CHAPTER

Big Data

1

Morteza Mardani[*], Gonzalo Mateos[**] and Georgios B. Giannakis[a,†]

[*] *Stanford University, Department of Electrical Engineering, 350 Serra Mall, Stanford, CA, 94305*
[**] *University of Rochester, Department of Electrical and Computer Engineering, 413 Hopeman, Rochester, NY, 14627* [†] *University of Minnesota, Department of Electrical and Computer Engineering, 200 Union Street SE, Minneapolis, MN 55455*
[a] *Corresponding:* `georgios@umn.edu`

CHAPTER OUTLINE HEAD

1.1 LEARNING FROM BIG DATA: OPPORTUNITIES AND CHALLENGES

We live in an era of "data deluge." Pervasive sensors collect massive amounts of information on every bit of our lives, churning out enormous streams of raw data in a wide variety of formats. To get a sense of scale, users of the Facebook social network happily feed 10 billion messages per day, click the "like" button 4.5 billion times and

Table 1.1 Notation

$\mathbf{x}$	vector
$\mathbf{X}$	matrix
$\mathcal{X}$	set
$(\cdot)^\top$	matrix transpose
$\mathrm{tr}\{\cdot\}$	matrix trace
σ_i	i-th singular value of a matrix
$\|\mathbf{X}\|_F^2 := \mathrm{tr}\{\mathbf{X}^\top \mathbf{X}\}$	matrix Frobenius norm
$\|\mathbf{X}\|_* := \sum_i \sigma_i$	matrix nuclear norm
$\|\mathbf{X}\| := \max_i \sigma_i$	matrix spectral norm
$\|\mathbf{x}\|_2$	vector ℓ_2-norm
$\otimes$	Kronecker product
$\mathrm{vec}(\mathbf{X})$	concatenates columns of $\mathbf{X}$ on top of each other
$\mathrm{unvec}(\mathbf{x})$	unfolds $\mathbf{x}$ to a matrix
$\mathcal{O}(\cdot)$	order of operation count
$\lambda_{\min}$	minimum eigen value
∇f	gradient of f
$\nabla^2 f$	Hessian of f

upload 350 million new pictures each and every day. Consumer data are collected every time we browse or purchase products online, as business models aim to provide services that are increasingly personalized. Automated sensors capture essentially every snapshot of complex phenomena of interest through high-resolution measurements. Mining information from these large volumes of data is expected to bring significant science and engineering advances along with consequent improvements in quality of life.

While big data may bring "big blessings," there are formidable challenges in dealing with large-scale datasets [1]. The *sheer volume* of data makes it often impossible to run analytics using central processors and storage units. Ubiquitous network data are also *geographically spread*, and collecting the data might be infeasible due to communication costs or privacy concerns. Consequently, datasets are often *incomplete* and thus a sizable portion of entries are missing. Moreover, large-scale data are prone to contain *corrupted measurements*, communication errors, and even suffer from *anomalies* such as cyberattacks. Furthermore, as many sources continuously generate data in *real time*, analytics must often be performed online as well as without an opportunity to revisit past data.

Conventional statistical inference tools cope with the notorious curse of dimensionality as well as with corruptions and anomalies by exploiting latent structure (of low intrinsic-dimensionality) in the data. Such structure typically emerges due to dependencies present in real world signals. In large-scale networks, complex interactions spanning the social, temporal, and spatial dimensions render such graph-indexed data highly correlated. For instance, the origin-to-destination (OD) traffic flows in the backbone of Internet Protocol (IP) networks exhibit dependencies mainly due to traffic generation patterns [2], which can facilitate network monitoring tasks such as identifying traffic volume anomalies resulting from cyberattacks [3].

In this context, the goal of this chapter is to develop a framework for scalable decentralized and streaming analytics that facilitates machine learning for big data. For simplicity of exposition, the presentation focuses on the matrix completion problem with applications to IP network health monitoring. However, the scope of the ensuing framework can be broadened to accommodate other fundamental low-rank recovery tasks such as robust PCA and low-rank plus compressed-sparse recovery. Modern datasets are oftentimes indexed by several variables or dimensions giving rise to a multi-way array (or tensor), in general. This chapter focuses on two-way arrays, or matrices, but extension to tensors are possible. Readers interested in delving into these generalizations are referred to [4, 5, 6, 7, 8].

