\mathbf{U}} = \mathbf{I}$, the problem can be recast as the following convex surrogate with a well-known solution [GL12]:

$$\max_{\hat{\mathbf{U}}^T\hat{\mathbf{U}}=\mathbf{I}} \operatorname{tr}(\hat{\mathbf{U}}^T\mathbf{K}\hat{\mathbf{U}}) = \sum_{i=1}^{k}\lambda_i, \tag{10.4.5}$$

where $\hat{\mathbf{U}}^* = \mathbf{V}\mathbf{Q}$, $\{\lambda_i\}_{i=1}^k$ are the largest eigenvalues of $\mathbf{K}$, $\mathbf{Q} \in \mathbb{R}^{k\times k}$ denotes an arbitrary orthonormal matrix, and the columns of $\mathbf{V} \in \mathbb{R}^{N\times k}$ are formed with the k eigenvectors corresponding to $\{\lambda_i\}_{i=1}^k$. This is equivalent to finding the k trailing eigenvectors of $\mathbf{I} - \mathbf{K}$. Due to the relaxation, the columns of $\hat{\mathbf{U}}^*$ most likely do not represent natural clusterings, and as such post-processing is required. Similarly, Ratio-Cut and Min-Cut can be converted to trace maximization problems [Lux07]. Furthermore, kernel PCA finds the k largest eigenvectors of the centered Gramian $\tilde{\mathbf{K}}$.

Algorithm 10.4.1: Unnormalized spectral clustering

Require: k, $\mathbf{L} \in \mathbb{R}^{N\times N}$.
Ensure: Clustered vertices.
1: Compute the k smallest eigenvectors $\{\mathbf{v}_i\}_{i=1}^k$ of $\mathbf{L}$.
Let $\mathbf{V} := [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k] \in \mathbb{R}^{N\times k}$.
2: Let $\{\mathbf{x}_i \in \mathbb{R}^k\}_{i=1}^N$ be the rows of $\mathbf{V}$; $\mathbf{x}_i$ corresponds to the i-th vertex.
3: Group $\{\mathbf{x}_i\}_{i=1}^N$ into k clusters $\{C_i\}_{i=1}^k$.

In large social networks, $\mathbf{L}$ is presumed to be sparse. In this case, methods such as Arnoldi/Lanczos iterations [GL12] can be used to efficiently compute

Algorithm 10.4.2: Kernel k-means

Require: k, $\mathbf{K} \in \mathbb{R}^{N \times N}$, maximum number of iterations T
Ensure: Clustered points
1: Randomly assign points to clusters.
2: **repeat**
3: **for** $i = 1$ **to** N **do**
4: For point $\boldsymbol{\phi}(\mathbf{x}_i)$ calculate closest centroid using equation 10.4.3
5: **end for**
6: Update point assignments; Assign each point to the cluster whose centroid is closest
7: $t \leftarrow t + 1$; update iteration counter
8: **until** No changes in assignments or $t > T$

the trailing eigenspace of $\mathbf{L}$, as usually only matrix-vector products are required. Readily available packages that tackle large-scale sparse eigenvalue problems exist [LSY98]. Distributed eigensolvers and parallel versions of k-means [ZMH09] can also be used. Care should be taken when the number of communities k is very large. While the trailing eigenvectors of $\mathbf{L}$ can be computed efficiently, the final clustering step would require clustering N k-dimensional vectors, which can prove challenging even for distributed versions of k-means. Multiple approaches that aim to tackle specifically large-scale spectral clustering tasks are available. These approaches come in three major flavors: Parallelization/distributed processing, random sampling and random projections.

A useful overview of performing spectral clustering for sparse Laplacian matrices in parallel is given in [CSB+11]. The parallelization can be performed using either MapReduce [DG08] or MPI [Sni98]. Pre-processing of the data using k-means and random projection trees is investigated in [YHJ09]. In this method the preprocessing step reduces the original N datapoints to $M < N$ representatives and performs spectral clustering on these M representatives, which results in a reduced graph which speeds up execution of the spectral clustering algorithm. However, as usually only similiraties between datapoints are given and not the datapoints themselves, algorithms such as kernel k-means or k-medoids, that can work using only similarities have to be employed.

Random sampling and random projections of the data are advocated in [SI09]. The random projection step involves projecting the data points to a lower dimensional space and computing the similarity matrix from these lower-dimensional representations of the data points. Again, as usually only the similarities are given and not the datapoints themselves, this part of the algorithm cannot provide the needed computational time reduction. Afterwards entries of the similarity matrix are randomly sampled and spectral clustering is performed using this reduced similarity matrix. Nyström's method is proposed in [WLRB09] and [FBCM04], to form a low-rank similarity matrix,

which is enabled by sampling the original similarity matrix and using the similarities between the sampled and non-sampled points. Finding the eigenspace of the new low-rank similarity matrix is much more efficient, however one should perform the sampling carefully, as it can drastically influence the final result. When the similarity matrix is sparse, the Nyström eigenspace can be very similar to the original eigenspace, leading to highly accurate clustering.

Random sampling of the similarity matrix is explored in [ST11], whereby entries of the similarity matrix are sampled randomly, based on a budget constraint, and all other entries are set to zero. This sparsifies the similarity matrix and leads to faster computation of the eigenvectors. Random sketching is promoted in [GKB13] and [LC10]. Entries of the similarity matrix are randomly sketched using a random projection matrix to reduce the size of the similarity matrix. This reduction allows for faster computation of the eigenspace.

Large-scale kernel k-means methods can also significantly speed-up the clustering process. Results in [DGK07] have shown that using kernel k-means instead of spectral clustering can reduce the computation time required. Again care should be taken when using kernel k-means. While the Laplacian matrix of a social network might be sparse, the similarity matrix in general is not, possibly increasing the clustering time of kernel k-means (compared to spectral clustering methods), especially in cases where the number of clusters k is small. Methods to scale the kernel k-means algorithm are also available. Random sampling of the kernel matrix is investigated in [CJHJ11], where the centroids (cluster representatives) are forced to reside on the subspace spanned by those sampled points. Simulated tests demonstrate that the resultant algorithm can tackle large datasets effectively. Parallelization of the kernel k-means algorithm is proposed in [EFKK14]. Here, low-dimensional embeddings allow kernel k-means to be used in a distributed fashion using the MapReduce framework.

A more recent method, called Sketching and Validation (SkeVa [TSG15]) proposes taking multiple sketches (random samples of $M < N$ entries) of the similarity matrix, performing kernel k-means to find clusterings, and relies on different sets of random samples to validate these sketches by assigning a score to them. Afterwards, the sketch that yielded the highest score is used to cluster the remaining data. This structured trial-and-error approach has shown promising results with respect to clustering accuracy and reduced computational time. Furthermore, as each of the sketching and validation runs is independent, this approach admits easy parallelization, thereby combining random sketching approaches and distributed computing, making it attractive for the task at hand. Simulated and real data tests indicate that this algorithm can tackle large-scale datasets much faster than traditional kernel k-means.

10.4.1.1 Numerical tests

Three methods are compared in this section with respect to clustering ac-

TABLE 10.4.1: Clustering times for Facebook egonet.

	Spectral Clustering	Kernel k-means	SkeVa (150 samples)	SkeVa (350 samples)
Time(s)	0.067	0.074	0.031	0.066

TABLE 10.4.2: Clustering times for arXiv General relativity collaboration network.

	Spectral Clustering	Kernel k-means	SkeVa (500 samples)	SkeVa (1,000 samples)
Time(s)	3.1	2.51	0.4	0.85

curacy and required time: Spectral clustering, kernel k-means and the Sketching and Validation method for kernel k-means (Kernel SkeVa) introduced in [TSG15]. Regarding spectral clustering, the normalized version of $\mathbf{L}$ is used [NJW+02], and Lanczos iterations are employed to evaluate the eigenspace of $\mathbf{L}$. The kernel used for kernel k-means and kernel SkeVa is the shortest path distance kernel $\mathbf{K} = (-1/2)\mathbf{JDJ}$, where $\mathbf{D}$ contains the shortest path distances between every node pair in the graph and $\mathbf{J} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T$ is the double centering operator, while $\mathbf{1}$ denotes the all-ones vector. Figure 10.4.1 shows the community detection result for the different algorithms on the largest connected component of a Facebook egonet with $N = 744$ vertices, and $30,023$ edges containing $k = 5$ communities [Les12]. Vertices of the graph represent friends of a particular user, and edges between the vertices indicate whether two people are friends with each other. All methods are able to distinguish the clearly defined communities in this graph. Kernel SkeVa misclassifies some nodes when only 150 nodes are sampled, but this is to be expected as not all nodes are sampled. Since the size of this network is small all methods require similar amounts of time to perform the clustering (see Table 10.4.1). Figure 10.4.2 shows the community detection result on the largest connected component of an arXiv collaboration network (General Relativity) with $N = 4,158$ vertices, and $13,422$ edges [Les11]. Vertices represent paper authors and edges indicate whether two people have co-authored a paper. It is assumed that $k = 36$ communities are present in this graph. Similar to the Facebook network, all algorithms are able to recognize the tight communities of this network, however the time required for kernel SkeVa is one order of magnitude lower.

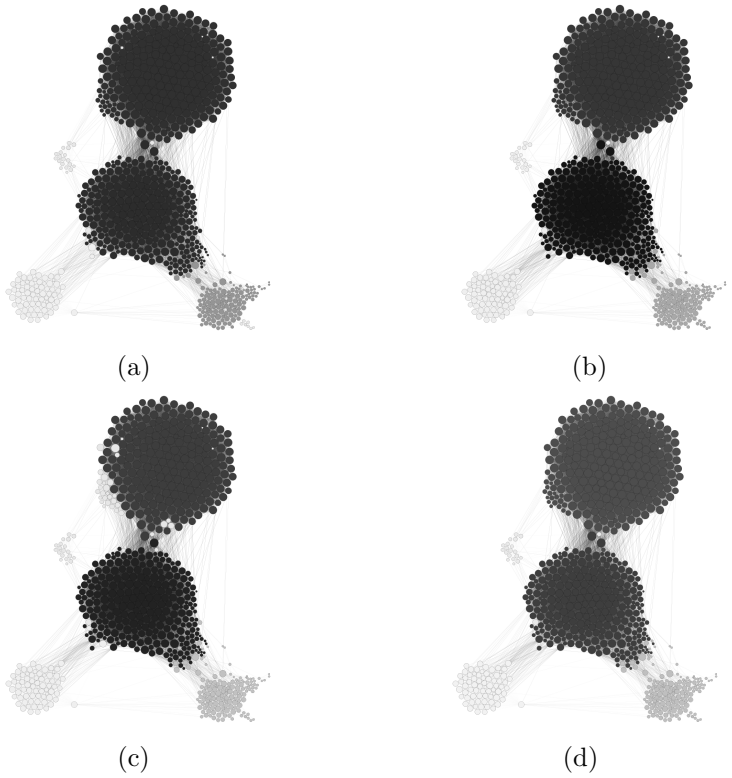


FIGURE 10.4.1: Community detection results for a Facebook egonet with $N = 744$ nodes and $k = 5$ communities using: (a) Normalized spectral clustering; (b) Kernel k-means; (c) Kernel SkeVa, where 150 nodes are sampled; and (d) Kernel SkeVa, where 350 nodes are sampled. Different shades of gray represent different communities.

10.4.2 Robust kernel PCA

Kernel (K)PCA is a generalization of (linear) PCA, seeking principal components in a *feature space* nonlinearly related to the *input space*, where the data in $\mathcal{T}_x$ live [SSM98]. KPCA has been shown effective in performing nonlinear feature extraction for pattern recognition [SSM98]. In addition, connections between KPCA and spectral clustering [HTF09, p. 548] motivate well the KPCA method outlined in this section, to robustly identify cohesive subgroups (communities) from social network data.

Consider a nonlinear function $\boldsymbol{\phi} : \mathbb{R}^D \to \mathcal{H}$, that maps elements from the input space $\mathbb{R}^D$ to a feature space $\mathcal{H}$ of arbitrarily large – possibly infinite – dimensionality. Given transformed training data $\mathcal{T}_{\mathcal{H}} := \{\boldsymbol{\phi}(\mathbf{y}_n)\}_{n=1}^N$, the proposed approach to robust KPCA fits the model [cf. the low-rank subspace

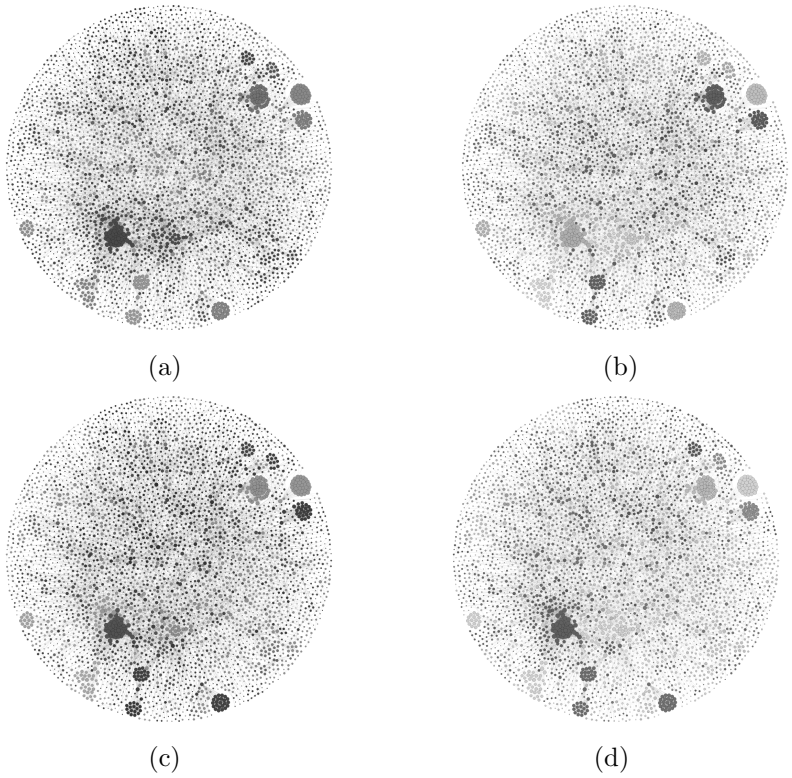


FIGURE 10.4.2: Community detection results for an arXiv collaboration network (General Relativity) with $N = 4,158$ nodes, and $k = 36$ communities using: (a) Normalized spectral clustering; (b) Kernel k-means; (c) Kernel SkeVa where 500 nodes are sampled; and (d) Kernel SkeVa where $1,000$ nodes are sampled. Different shades of gray represent different communities.

model in (10.3.3)]:

$$\boldsymbol{\phi}(\mathbf{y}_n) = \mathbf{m} + \mathbf{P}\mathbf{q}_n + \mathbf{o}_n + \mathbf{e}_n, \quad n = 1, \ldots, N, \tag{10.4.6}$$

where, again, $\mathbf{o}_n$ is an outlier vector, and $\mathbf{m}$ denotes the location (mean) vector in feature space $\mathcal{H}$. A natural criterion is ($\boldsymbol{\Phi} := [\boldsymbol{\phi}(\mathbf{y}_1), \ldots, \boldsymbol{\phi}(\mathbf{y}_N)]$ and $\mathbf{1}_N^T$ is the $N \times 1$ row vector of all ones):

$$\min_{\mathbf{m},\mathbf{P},\mathbf{Q},\mathbf{O}} \|\boldsymbol{\Phi} - \mathbf{m}\mathbf{1}_N^T - \mathbf{P}\mathbf{Q}^T - \mathbf{O}\|_F^2 + \frac{\lambda_*}{2}(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2) + \lambda_2 \sum_{n=1}^{N} \|\mathbf{o}_n\|_2, \tag{10.4.7}$$

where $\sum_{n=1}^{N} \|\mathbf{o}_n\|_2$ is the so-termed group Lasso penalty [YL06]. It is a high-dimensional extension of the ℓ_1-norm, that encourages columnwise (vector) sparsity on the estimator of $\mathbf{O}$ [cf. entrywise sparsity with the ℓ_1-norm]. This

way, one can declare whether the corresponding training vector $\boldsymbol{\phi}(\mathbf{y}_1)$ is an outlier or not. Except for the principal components' matrix $\mathbf{Q} \in \mathbb{R}^{N\times\rho}$, both the data and the unknowns in (10.4.7) are now vectors/matrices of generally infinite dimension. In principle, this challenges the optimization task since it is impossible to store, or, perform updates of such quantities directly. For these reasons, assuming zero-mean data $\boldsymbol{\phi}(\mathbf{y}_n)$, or, the possibility of mean compensation for that matter, cannot be taken for granted here. Thus, it is important to explicitly consider the estimation of $\mathbf{m}$ [which for instance, was not explicitly accounted for in (10.3.3)].

Interestingly, this hurdle can be overcome by endowing $\mathcal{H}$ with the structure of a reproducing kernel Hilbert space (RKHS), where inner products between any two members of $\mathcal{H}$ boil down to evaluations of the reproducing kernel $K_{\mathcal{H}} : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$, i.e., $\langle \boldsymbol{\phi}(\mathbf{y}_i), \boldsymbol{\phi}(\mathbf{y}_j)\rangle_{\mathcal{H}} = K_{\mathcal{H}}(\mathbf{y}_i, \mathbf{y}_j)$. Specifically, it is possible to form the kernel matrix $\mathbf{K} := \boldsymbol{\Phi}^T\boldsymbol{\Phi} \in \mathbb{R}^{N\times N}$, without directly working with the vectors in $\mathcal{H}$. This so-termed *kernel trick* is the crux of most kernel methods in machine learning [HTF09], including kernel PCA [SSM98]. The problem of selecting $K_{\mathcal{H}}$ (and $\boldsymbol{\phi}$ indirectly) will not be considered here.

Building on these ideas, it is shown in [MG12b] that natural alternating-minimization (AM) iterations one can devise to optimize (10.4.7) can be *kernelized*, to solve (10.4.7) at affordable computational complexity and memory storage requirements that do not depend on the dimensionality of $\mathcal{H}$. Specifically, for $k \geq 1$, [MG12b] shows that the sequence of AM iterates obtained to solve (10.4.7) can be written as $\mathbf{m}(k) = \boldsymbol{\Phi}\boldsymbol{\mu}(k)$, $\mathbf{P}(k) = \boldsymbol{\Phi}\boldsymbol{\Pi}(k)$, and $\mathbf{O}(k) = \boldsymbol{\Phi}\boldsymbol{\Omega}(k)$. The quantities $\boldsymbol{\mu}(k) \in \mathbb{R}^N$, $\boldsymbol{\Pi}(k) \in \mathbb{R}^{N\times\rho}$, and $\boldsymbol{\Omega}(k) \in \mathbb{R}^{N\times N}$ are then recursively updated as in Algorithm 10.4.3, without the need of operating with vectors in $\mathcal{H}$.

In order to run the robust KPCA algorithm (tabulated as Algorithm 10.4.3), one does not have to store or process the quantities $\mathbf{m}(k)$, $\mathbf{U}(k)$, and $\mathbf{O}(k)$. As per [MG12b, Prop. 4], the iterations can be equivalently carried out by cycling through *finite-dimensional* 'sufficient statistics' $\boldsymbol{\mu}(k) \to \boldsymbol{\Pi}(k) \to \mathbf{Q}(k) \to \boldsymbol{\Omega}(k)$. In other words, the iterations of the robust kernel PCA algorithm are devoid of algebraic operations among vectors in $\mathcal{H}$. Recall that the size of matrix $\mathbf{Q}$ is independent of the dimensionality of $\mathcal{H}$.

Because $\mathbf{O}(k) = \boldsymbol{\Phi}\boldsymbol{\Omega}(k)$ and upon convergence of the algorithm, the outlier vector norms are computable in terms of $\mathbf{K}$, i.e., $[\|\mathbf{o}_1(\infty)\|_2^2, \ldots, \|\mathbf{o}_N(\infty)\|_2^2]^T = \text{diag}[\boldsymbol{\Omega}^T(\infty)\mathbf{K}\boldsymbol{\Omega}(\infty)]$. These are critical towards identifying outlying vectors $\mathbf{y}_n$, since for those $\|\mathbf{o}_n(\infty)\|_2 > 0$. Moreover, the principal component corresponding to any given new data point $\mathbf{y}$ is obtained through the projection $\mathbf{1} = \mathbf{P}^T(\infty)[\boldsymbol{\phi}(\mathbf{y}) - \mathbf{m}(\infty)] = \boldsymbol{\Pi}^T(\infty)\boldsymbol{\Phi}^T\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\Pi}^T(\infty)\mathbf{K}\boldsymbol{\mu}(\infty)$, which is again computable after N evaluations of the kernel function $K_{\mathcal{H}}$.

10.4.2.1 Numerical tests

Here robust KPCA is used to identify communities and outliers in a social network of $N = 115$ college football teams, by capitalizing on the connection

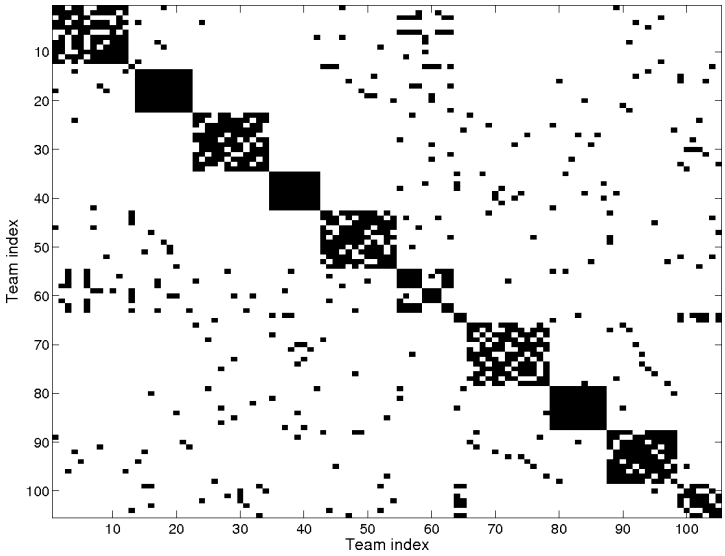
Algorithm 10.4.3: : Robust KPCA solver

Initialize $\mathbf{\Omega}(0) = \mathbf{0}_{N\times N}$, $\mathbf{Q}(0)$ randomly, and form $\mathbf{K} = \mathbf{\Phi}^T\mathbf{\Phi}$.
for $k = 1, 2, \ldots$ **do**
  Update $\boldsymbol{\mu}(k) = [\mathbf{I}_N - \mathbf{\Omega}(k-1)]\mathbf{1}_N/N$.
  Form $\mathbf{\Phi}_o(k) = \mathbf{I}_N - \boldsymbol{\mu}(k)\mathbf{1}_N^T - \mathbf{\Omega}(k-1)$.
  Update $\mathbf{\Pi}(k) = \mathbf{\Phi}_o(k)\mathbf{Q}(k-1)[\mathbf{Q}^T(k-1)\mathbf{Q}(k-1) + (\lambda_*/2)\mathbf{I}_\rho]^{-1}$.
  Update $\mathbf{Q}(k) = \mathbf{\Phi}_o^T(k)\mathbf{K}\mathbf{\Pi}(k)[\mathbf{\Pi}^T(k)\mathbf{K}\mathbf{\Pi}(k) + (\lambda_*/2)\mathbf{I}_\rho]^{-1}$.
  Form $\boldsymbol{\delta}_n(k) = \mathbf{e}_{N,n} - \boldsymbol{\mu}(k) - \mathbf{\Pi}(\boldsymbol{k})\mathbf{q}_n(k)$, $n = 1, \ldots, N$
  Form $\mathbf{\Delta}(k) = \text{diag}\left(\frac{(\boldsymbol{\delta}_1^T(k)\mathbf{K}\boldsymbol{\delta}_1(k) - \frac{\lambda_2}{2})_+}{\boldsymbol{\delta}_1^T(k)\mathbf{K}\boldsymbol{\delta}_1(k)}, \ldots, \frac{(\boldsymbol{\delta}_N^T(k)\mathbf{K}\boldsymbol{\delta}_N(k) - \frac{\lambda_2}{2})_+}{\boldsymbol{\delta}_N^T(k)\mathbf{K}\boldsymbol{\delta}_N(k)}\right)$.
  Update $\mathbf{\Omega}(k) = [\mathbf{I}_N - \boldsymbol{\mu}(k)\mathbf{1}_N^T - \mathbf{\Pi}(\boldsymbol{k})\mathbf{Q}^T(k)]\mathbf{\Delta}(k)$.
end for

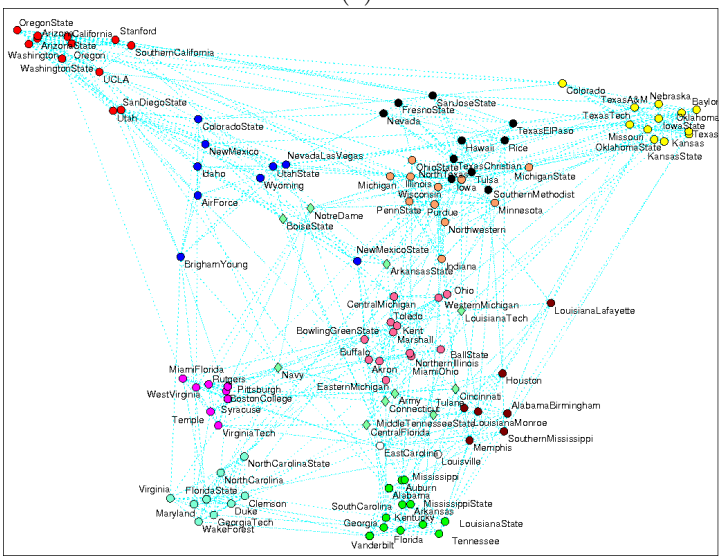
between KPCA and spectral clustering [HTF09, p. 548]. Nodes in the network graph represent teams belonging to eleven conferences (plus five independent teams), whereas (unweighted) edges joining pairs of nodes indicate that both teams played against each other during the Fall 2000 Division I season [GN02]. The kernel matrix used to run robust KPCA is $\mathbf{K} = \zeta\mathbf{I}_N + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{A}$ and $\mathbf{D}$ denote the graph adjacency and degree matrices, respectively; while $\zeta > 0$ is chosen to render $\mathbf{K}$ positive semi-definite. The choice of the normalized graph Laplacian as kernel matrix is at the heart of the equivalence between KPCA and spectral clustering [HTF09]. The tuning parameters are chosen as $\lambda_2 = 1.297$ so that $\|\hat{\mathbf{O}}\|_0 = 10$, while $\lambda_* = 1$, and $\rho = 3$. Figure 10.4.3 (a) shows the entries of $\mathbf{K}$, where rows and columns are permuted to reveal the clustering structure found by robust KPCA (after removing the outliers); see also Figure 10.4.3 (b). The quality of the clustering is assessed through the adjusted rand index (ARI) after excluding outliers [FKG11], which yielded the value 0.8967. Four of the teams deemed as outliers are Connecticut, Central Florida, Navy, and Notre Dame, which are indeed independent teams not belonging to any major conference. The community structure of traditional powerhouse conferences such as Big Ten, Big 12, ACC, Big East, and SEC was identified exactly.

10.5 Topology tracking from information cascades

It has been observed in many settings that information often spreads in *cascades* by following implicit links between nodes in a network. Examples include the propagation of viral news events between blogs, adoption of emerging fashion trends within an age group, or acquisition of new buying habits by consumer groups. Consider the example of a terrorist attack reported within minutes on mainstream news websites. An information cascade may emerge



(a)



(b)

FIGURE 10.4.3: (a) Entries of $\mathbf{K}$ after removing the outliers, where rows and columns are permuted to reveal the clustering structure found by robust KPCA; and (b) Graph depiction of the clustered network [MG12b]. Teams belonging to the same estimated conference (cluster) are colored identically. The outliers are represented as diamond-shaped nodes.

because these websites' readership includes bloggers who subsequently write about the attack, influencing their own readers in turn to do the same. The underlying dynamics for propagation of such information are remarkably similar to those governing the rapid spread of infectious diseases within a population, leading to the so-termed *contagions* [Rog95, EK10, BMG14]. In general, one is only able to observe the nodes of such networks and the times when they got "infected" by a contagion, but not their link topology.

Knowledge of the network topology is crucial for several reasons. Viral web advertising can be more effectively achieved if a small set of influential initiators are identified through the link structure. Furthermore, knowledge of the structure of hidden needle-sharing networks among communities of injecting drug users can aid formulation of policies for curbing contagious diseases. Other examples include assessment of the reliability of heavily inter-connected systems such as the power grid, or, estimating risk exposure among investment banks in a highly interdependent global economy. In general, unveiling the network topology can be used to predict the behavior of complex systems [Kol09, RLS10, BMG14].

Key to topology identification from a cascade is the ease of observation of its evolution over the unknown network. Indeed, this is tantamount to simply recording the times when each node got infected by a cascade. Well-studied diffusion models based on epidemiological studies have been put forth to identify an underlying network topology [VR07, EK10, Jac10]. Undoubtedly a challenging inference task, analysis of information cascades over modern social networks leads to the fundamental big data challenges. Cascades typically propagate over very large web-scale networks, and are acquired sequentially in infinite streams. More importantly, the underlying network topology is generally dynamic and varies over time.

Network inference from temporal traces of infection events has recently emerged as an active area of research. According to the taxonomy in [Kol09, Ch. 7], this can be viewed as a problem involving inference of *association networks*. Two other broad classes of network topology identification problems entail (individual) link prediction, or, *tomographic inference*. Several approaches postulate probabilistic models and rely on maximum likelihood estimation (MLE) to infer static edge weights as pairwise transmission rates between nodes [RBS11, ML13]. MLE-based stochastic gradient descent (SGD) iterations have been leveraged for inference of temporal diffusion networks [RLS10]. Most contemporary diffusion models attribute node infection events to the network topology alone (*endogenous* factors), and ignore *exogenous* factors such as non-topological information sources. Modeling causal endogenous and exogenous factors is the mainstay of *structural equation models (SEMs)*, and the rest of this section will focus on such a general approach that was recently advocated in [BMG14].

10.5.1 Dynamic SEMs for tracking cascades

Structural equation modeling refers to a family of statistical methods that model causal relationships between interacting variables in a complex system; see e.g., [Kap09]. Their appeal can be attributed to simplicity and the inherent ability to capture edge directionalities in graphs. They have been adopted in economics, psychometrics [Mut84], social sciences [Gol72], and genetics [LdH08, CBG13], among others.

Reasoning that infection times depend on both topological (endogenous) and external (exogenous) influences, a novel SEM-based scheme was proposed in [BMG14] for cascade modeling. Topological influences are modeled as linear combinations of infection times of other nodes in the network, whose weights correspond to entries in a time-varying asymmetric adjacency matrix. External influences such as those due to on-site reporting in news propagation contexts are useful for model identifiability, as they have been shown necessary to resolve directional ambiguities [BBG13]. It is assumed that the network varies slowly with time, facilitating adaptive parameter estimation by minimizing a sparsity-promoting exponentially-weighted LS criterion. Furthermore, inherent sparse connectivity of social networks is accounted for by ℓ_1-norm regularization [CGH09, ABG10, KST11, AG11].

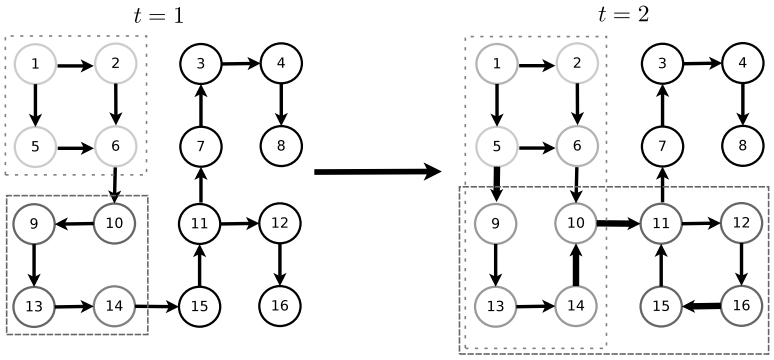


FIGURE 10.5.1: Two cascades propagating over a dynamic directed 16-node network during time intervals $t = 1$ and $t = 2$. Both cascades are initially observable at 4 nodes and they propagate to include 4 extra nodes during $t = 2$. Changes to the network topology are depicted by the new, thicker edges during $t = 2$.

10.5.1.1 Model and problem statement

Consider a dynamic N-node network observed over time intervals $t = 1, \ldots, T$, represented by a graph whose temporal topology is encoded through a time-series of unknown, time-indexed, and weighted adjacency matrices $\{\mathbf{A}^t \in \mathbb{R}^{N \times N}\}_{t=1}^{T}$. Per convention in network studies, entry (i, j) of $\mathbf{A}^t$ (hence-

forth denoted by a_{ij}^t) is nonzero only if a directed edge connects nodes i and j during the time interval t. The network topology is assumed to remain fixed per time interval t, but can change across intervals.

Over the course of the observation interval, many contagions propagate over the time-varying network as illustrated in the 16-node directed network in Figure 10.5.1. Suppose a fixed number of contagions C is sampled, and the difference between infection time of node i by contagion c and the earliest observation time is denoted by $y_{ic}^t \geq 0$. For uninfected nodes at interval t, y_{ic}^t is infinite and will be set to a large positive value for practical considerations. Assume that the susceptibility x_{ic} of node i to external (non-topological) infection by contagion c is known and time invariant over the observation interval. In the web context, x_{ic} can be set to the search engine rank of website i w.r.t. keywords associated with c.

The model in [BMG14] postulates that y_{ic}^t is linearly related to x_{ic} and the infection times of its single-hop neighbors. Events adhering to this model of network-facilitated propagation abound on the web where mention of e.g., a major baseball event by a blog will not only depend on the times when similar blogs first reported the event, but also the level of interest of the blogger in baseball as a sport. Similarly, in epidemiological studies an individual's infection time by an infectious disease depends on both the infection times of her immediate contacts as well as her immunity level to the disease. As a result y_{ic}^t is modeled according to the following linear *dynamic* structural equation model (SEM):

$$y_{ic}^t = \sum_{j \neq i} a_{ij}^t y_{jc}^t + b_{ii}^t x_{ic} + e_{ic}^t, \tag{10.5.1}$$

where b_{ii}^t captures the time-varying level of influence of external sources, and e_{ic}^t accounts for measurement errors and unmodeled dynamics. It follows from (10.5.1) that if $a_{ij}^t \neq 0$, then y_{ic}^t is affected by the value of y_{jc}^t. Rewriting (10.5.1) for the entire network leads to the vector model:

$$\mathbf{y}_c^t = \mathbf{A}^t \mathbf{y}_c^t + \mathbf{B}^t \mathbf{x}_c + \mathbf{e}_c^t, \tag{10.5.2}$$

where the $N \times 1$ vector $\mathbf{y}_c^t := [y_{1c}^t, \ldots, y_{Nc}^t]^T$ collects the node infection times by contagion c during interval t, $\mathbf{B}^t := \text{diag}(b_{11}^t, \ldots, b_{NN}^t)$; and likewise $\mathbf{x}_c := [x_{1c}, \ldots, x_{Nc}]^T$ and $\mathbf{e}_c^t := [e_{1c}^t, \ldots, e_{Nc}^t]^T$. Collecting observations for all C contagions yields the dynamic matrix SEM:

$$\mathbf{Y}^t = \mathbf{A}^t \mathbf{Y}^t + \mathbf{B}^t \mathbf{X} + \mathbf{E}^t, \tag{10.5.3}$$

where $\mathbf{Y}^t := [\mathbf{y}_1^t, \ldots, \mathbf{y}_C^t]$, $\mathbf{X} := [\mathbf{x}_1, \ldots, \mathbf{x}_C]$, and $\mathbf{E}^t := [\mathbf{e}_1^t, \ldots, \mathbf{e}_C^t]$ are all $N \times C$ matrices. Given $\{\mathbf{Y}^t\}_{t=1}^T$ and $\mathbf{X}$, the goal is to track the underlying network topology $\{\mathbf{A}^t\}_{t=1}^T$, and the effect of external influences $\{\mathbf{B}^t\}_{t=1}^T$. In order to cope with constraints due to limited measurement budgets, it is desirable that $C \ll N$. Unfortunately without further constraints, this compromises the identifiability of (10.5.3). The approach outlined next overcomes this limitation by leveraging the edge sparsity inherent to social networks.

10.5.1.2 Exponentially-weighted least-squares estimator

For the sake of exposition, consider the *static* setting with all $\{\mathbf{Y}^t\}_{t=1}^T$ available. Leveraging the squared error cost leads to the batch problem:

$$\{\hat{\mathbf{A}}, \hat{\mathbf{B}}\} = \arg\min_{\mathbf{A},\mathbf{B}} \quad \frac{1}{2}\sum_{t=1}^{T} \|\mathbf{Y}^t - \mathbf{A}\mathbf{Y}^t - \mathbf{B}\mathbf{X}\|_F^2 + \lambda\|\mathbf{A}\|_1$$
$$\text{s. t.} \quad a_{ii} = 0,\ b_{ij} = 0,\ \forall i \neq j, \tag{10.5.4}$$

where $\lambda > 0$ controls the sparsity level of $\hat{\mathbf{A}}$. Reasonably assuming the absence of a self-loop at node i leads to the constraint $a_{ii} = 0$, while having $b_{ij} = 0$, $\forall i \neq j$, ensures that $\hat{\mathbf{B}}$ is diagonal as in (10.5.2). Note that the estimator (10.5.4) tacitly assumes equal residual variances since the infection times per cascade result from the same contagion over the entire network.

In big data settings, measurements are more likely to be acquired sequentially over large social networks ($\geq 10^6$ nodes), motivating online estimation algorithms with minimal storage requirements. As a result, preference is given to recursive solvers facilitating sequential topology inference. Incorporating a "forgetting factor" that assigns more weight to more recent residuals then makes it possible to track slow temporal topological variations. Note that the batch estimator (10.5.4) yields the single estimates $\{\hat{\mathbf{A}}, \hat{\mathbf{B}}\}$ that best fit the data $\{\mathbf{Y}^t\}_{t=1}^T$ and $\mathbf{X}$ over the entire measurement horizon $t = 1, \ldots, T$, and as such (10.5.4) neglects potential network variations across time intervals.

For $t = 1, \ldots, T$, the *sparsity-regularized exponentially-weighted LS estimator (EWLSE)* is given by:

$$\{\hat{\mathbf{A}}^t, \hat{\mathbf{B}}^t\} = \arg\min_{\mathbf{A},\mathbf{B}} \quad \frac{1}{2}\sum_{\tau=1}^{t} \beta^{t-\tau}\|\mathbf{Y}^\tau - \mathbf{A}\mathbf{Y}^\tau - \mathbf{B}\mathbf{X}\|_F^2 + \lambda_t\|\mathbf{A}\|_1$$
$$\text{s. t.} \quad a_{ii} = 0,\ b_{ij} = 0,\ \forall i \neq j, \tag{10.5.5}$$

where $\beta \in (0, 1]$ is the forgetting factor that forms estimates $\{\hat{\mathbf{A}}^t, \hat{\mathbf{B}}^t\}$ using all measurements acquired until time t. Whenever $\beta < 1$, past data are exponentially discarded thus enabling tracking of dynamic network topologies. The first summand in the cost corresponds to an exponentially-weighted moving average (EWMA) of the squared model residuals norms. The EWMA can be seen as an average modulated by a sliding window of equivalent length $1/(1-\beta)$, which clearly grows as $\beta \to 1$. In the so-termed infinite-memory setting whereby $\beta = 1$, (10.5.5) boils down to the batch estimator (10.5.4). Notice that λ_t is allowed to vary with time in order to capture the generally changing edge sparsity level. In a linear regression context, a related EWLSE was put forth in [ABG10] for adaptive estimation of sparse signals; see also [KST11] for a projection-based adaptive algorithm.

10.5.2 Topology tracking algorithm

Proximal gradient (PG) algorithms have been popularized for ℓ_1-norm regularized linear regression problems, through the class of *iterative shrinkage-thresholding algorithms (ISTA)*; see e.g., [DDM04] and [PB13] for a comprehensive tutorial treatment. The main advantage of ISTA over off-the-shelf interior point methods is its computational simplicity. Iterations boil down to matrix-vector multiplications involving the regression matrix, followed by a soft-thresholding operation [HTF09, p. 93].

Introducing the optimization variable $\mathbf{V} := [\mathbf{A}\ \mathbf{B}]$, it follows that the gradient of $f(\mathbf{V}) := \frac{1}{2}\sum_{\tau=1}^{t} \beta^{t-\tau}\|\mathbf{Y}^\tau - \mathbf{A}\mathbf{Y}^\tau - \mathbf{B}\mathbf{X}\|_F^2$ is Lipschitz continuous, i.e., $\|\nabla f(\mathbf{V}_1) - \nabla f(\mathbf{V}_2)\| \leq L_f\|\mathbf{V}_1 - \mathbf{V}_2\|$, $\forall\, \mathbf{V}_1$, $\mathbf{V}_2$ in the domain of f. The Lipschitz constant L_f is time varying, but its dependence on t is kept implicit for notational convenience. Instead of directly optimizing the cost in (10.5.5), PG algorithms minimize a sequence of overestimators evaluated at judiciously chosen points.

Let $k = 1, 2, \ldots$ denote iterations and define $g(\mathbf{V}) := \lambda_t\|\mathbf{A}\|_1$, PG algorithms iteratively solve:

$$\mathbf{V}[k] := \arg\min_{\mathbf{V}} \left\{ \frac{L_f}{2}\|\mathbf{V} - \mathbf{G}(\mathbf{V}[k-1])\|_F^2 + g(\mathbf{V}) \right\}, \tag{10.5.6}$$

where $\mathbf{G}(\mathbf{V}[k-1]) := \mathbf{V}[k-1] - (1/L_f)\nabla f(\mathbf{V}[k-1])$ corresponds to a gradient-descent step taken from $\mathbf{V}[k-1]$, with step-size equal to $1/L_f$. The optimization problem (10.5.6) is known as the *proximal operator* of the function g/L_f evaluated at $\mathbf{G}(\mathbf{V}[k-1])$, and is denoted as $\text{prox}_{g/L_f}(\mathbf{G}(\mathbf{V}[k-1]))$. Henceforth adopting the notation $\mathbf{G}[k-1] := \mathbf{G}(\mathbf{V}[k-1])$ for convenience, the PG iterations can be compactly rewritten as $\mathbf{V}[k] = \text{prox}_{g/L_f}(\mathbf{G}[k-1])$.

A key element to the success of PG algorithms stems from the possibility of efficiently evaluating the proximal operator (cf. (10.5.6)). Specializing to (10.5.5), note that (10.5.6) decomposes into:

$$\begin{aligned} \mathbf{A}[k] &:= \arg\min_{\mathbf{A}} \left\{ \frac{L_f}{2}\|\mathbf{A} - \mathbf{G}_A[k-1]\|_F^2 + \lambda_t\|\mathbf{A}\|_1 \right\} \\ &= \mathcal{S}_{\lambda_t/L_f}(\mathbf{G}_A[k-1]) \end{aligned} \tag{10.5.7}$$

$$\mathbf{B}[k] := \arg\min_{\mathbf{B}} \left\{ \|\mathbf{B} - \mathbf{G}_B[k-1]\|_F^2 \right\} = \mathbf{G}_B[k-1], \tag{10.5.8}$$

subject to the constraints in (10.5.5), which so far have been left implicit, and $\mathbf{G} := [\mathbf{G}_A\ \mathbf{G}_B]$. Letting $\mathcal{S}_\mu(\mathbf{M})$ with (i,j)-th entry given by $\text{sign}(m_{ij})\max(|m_{ij}| - \mu, 0)$ denote the soft-thresholding operator, it follows that $\text{prox}_{\lambda_t\|\cdot\|_1/L_f}(\cdot) = \mathcal{S}_{\lambda_t/L_f}(\cdot)$, e.g., [DDM04, HTF09]. Because there is no regularization on the matrix $\mathbf{B}$, the corresponding update (10.5.8) boils-down to a simple gradient-descent step.

What remains now is to obtain expressions for the gradient of $f(\mathbf{V})$ with respect to $\mathbf{A}$ and $\mathbf{B}$, which are required to form the matrices $\mathbf{G}_A$ and $\mathbf{G}_B$. To

this end, note that by incorporating the constraints $a_{ii} = 0$ and $b_{ij} = 0$, $\forall j \neq i$, $i = 1, \ldots N$, one can simplify the expression of $f(\mathbf{V})$ as:

$$f(\mathbf{V}) := \frac{1}{2} \sum_{\tau=1}^{t} \sum_{i=1}^{N} \beta^{t-\tau} \|(\mathbf{y}_i^\tau)^T - \mathbf{a}_{-i}^T \mathbf{Y}_{-i}^\tau - b_{ii} \mathbf{x}_i^T\|_F^2, \tag{10.5.9}$$

where $(\mathbf{y}_i^\tau)^T$ and $\mathbf{x}_i^T$ denote the i-th row of $\mathbf{Y}^\tau$ and $\mathbf{X}$, respectively; while $\mathbf{a}_{-i}^T$ denotes the $1 \times (N-1)$ vector obtained by removing entry i from the i-th row of $\mathbf{A}$, and likewise $\mathbf{Y}_{-i}^\tau$ is the $(N-1) \times C$ matrix obtained by removing row i from $\mathbf{Y}^\tau$. It is apparent from (10.5.9) that $f(\mathbf{V})$ is separable across the trimmed row vectors $\mathbf{a}_{-i}^\top$, and the diagonal entries b_{ii}, $i = 1, \ldots, N$. The sought gradients are:

$$\nabla_{\mathbf{a}_{-i}} f(\mathbf{V}) = \mathbf{\Sigma}_{-i}^t \mathbf{a}_{-i} + \bar{\mathbf{Y}}_{-i}^t \mathbf{x}_i b_{ii} - \boldsymbol{\sigma}_{-i}^t \tag{10.5.10}$$

$$\nabla_{b_{ii}} f(\mathbf{V}) = \mathbf{a}_{-i}^T \bar{\mathbf{Y}}_{-i}^t \mathbf{x}_i + \frac{1-\beta^t}{1-\beta} b_{ii} \|\mathbf{x}_i\|_2^2 - (\bar{\mathbf{y}}_i^\tau)^T \mathbf{x}_i, \tag{10.5.11}$$

where $(\bar{\mathbf{y}}_i^t)^T$ denotes the i-th row of $\bar{\mathbf{Y}}^t := \sum_{\tau=1}^t \beta^{t-\tau} \mathbf{Y}^\tau$, and $\bar{\mathbf{Y}}_{-i}^t := \sum_{\tau=1}^t \beta^{t-\tau} \mathbf{Y}_{-i}^\tau$. Similarly, $\boldsymbol{\sigma}_{-i}^t := \sum_{\tau=1}^t \beta^{t-\tau} \mathbf{Y}_{-i}^\tau \mathbf{y}_i^\tau$ and $\mathbf{\Sigma}_{-i}^t$ is obtained by removing the i-th row and i-th column from $\mathbf{\Sigma}^t := \sum_{\tau=1}^t \beta^{t-\tau} \mathbf{Y}^\tau (\mathbf{Y}^\tau)^T$. From (10.5.7)-(10.5.8) and (10.5.10)-(10.5.11), the parallel ISTA iterations:

$$\nabla_{\mathbf{a}_{-i}} f[k] = \mathbf{\Sigma}_{-i}^t \mathbf{a}_{-i}[k] + \bar{\mathbf{Y}}_{-i}^t \mathbf{x}_i b_{ii}[k] - \boldsymbol{\sigma}_{-i}^t \tag{10.5.12}$$

$$\nabla_{b_{ii}} f[k] = \mathbf{a}_{-i}^T[k] \bar{\mathbf{Y}}_{-i}^t \mathbf{x}_i + \frac{(1-\beta^t)}{1-\beta} b_{ii}[k] \|\mathbf{x}_i\|_2^2 - (\bar{\mathbf{y}}_i^t)^T \mathbf{x}_i \tag{10.5.13}$$

$$\mathbf{a}_{-i}[k+1] = \mathcal{S}_{\lambda_t/L_f} \left(\mathbf{a}_{-i}[k] - (1/L_f) \nabla_{\mathbf{a}_{-i}} f[k] \right) \tag{10.5.14}$$

$$b_{ii}[k+1] = b_{ii}[k] - (1/L_f) \nabla_{b_{ii}} f[k] \tag{10.5.15}$$

are provably convergent to the globally optimal solution $\{\hat{\mathbf{A}}^t, \hat{\mathbf{B}}^t\}$ of (10.5.5), as per the general convergence results available for PG methods and ISTA in particular [DDM04, PB13].

Computation of the gradients in (10.5.12)-(10.5.13) requires one matrix-vector mutiplication by $\mathbf{\Sigma}_{-i}^t$ and one by $\bar{\mathbf{Y}}_{-i}^t$, in addition to three vector inner-products, plus a few (negligibly complex) scalar and vector additions. Both the update of $b_{ii}[k+1]$ as well as the soft-thresholding operation in (10.5.14) entail negligible computational complexity. Per iteration, the actual rows of the adjacency matrix are obtained by zero-padding the updated $\mathbf{a}_{-i}[k]$, namely setting:

$$\mathbf{a}_i^T[k] = [a_{-i,1}[k] \ldots a_{-i,i-1}[k] \; 0 \; a_{-i,i}[k] \ldots a_{-i,N}[k]]. \tag{10.5.16}$$

This way, the desired SEM parameter estimates at time t are given by $\hat{\mathbf{A}}^t = [\mathbf{a}_1^\top[k], \ldots, \mathbf{a}_N^\top[k]]^T$ and $\hat{\mathbf{B}}^t = \text{diag}(b_{11}[k], \ldots, b_{NN}[k])$, for k large enough so that convergence has been attained.

Algorithm 10.5.1: Pseudo real-time ISTA for topology tracking

Require: $\{\mathbf{Y}^t\}_{t=1}^T$, $\mathbf{X}$, β.
Initialize $\hat{\mathbf{A}}^0 = \mathbf{0}_{N\times N}, \hat{\mathbf{B}}^0 = \mathbf{\Sigma}^0 = \mathbf{I}_N, \bar{\mathbf{Y}}^0 = \mathbf{0}_{N\times C}, \lambda_0$.
for $t = 1, \ldots, T$ **do**
  Update λ_t, L_f and $\mathbf{\Sigma}^t$, $\bar{\mathbf{Y}}^t$ via (10.5.17).
  Initialize $\mathbf{A}[0] = \hat{\mathbf{A}}^{t-1}$, $\mathbf{B}[0] = \hat{\mathbf{B}}^{t-1}$, and set $k = 0$.
  while not converged **do**
    for $i = 1 \ldots N$ (in parallel) **do**
      Compute $\mathbf{\Sigma}^t_{-i}$ and $\bar{\mathbf{Y}}^t_{-i}$.
      Form gradients at $\mathbf{a}_{-i}[k]$ and $b_{ii}[k]$ via (10.5.12)-(10.5.13).
      Update $\mathbf{a}_{-i}[k+1]$ via (10.5.14).
      Update $b_{ii}[k+1]$ via (10.5.15).
      Update $\mathbf{a}_i[k+1]$ via (10.5.16).
    end for
    $k = k + 1$.
  end while
  return $\hat{\mathbf{A}}^t = \mathbf{A}[k]$, $\hat{\mathbf{B}}^t = \mathbf{B}[k]$.
end for

Solving (10.5.5) **over the entire time horizon** $t = 1, \ldots, T$**.** To track the dynamically-evolving network topology, one can go ahead and solve (10.5.5) sequentially for each $t = 1, \ldots, T$ as data arrive, using (10.5.12)-(10.5.15). Because the network is assumed to vary slowly across time, it is convenient to warm-restart the ISTA iterations, that is, at time t initialize $\{\mathbf{A}[0], \mathbf{B}[0]\}$ with the solution $\{\hat{\mathbf{A}}^{t-1}, \hat{\mathbf{B}}^{t-1}\}$. This way, for smooth network variations one expects convergence to be attained after few iterations.

To obtain the new SEM parameter estimates via (10.5.12)-(10.5.15), it suffices to update (possibly) λ_t and the Lipschitz constant L_f, as well as the data-dependent EWMAs $\mathbf{\Sigma}^t$, and $\bar{\mathbf{Y}}^t$. Interestingly, the potential growing-memory problem in storing the entire history of data $\{\mathbf{Y}^t\}_{t=1}^T$ can be avoided by performing the recursive updates:

$$\mathbf{\Sigma}^t = \beta\mathbf{\Sigma}^{t-1} + \mathbf{Y}^t(\mathbf{Y}^t)^T, \quad \bar{\mathbf{Y}}^t = \beta\bar{\mathbf{Y}}^{t-1} + \mathbf{Y}^t. \tag{10.5.17}$$

The complexity in evaluating the Gram matrix $\mathbf{Y}^t(\mathbf{Y}^t)^T$ dominates the per-iteration computational cost of the algorithm. To circumvent the need of re-computing the Lipschitz constant per time interval, the step-size $1/L_f$ in (10.5.14)-(10.5.15) can be selected by a line search [PB13]. One choice is the backtracking step-size rule [BT09], for which convergence to $\{\hat{\mathbf{A}}^t, \hat{\mathbf{B}}^t\}$ can be established as well.

Algorithm 10.5.1 summarizes the steps outlined in this section for tracking the dynamic network topology, given temporal traces of infection events $\{\mathbf{Y}^t\}_{t=1}^T$ and susceptibilities $\mathbf{X}$. It is termed *pseudo real-time* ISTA, since in principle one needs to run multiple (inner) ISTA iterations till convergence

per time interval $t = 1, \ldots, T$. This will in turn incur an associated delay, that may (or may not) be tolerable depending on the specific network inference problem at hand. Nevertheless, numerical tests indicate that in practice 5-10 inner iterations suffice for convergence; see [BMG14] for further details.

10.5.2.1 Accelerated convergence

For big data applications, first-order methods such as ISTA are often the only admissible option. Recently, several efforts have led to improvement of the sublinear global rate of convergence exhibited by PG algorithms while retaining their computational simplicity see e.g., [Nes83, Nes05, BT09] and references therein.

The so-termed *accelerated* (A)PG algorithm has been shown to remarkably attain convergence speedups in [Nes05]. APG algorithms generate the following sequence of iterates:

$$\mathbf{V}[k] = \arg\min_{\mathbf{V}} Q(\mathbf{V}, \mathbf{U}[k-1]) = \text{prox}_{g/L_f}(\mathbf{G}(\mathbf{U}[k-1])),$$

where

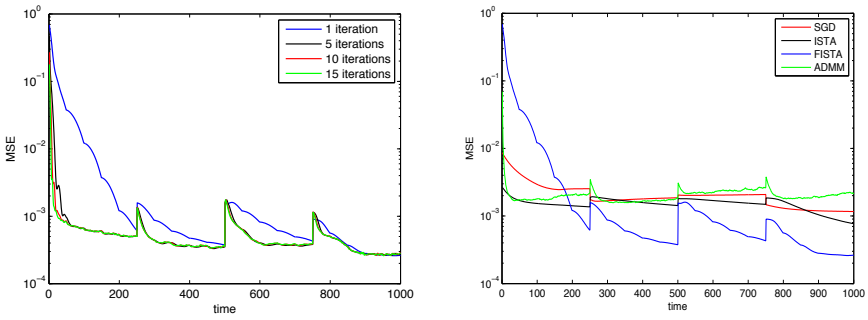
$$\mathbf{U}[k] := \mathbf{V}[k-1] + \left(\frac{c[k-1]-1}{c[k]}\right)(\mathbf{V}[k-1] - \mathbf{V}[k-2]) \tag{10.5.18}$$

$$c[k] = \frac{1+\sqrt{4c^2[k-1]+1}}{2}. \tag{10.5.19}$$

The accelerated PG algorithm [a.k.a. fast (F)ISTA] utilizes a linear combination of the previous two iterates $\{\mathbf{V}[k-1], \mathbf{V}[k-2]\}$. The iteration-dependent combination weights are a function of the scalar sequence (10.5.19). FISTA affords a (worst-case) convergence rate guarantee of $\mathcal{O}(1/\sqrt{\epsilon})$ iterations to return an ϵ-optimal solution measured by its objective value (ISTA instead affords $\mathcal{O}(1/\epsilon)$) [BT09, Nes05]. With a few minor changes, (10.5.12)-(10.5.15) can be modified to attain accelerated convergence (see [BMG14] for details). A slight compromise to adopting FISTA is the increased memory cost for storing the two prior estimates of $\mathbf{A}$ and $\mathbf{B}$.

10.5.3 Real-time operation

Under streaming big data settings, it may be impractical to run multiple inner (F)ISTA iterations per time interval in the quest for convergence. Infact a high-quality answer obtained slowly may not be as valuable as a medium-quality answer that is obtained quickly. The remainder of this section focuses on strategies for online operation of the topology tracking algorithms, namely: i) termination of inner iterations prematurely; and ii) pursuing stochastic gradient iterations.



a) b)

FIGURE 10.5.2: a) MSE (i.e., $\sum_{i,j}(\hat{a}_{ij}^t - a_{ij}^t)^2/N^2$) performance of Algorithm 10.5.1 versus time. For each t, (10.5.5) is solved "inexactly" for $k = 1$, 5, 10, and 15 inner iterations. It is apparent that $k = 5$ iterations suffice to attain convergence to the minimizer of (10.5.5) per t, especially after a short transient where the warm-restarts offer increasingly better initializations. b) MSE performance of real-time algorithms versus time. Real-time FISTA, Algorithm 10.5.2 (SGD), as well as inexact versions of Algorithm 10.5.1 (ISTA) and the ADMM solver in [BMG13] are compared.

10.5.3.1 Premature termination

Consider a scenario where the underlying network processes are stationary (or piecewise stationary with sufficiently long coherence time). Premature termination is justified by the fact that the solution of (10.5.5) for each $t = 1, \ldots, T$ does not need to be very accurate since it is just an intermediate step in the outer loop matched to the time-instants of data acquisition. In fact, it may be reasonable to run a single inner-iteration (so that k coincides with the time index t).

For synthetically-generated data according to the setup described in [BMG14], Figure 10.5.2 shows the time evolution of the mean-square error (MSE) estimation performance upon running FISTA. For each time interval t, (10.5.5) is solved "inexactly" after running only $k = 1, 5, 10$ and 15 inner iterations. Note that realtime operation corresponds to $k = 1$.

10.5.3.2 Stochastic gradient descent iterations

Supposing $\beta = 0$ in (10.5.5), the resulting cost function can be expressed as $f_t(\mathbf{V}) + g(\mathbf{V})$, where $\mathbf{V} := [\mathbf{A}\ \mathbf{B}]$ and $f_t(\mathbf{V}) := (1/2)\|\mathbf{Y}^t - \mathbf{A}\mathbf{Y}^t - \mathbf{B}\mathbf{X}\|_F^2$ only accounts for data acquired during time interval t. Solving the simplified optimization problem based only on instantaneous data can be accomplished by following stochastic gradient descent (SGD) iterations whose simplicity and tracking capabilities are well documented. Thus, one obtains the following

Algorithm 10.5.2: SGD algorithm for topology tracking

Require: $\{\mathbf{Y}^t\}_{t=1}^T$, $\mathbf{X}$, η.
Initialize $\mathbf{A}[1] = \mathbf{0}_{N\times N}, \mathbf{B}[1] = \mathbf{I}_N, \lambda_1$.
for $t = 1, \ldots, T$ **do**
  Update λ_t.
  for $i = 1 \ldots N$ (in parallel) **do**
    Form gradients at $\mathbf{a}_{-i}[t]$ and $b_{ii}[t]$ via (10.5.20)-(10.5.21).
    Update $\mathbf{a}_{-i}[t+1]$ via (10.5.22).
    Update $b_{ii}[t+1]$ via (10.5.23).
    Update $\mathbf{a}_i[t+1]$ via (10.5.16).
  end for
  return $\hat{\mathbf{A}}^t = \mathbf{A}[t+1], \hat{\mathbf{B}}^t = \mathbf{B}[t+1]$.
end for

updates:

$$\nabla_{\mathbf{a}_{-i}} f_t[t] = \mathbf{Y}^t_{-i}\left((\mathbf{Y}^t_{-i})^T\mathbf{a}_{-i}[t] + \mathbf{x}_i b_{ii}[t] - \mathbf{y}^t_i\right) \tag{10.5.20}$$

$$\nabla_{b_{ii}} f_t[t] = \mathbf{a}^T_{-i}[t]\mathbf{Y}^t_{-i}\mathbf{x}_i + b_{ii}[t]\|\mathbf{x}_i\|^2 - (\mathbf{y}^t_i)^T\mathbf{x}_i \tag{10.5.21}$$

$$\mathbf{a}_{-i}[t+1] = \mathcal{S}_{\lambda_t/\eta}\left(\mathbf{a}_{-i}[t] - \eta\nabla_{\mathbf{a}_{-i}} f_t[t]\right) \tag{10.5.22}$$

$$b_{ii}[t+1] = b_{ii}[t] - \eta\nabla_{b_{ii}} f_t[t]. \tag{10.5.23}$$

Compared to the parallel ISTA iterations in Algorithm 10.5.1 [cf. (10.5.12)-(10.5.14)], three main differences are noteworthy: (i) iterations k are merged with the time intervals t of data acquisition; (ii) the stochastic gradients $\nabla_{\mathbf{a}_{-i}} f_t[t]$ and $\nabla_{b_{ii}} f_t[t]$ involve the (noisy) data $\{\mathbf{Y}^t(\mathbf{Y}^t)^T, \mathbf{Y}^t\}$ instead of their time-averaged counterparts $\{\mathbf{\Sigma}^t, \bar{\mathbf{Y}}^t\}$; and (iii) a generic constant step-size η is utilized for the gradient descent steps.

The overall SGD algorithm is tabulated under Algorithm 10.5.2. Accelerated versions could be developed as well, at the expense of a marginal increase in computational complexity and doubling of memory requirements.

10.5.4 Experiments on real data

The tracking algorithms were tested on real cascade data obtained by monitoring blog posts and news articles on the web between March 2011 and February 2012 (45 weeks) [LK14]. Popular textual phrases (a.k.a. *memes*) due to globally-popular topics during this period were identified and the times when they were mentioned on the websites were recorded as Unix timestamps (i.e., number of hours since midnight on January 1, 1970). In order to test the tracking algorithms, cascade traces related to two keywords were extracted: i) "Kim Jong-un" the current leader of North Korea whose popularity rose after the death of his father (and predecessor); and ii) "Reid Hoffman" the founder of *LinkedIn*. Only significant cascades that propagated to at least 7

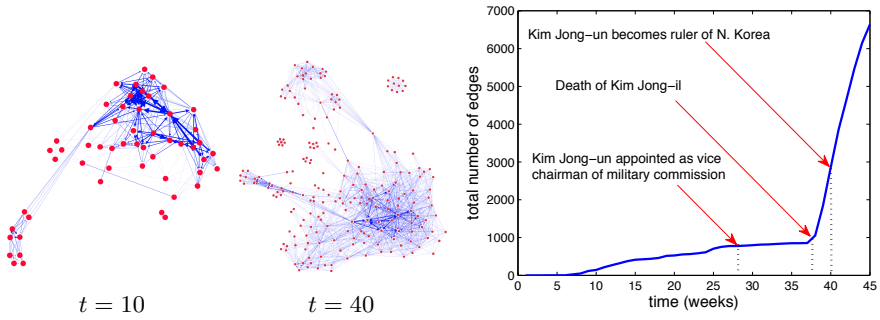


FIGURE 10.5.3: (Top) Visualization of estimated networks from information cascades related to the topic "Kim Jong-un" at $t = 10$ and $t = 40$ weeks. (Bottom) Evolution of total number of inferred edges [BMG14].

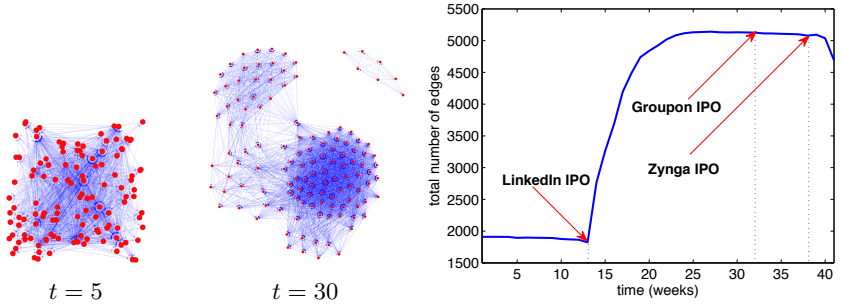


FIGURE 10.5.4: (Top) Visualization of estimated networks obtained by tracking "Reid Hoffman" cascades at $t = 5$ and $t = 30$ weeks. (Bottom) Evolution of total number of inferred edges [BMG14].

websites were retained. This resulted in $N = 360$ websites, $C = 466$ cascades, and $T = 45$ weeks for "Kim Jong-un" memes. Similarly, $N = 125$, $C = 85$, and $T = 41$ weeks for "Reid Hoffman".

In cases where website i made no mention of cascade c during interval t, y_{ic}^t was set to $100t_{\max}$ (i.e., a large number), where $t_{\max}$ denotes the largest timestamp in the dataset. The entries of marix $\mathbf{X}$ typically capture prior knowledge about susceptibility of nodes to contagions. In the web context, $\mathbf{x}_{ic}$ could be aptly set to the average search engine ranking of website i on keywords pertaining to c. In the absence of such real data for the observation interval, the entries of $\mathbf{X}$ were uniformly sampled over the interval $[0, 0.01]$.

Experimental results. Algorithm 10.5.1 was run on both datasets with $\beta = 0.9$ and $\lambda_t = 100$. Figure 10.5.3 (top) depicts drawings of the inferred network for Kim Jong-un at $t = 10$ and $t = 40$ weeks. Speculation about the possible successor of the dying North Korean ruler, Kim Jong-il, rose until his death on December 17, 2011 (week 38). He was succeeded by Kim Jong-un on

December 30, 2011 (week 40). The visualizations show an increasing number of edges over the 45 weeks, illustrating the growing interest of international news websites and blogs in the new ruler, about whom little was known in the first 10 weeks. Unfortunately, the observation horizon does not go beyond $T = 45$ weeks. A longer span of data would have been useful to investigate the rate at which global news coverage on the topic eventually subsided.

Figure 10.5.3 (bottom) depicts the time evolution of the total number of edges in the inferred dynamic network. Of particular interest are the weeks during which: i) Kim Jong-un was appointed as the vice chairman of the North Korean military commission; ii) Kim Jong-il died; and iii) Kim Jong-un became the ruler of North Korea. These events were the topics of many online news articles and political blogs, an observation that is reinforced by the experimental results shown in the plot.

The results of running Algorithm 10.5.1 on the second dataset are shown in Figure 10.5.4. Although Reid Hoffman was already popular in technology media coverage, his visibility in popular news and blogs increased tremendously following the highly successful initial public offering (IPO) of LinkedIn on May 19, 2011. Towards the end of 2011, a number of other successful technology companies like Groupon and Zynga went public, possibly stabilizing the amount of media coverage on Reid Hoffman. In fact, the drop in number of edges towards week 41 could be attributed to the captivation of media attention by the IPOs that occurred later in the year.

10.6 Conclusion

Big data analytics has risen to prominence within the last few years, and it remains a very active research area for the foreseeable future. In parallel, computational analysis of social networks has recently emerged as a versatile, cross-disciplinary field. Interestingly, a number of problems encountered in mining the web, learning and prediction of consumer behavior, and the dynamics of the spread of infectious diseases, all lie at the intersection of social networks, big data, and efficient (online) optimization.

Towards addressing these big data problems, signal processing and machine learning offer a robust framework for advanced data analytics. A fair and totally balanced survey of all pertinent issues and approaches is impossible within the scope of one chapter. Nevertheless, this chapter presented several interesting problems of recent interest within the social media community, namely: visualization of large social graphs, inference and imputation over social networks, community discovery, and tracking dynamic network topologies from information cascades. Efficient algorithms scaling well under big data settings along with experimental tests have been presented, and

wherever possible, references to contemporary and prior approaches have been highlighted.

10.7 Acknowledgments

The authors wish to thank the following friends, colleagues, and co-authors who contributed to their joint publications from which the material of this chapter was extracted: Drs. J. A. Bazerque, P. Forero, S. -J. Kim, M. Mardani, K. Rajawat, and K. Slavakis. The work was supported in part by NSF grants 1343248, 1423316, 1442686, and Eager 1500713; as well as the MURI grant no. AFOSR FA9550-10-1-0567; and the NIH grant no. 1R01GM10GM4975-01.

Bibliography

[ABG10] D. Angelosante, J. A. Bazerque, and G. B. Giannakis. Online adaptive estimation of sparse signals: where RLS meets the ℓ_1-norm. *IEEE Transactions on Signal Processing*, 58:3436–3447, 2010.

[AG11] D. Angelosante and G. B. Giannakis. Sparse graphical modeling of piecewise-stationary time series. In *Proc. of International Conference on Acoustics, Speech and Signal Processing*, pages 1960–1963, 2011.

[AHDBV06] J. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani. Large scale networks fingerprinting and visualization using the k-core decomposition. *Advances in Neural Information Processing Systems*, 18:41–50, 2006.

[AMF12] L. Akoglu, M. McGlohon, and C. Faloutsos. OddBall: Spotting anomalies in weighted graphs. In *Proc. Pacific Asia Knowledge Discovery and Data Mining*, pages 410–421, 2012.

[BBG13] J. A. Bazerque, B. Baingana, and G. B. Giannakis. Identifiability of sparse structural equation models for directed and cyclic networks. In *Proc. of Global Conference on Signal and Info. Processing*, pages 839–842, 2013.

[BC03] U. Brandes and S. R. Corman. Visual unrolling of network evolution and the analysis of dynamic discourse. *Information Visualization*, 2(1):40–50, 2003.

[Ber99] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[BG05] I. Borg and P. J. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

[BG13] B. Baingana and G. B. Giannakis. Centrality-constrained graph embedding. In *Proc. of the 38th International Conference on Acoustics, Speech and Signal Processing*, pages 3113–3117, 2013.

[BG15] B. Baingana and G. B. Giannakis. Kernel-based embeddings for large graphs with centrality constraints. In *Proc. of the 40th*

International Conference on Acoustics, Speech and Signal Processing, 2015.

[BMG13] B. Baingana, G. Mateos, and G. B. Giannakis. Dynamic structural equation models for tracking topologies of social networks. In *Proc. of 5th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, pages 292–295, 2013.

[BMG14] B. Baingana, G. Mateos, and G. B. Giannakis. Proximal-gradient algorithms for tracking cascades over social networks. *IEEE Journal of Selected Topics in Signal Processing*, 8(4):563–575, 2014.

[BNS06] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[BP98] S. Brin and L. Page. The anatomy of a large scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107–117, 1998.

[BP11] U. Brandes and C. Pich. More flexible radial layout. *Journal of Graph Algorithms and Applications*, 15:157–173, 2011.

[BPC+11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3:1–122, 2011.

[BT99] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena-Scientific, second edition, 1999.

[BT09] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2:183–202, 2009.

[BW97] U. Brandes and D. Wagner. A Bayesian paradigm for dynamic graph layout. In *Proc. of the 5th International Symposium on Graph Drawing*, pages 236–247, 1997.

[BW98] U. Brandes and D. Wagner. Dynamic grid embedding with few bends and changes. In *Proc. of the 9th Annual International Symposium on Algorithms and Computation*, pages 89–98, 1998.

[CBG13] X. Cai, J. A. Bazerque, and G. B. Giannakis. Inference of gene regulatory networks with sparse structural equation models exploiting genetic perturbations. *PLoS Computational Biology*, 9(5):1–13, 2013.

[CDK+99] S. Chakrabarti, B. E. Dom, S. R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, and J. Kleinberg. Mining the web's link structure. *IEEE Computer*, 32(8):60–67, 1999.

[CG84] A. Chistov and D. Grigorev. Complexity of quantifier elimination in the theory of algebraically closed fields. In *Math. Found. of Computer Science*, volume 176 of *Lecture Notes in Computer Science*, pages 17–31. Springer, 1984.

[CGH09] Y. Chen, Y. Gu, and A. O. Hero III. Sparse LMS for system identification. In *Proc. of International Conference on Acoustics, Speech and Signal Processing*, pages 3125–3128, 2009.

[CJHJ11] R. Chitta, R. Jin, T. C. Havens, and A. K. Jain. Approximate kernel k-means: Solution to large scale kernel clustering. In *Proc. of the 17th ACM International Conference on Knowledge discovery and data mining*, pages 895–903, 2011.

[CLL+10] J. Chen, W. Li, A. Lau, J. Cao, and K. Eang. Automated load curve data cleansing in power systems. *IEEE Transactions on Smart Grid*, 1(1):213–221, 2010.

[CLMW11] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(1):1–37, 2011.

[CPR07] M. Coates, Y. Pointurier, and M. Rabbat. Compressed network monitoring for IP and all-optical networks. In *Proc. ACM Internet Measurement Conference*, pages 241–252, 2007.

[CSB+11] W. Y. Chen, Y. Song, H. Bai, C. J. Lin, and E. Y. Chang. Parallel spectral clustering in distributed systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):568–586, 2011.

[CSPW11] V. Chandrasekaran, S. Sanghavi, P. R. Parrilo, and A. S. Willsky. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21(2):572–596, 2011.

[CW08] E. Candès and M. B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008.

[DBD13] U. Dogrusoz, M. E. Belviranli, and A. Dilek. Cise: A circular spring embedder layout algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 19:953–966, 2013.

[DDM04] I. Daubechies, M. Defrise, and C. D. Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57:1413–1457, 2004.

[DG08] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[DGK04] I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: Spectral clustering and normalized cuts. In *Proc. of the 10th ACM International Conference on Knowledge Discovery and Data Mining*, pages 551–556, 2004.

[DGK07] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, 2007.

[Don06] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289 –1306, 2006.

[EFKK14] A. Elgohary, A. K. Farahat, M. S. Kamel, and F. Karray. Embed and conquer: Scalable embeddings for kernel k-means on MapReduce. In *SIAM International Conference on Data Mining*, 2014.

[EH07] W. Eberle and L. B. Holder. Discovering structural anomalies in graph-based data. In *Proc. of International Conference on Data Mining*, pages 393–398, 2007.

[EK10] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

[FBCM04] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.

[FFF99] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. In *Proc. of Special Interest Group on Data Communications*, pages 251–262, 1999.

[FKG11] P. Forero, V. Kekatos, and G. B. Giannakis. Outlier-aware robust clustering. In *Proc. of International Confernce on Acoustics, Speech and Signal Processing*, pages 2244–2247, 2011.

[For10] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[FPRS07] F. Fouss, A. Pirotte, J. M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. on Knowledge and Data Engineering*, 19:355–369, 2007.

[Fre77] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41, 1977.

[FRG14] P. Forero, K. Rajawat, and G. B. Giannakis. Prediction of partially observed dynamical processes over networks via dictionary learning. *IEEE Transactions on Signal Processing*, 62(13):3305–3320, 2014.

[FT08] Y. Frishman and A. Tal. Online dynamic graph drawing. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):727–740, 2008.

[GKB13] A. Gittens, P. Kambadur, and C. Boutsidis. Approximate spectral clustering via randomized sketching. *Computing Research Repository*, 2013.

[GL12] G. H. Golub and C. F. V. Loan. *Matrix Computations*, volume 3. JHU Press, 2012.

[GN02] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[Gol72] A. S. Goldberger. Structural equation methods in the social sciences. *Econometrica*, 40:979–1001, 1972.

[Het00] H. W. Hethcote. The mathematics of infectious diseases. *SIAM Review*, 42(4):599–653, 2000.

[HL12] L. Harrison and A. Lu. The future of security visualization: Lessons from network visualization. *IEEE Network*, 26:6–11, 2012.

[HTF09] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, second edition, 2009.

[Int] Internet2. Internet2 network flow data. http://www.internet2.edu. Accessed: 2014.

[Jac10] M. O. Jackson. *Social and Economic Networks*. Princeton University Press, 2010.

[Kap09] D. Kaplan. *Structural Equation Modeling: Foundations and Extensions*. Sage Publications, second edition, 2009.

[KK89] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31:7–15, 1989.

[KM08] D. Kempe and S. McSherry. A decentralized algorithm for spectral analysis. *Journal of Computer and Systems Sciences*, 74:70–83, 2008.

[Kol09] E. D. Kolaczyk. *Statistical Analysis of Network Data: Methods and Models*. Springer, 2009.

[KST11] Y. Kopsinis, K. Slavakis, and S. Theodoridis. Online sparse system identification and signal reconstruction using projections onto weighted ℓ_1 balls. *IEEE Transactions on Signal Processing*, 59:936–952, 2011.

[KTF09] U. Kang, C. E. Tsourakakis, and C. Faloutsos. PEGASUS: A peta-scale graph mining system – implementation and observations. In *Proc. of International Conference on Data Mining*, pages 229–238, 2009.

[LC10] F. Lin and W. W. Cohen. Power iteration clustering. In *Proc. of the 27th International Conference on Machine Learning*, pages 655–662, 2010.

[LCD04] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *Proc. of the 2004 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 219–230, 2004.

[LdH08] B. Liu, A. de la Fuente, and I. Hoeschele. Gene network inference via structural equation modeling in genetical genomics experiments. *Genetics*, 178:1763–1776, 2008.

[Les11] J. Leskovec. General relativity and quantum cosmology collaboration network. *Stanford Network Analysis Project*, 2011.

[Les12] J. Leskovec. Social circles: Facebook. *Stanford Network Analysis Project*, 2012.

[LGW+11] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. Ma. Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. *UIUC Tech. Report UILU-ENG-09-2214*, 2011.

[LK14] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[LKF07] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007.

[LS08] L. Leydesdorff and T. Schank. Dynamic animations of journal maps: Indicators of structural changes and interdisciplinary developments. *Journal of the American Society for Information Science and Technology*, 59(11):1810–1818, 2008.

[LSY98] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users' Guide: Solution of Large-scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, volume 6. SIAM, 1998.

[Lux07] U. V. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[LWH03] B. Luo, R. C. Wilson, and E. R. Hancock. Spectral embedding of graphs. *Pattern Recognition*, 36:2213–2230, 2003.

[MAB13] B. A. Miller, N. Arcolano, and N. T. Bliss. Efficient anomaly detection in dynamic, attributed graphs: Emerging phenomena and big data. In *Proc. International Conference Intelligence and Security Informatics*, pages 179–184, 2013.

[Mah11] M. W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3(2):123–224, 2011.

[MBG10] G. Mateos, J. A. Bazerque, and G. B. Giannakis. Distributed sparse linear regression. *IEEE Transactions on Signal Processing*, 58(10):5262–5276, 2010.

[MBPS10] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

[MG12a] G. Mateos and G. B. Giannakis. Robust nonparametric regression via sparsity control with application to load curve data cleansing. *IEEE Transactions on Signal Processing*, 60(4):1571–1584, 2012.

[MG12b] G. Mateos and G. B. Giannakis. Robust PCA as bilinear decomposition with outlier-sparsity regularization. *IEEE Transactions on Signal Processing*, 60(10):5176–5190, 2012.

[MG13] G. Mateos and G. B. Giannakis. Load curve data cleansing and imputation via sparsity and low rank. *IEEE Transactions on Smart Grid*, 4(4):2347–2355, 2013.

[ML13] S. Meyers and J. Leskovec. On the convexity of latent social network inference. In *Proc. of Neural Information Processing Systems*, pages 1741–1749, 2013.

[MM13] S. Mankad and G. Michailidis. Structural and functional discovery in dynamic networks with non-negative matrix factorization. *Physical Review E*, 88(4):042812, 2013.

[MMBd05] J. Moody, D. McFarland, and S. Bender-deMoll. Dynamic network visualization. *American Journal of Sociology*, 110(4):1206–1241, 2005.

[MMBd06] J. Moody, D. McFarland, and S. Bender-deMoll. The art and science of dynamic network visualization. *Journal of Social Structure*, 7(2):1–38, 2006.

[MMG13a] M. Mardani, G. Mateos, and G. B. Giannakis. Decentralized sparsity-regularized rank minimization: Algorithms and applications. *IEEE Transactions on Signal Processing*, 61:5374–5388, 2013.

[MMG13b] M. Mardani, G. Mateos, and G. B. Giannakis. Dynamic anomalography: Tracking network anomalies via sparsity and low rank. *IEEE Journal of Selected Topics in Signal Processing*, 7:50–66, 2013.

[MMG13c] M. Mardani, G. Mateos, and G. B. Giannakis. Recovery of low-rank plus compressed sparse matrices with application to unveiling traffic anomalies. *IEEE Transactions on Information Theory*, 59:5186–5205, 2013.

[MR13] G. Mateos and K. Rajawat. Dynamic network cartography. *IEEE Signal Processing Magazine*, 30(3):129–143, 2013.

[MT08] H. D. K. Moonesinghe and P.-N. Tan. OutRank: A graph-based outlier detection framework using random walks. *International Journal on Artificial Intelligence Tools*, 17(1):1–18, 2008.

[Mut84] B. Muthén. A general structural equation model with dichotomous, ordered categorical, and continuous latent variable indicators. *Pyschometrika*, 49:115–132, 1984.

[Nat95] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24:227–234, 1995.

[NC03] C. C. Noble and D. J. Cook. Graph-based anomaly detection. In *Proc. of Special Interest Group on Knowledge Discovery and Data Mining*, pages 631–636, 2003.

[Nes83] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27:372–376, 1983.

[Nes05] Y. Nesterov. Smooth minimization of nonsmooth functions. *Mathematical Programming*, 103:127–152, 2005.

[New10] M. E. J. Newman. *Networks: An Introduction*. Oxford University Press, 2010.

[NJW+02] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 2, pages 849–856, 2002.

[OF97] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997.

[PB13] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1:123–231, 2013.

[PPY13] Y. Park, C. E. Priebe, and A. Youssef. Anomaly detection in times series of graphs using fusion of graph invariants. *IEEE Journal of Selected Topics in Signal Processing*, 7(1):67–75, 2013.

[PT98] A. Papakostas and I. Tollis. Algorithms for area-efficient orthogonal drawings. *Computational Geometry: Theory and Applications*, 9:83–110, 1998.

[RBL+07] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proc. of the 24th International Conference on Machine learning*, pages 759–766, 2007.

[RBS11] M. G. Rodriguez, D. Balduzzi, and B. Scholkopf. Uncovering the temporal dynamics of diffusion networks. In *Proc. of 28th International Conference on Machine Learning*, 2011.

[RLS10] M. G. Rodriguez, J. Leskovec, and B. Scholkopf. Structure and dynamics of information pathways in online media. In *Proc. of 6th ACM International Conference on Web Search and Data Mining*, 2010.

[Rog95] E. M. Rogers. *Diffusion of Innovations*. Free Press, fourth edition, 1995.

[Rou10] M. Roughan. A case study of the accuracy of SNMP measurements. *Journal of Electrical and Computer Engineering*, 2010.

[RR13] B. Recht and C. Re. Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation*, 5(2):201–226, 2013.

[Sab66] G. Sabidussi. The centrality index of a graph. *Psychometrika*, 31:581–683, 1966.

[SGM14] K. Slavakis, G. B. Giannakis, and G. Mateos. Modeling and optimization for big data analytics. *IEEE Signal Processing Magazine*, 31:18–31, 2014.

[SI09] T. Sakai and A. Imiya. Fast spectral clustering with random projection and sampling. In *Proc. of the 6th International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 372–384, 2009.

[SJ09] B. Shaw and T. Jebara. Structure preserving embedding. In *Proc. of International. Conference on Machine Learning*, pages 937–944, 2009.

[SM00] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[Sni98] M. Snir. *MPI–the Complete Reference: The MPI Core*, volume 1. MIT press, 1998.

[SQCF05] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *Proc. of International Conference on Data Mining*, 2005.

[SRJ04] N. Srebro, J. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *Proc. Advances in Neural Information Processing Systems*, pages 1329–1336, 2004.

[SS05] N. Srebro and A. Shraibman. Rank, trace-norm and max-norm. In *Proc. of Learning Theory*, pages 545–560. Springer, 2005.

[SS11] S. Shalev-Schwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2011.

[SSM98] B. Scholkopf, A. J. Smola, and K. R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[ST11] O. Shamir and N. Tishby. Spectral clustering on a budget. In *Intl. Conf. on Artificial Intelligence and Statistics*, pages 661–669, 2011.

[TF10] I. Tošić and P. Frossard. Dictionary learning. *IEEE Signal Processing Magazine*, 28:27–38, 2010.

[TL11] H. Tong and C.-Y. Lin. Non-negative residual matrix factorization with application to graph anomaly detection. In *Proc. of SIAM Conference on Data Mining*, pages 143–153, 2011.

[TSG15] P. Traganitis, K. Slavakis, and G. B. Giannakis. Big data spectral clustering via sketching and validation. *IEEE Journal of Selected Topics in Signal Processing*, 2015.

[VR07] F. Vega-Redondo. *Complex Social Networks*. Cambridge University Press, 2007.

[Wah90] G. Wahba. *Spline Models for Observational Data*, volume 59. SIAM, 1990.

[WLRB09] L. Wang, C. Leckie, K. Ramamohanarao, and J. Bezdek. Approximate spectral clustering. In *Proc. of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 134–146, 2009.

[WS98] D. J. Watts and S. H. Strogatz. Collective dynamics of small world networks. *Nature*, 393(6684):440–442, 1998.

[WTPP14] H. Wang, M. Tang, Y. Park, and C. E. Priebe. Locality statistics for anomaly detection in time series of graphs. *IEEE Transactions on Signal Processing*, 62(3):703–717, 2014.

[XKI12] K. S. Xu, M. Kliger, and A. O. H. III. A regularized graph layout framework for dynamic network visualization. *Data Mining and Knowledge Discovery*, 27(1):84–116, 2012.

[YHJ09] D. Yan, L. Huang, and M. I. Jordan. Fast approximate spectral clustering. In *Proc. of the 15th ACM International Conference on Knowledge Discovery and Data Mining*, pages 907–916, 2009.

[YL06] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B*, 68:49–67, 2006.

[YLZ+13] J. Yang, Y. Liu, X. Zhang, X. Yuan, Y. Zhao, S. Barlowe, and S. Liu. Piwi: Visually exploring graphs based on their community structure. *IEEE Transactions on Visualization and Computer Graphics*, 19:1034–1047, 2013.

[YY13] X. M. Yuan and J. Yang. Sparse and low-rank matrix decomposition via alternating direction method. *Pacific Journal of Optimization*, 9(1):167–180, 2013.

[ZGGR05] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan. Network anomography. In *Proc. of ACM Internet Measurement Conference*, pages 30–30, 2005.

[ZLW+10] Z. Zhou, X. Li, J. Wright, E. Candès, and Y. Ma. Stable principal component pursuit. In *Proc. of International Symposium on Information Theory*, pages 1518–1522, 2010.

[ZMH09] W. Zhao, H. Ma, and Q. He. Parallel k-means clustering based on MapReduce. In *Proc. of the 1st International Conference on Cloud Computing*, pages 674–679, 2009.

[ZRLD05] Y. Zhang, M. Roughan, C. Lund, and D. L. Donoho. Estimating point-to-point and point-to-multipoint traffic matrices: An information-theoretic approach. *IEEE/ACM Transactions on Networking*, 13(5):947 – 960, Oct. 2005.

[ZRWQ09] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu. Spatio-temporal compressive sensing and Internet traffic matrices. In *Proc. of ACM SIGCOM Conference on Data Commun.*, pages 267–278, 2009.