1.2 MATRIX COMPLETION AND NUCLEAR NORM

Let $\mathbf{X} := [x_{l,t}] \in \mathbb{R}^{L\times T}$ be a *low-rank* matrix [$\text{rank}(\mathbf{X}) \ll \min(L,T)$] and a set $\Omega \subseteq \{1,\dots,L\}\times\{1,\dots,T\}$ of index pairs (l,t) that define a sampling of the entries of $\mathbf{X}$. Given a number of (possibly) noise corrupted measurements

$$y_{l,t} = x_{l,t} + v_{l,t}, \quad (l,t)\in\Omega \tag{1.1}$$

the goal is to estimate low-rank $\mathbf{X}$, by denoising the observed entries and imputing the missing ones. Introducing the sampling operator $\mathcal{P}_\Omega(\cdot)$ which sets the entries of its matrix argument not in Ω to zero and leaves the rest unchanged, the data model can be compactly written in matrix form as

$$\mathcal{P}_\Omega(\mathbf{Y}) = \mathcal{P}_\Omega(\mathbf{X}+\mathbf{V}). \tag{1.2}$$

A natural estimator accounting for the low rank of $\mathbf{X}$ will be sought to fit the data $\mathcal{P}_\Omega(\mathbf{Y})$ in the least-squares (LS) error sense, as well as minimize the rank of $\mathbf{X}$; see e.g. [9]. However, minimizing the non-convex matrix rank demands combinatorial complexity, and it is NP-complete [10, 11]. Adopting the nuclear norm $\|\mathbf{X}\|_* := \sum_i \sigma_i(\mathbf{X})$ (σ_i signifies i-th singular value) as a convex surrogate of rank [12, 13], one is then motivated to solve

$$\text{(P1)} \qquad \min_{\mathbf{X}} \frac{1}{2}\|\mathcal{P}_\Omega(\mathbf{Y}-\mathbf{X})\|_F^2 + \lambda\|\mathbf{X}\|_* \tag{1.3}$$

where $\lambda \geq 0$ is the rank-controlling parameter. Being convex (P1) is appealing, and it offers well-documented guarantees for stable and exact recovery in numerous tasks such as matrix completion [14, 9], and low-rank plus (compressed) sparse matrix decomposition [15, 16, 17]. For the matrix completion setting, typical results establish that if the energy of $\mathbf{X}$ is sufficiently *spread out*, which can be fulfilled for instance when the singular vectors are non-spiky, then with only $O(Pr\log^2(P))$ randomly chosen matrix entries ($r :=$ rank($\mathbf{X}$) and $P := \max(L, T)$), one can accurately recover the missing entries in the complement of Ω [14, 9].

However, nuclear-norm regularization lacks separability across rows and columns of $\mathbf{X}$ because singular values $\sigma_i(\mathbf{X})$ depend on all entries of the matrix. This coupling challenges streaming and decentralized data analytics, where columns of $\mathbf{X}$ are acquired sequentially in time, or, rows of $\mathbf{X}$ are geographically dispersed throughout a network, respectively. The following section introduces an alternative characterization of the nuclear-norm that proves instrumental to develop scalable decentralized and online algorithms to tackle (P1).

Separable rank regularization. Being low-rank matrix $\mathbf{X}$ admits a bilinear factorization $\mathbf{X} = \mathbf{L}\mathbf{Q}^\top$, with factor matrices $\mathbf{L} \in \mathbb{R}^{L\times\rho}$ and $\mathbf{Q} \in \mathbb{R}^{T\times\rho}$. The value of $\rho \geq r$ is chosen sufficiently large to overestimate rank($\mathbf{X}$). Interestingly, the nuclear norm of $\mathbf{X}$ can be alternatively written as the solution of the following non-convex problem [18, 19]

$$\|\mathbf{X}\|_* := \min_{\{\mathbf{L},\mathbf{Q}\}} \ \frac{1}{2}\left\{\|\mathbf{L}\|_F^2 + \|\mathbf{Q}\|_F^2\right\}, \qquad \text{s. to} \quad \mathbf{X} = \mathbf{L}\mathbf{Q}^\top. \tag{1.4}$$

The optimization (1.4) is over all possible bilinear factorizations of $\mathbf{X}$, so that the number of columns of $\mathbf{L}$ and $\mathbf{Q}$ is also a variable. For an arbitrary matrix $\mathbf{X}$ with SVD $\mathbf{X} = \mathbf{U}_X\mathbf{\Sigma}_X\mathbf{V}_X^\top$, the minimum in (1.4) is attained for $\mathbf{L} = \mathbf{U}_X\mathbf{\Sigma}_X^{1/2}$ and $\mathbf{Q} = \mathbf{V}_X\mathbf{\Sigma}_X^{1/2}$. Establishing the uniqueness of such solution requires semidefinite programming (SDP) arguments [18].

Adopting this characterization is useful since the Frobenius norm cost in (1.4) is separable across the entries of the factor matrices, but it comes at the price of non-convexity for the corresponding recovery task. However, as argued later under certain conditions this choice comes with no loss of optimality. Next, we leverage (1.4) to obtain a separable cost equivalent to that in (P1), that can be minimized in a decentralized fashion via the alternating-direction method of multipliers [20, 21, 22].

1.3 DECENTRALIZED ANALYTICS

The matrix completion task in (P1) assumes that the samples $\mathcal{P}_\Omega(\mathbf{Y})$ are entirely available at a central processing unit, and they can be jointly processed to infer $\mathbf{X}$. Collecting the entire data is challenging in various applications, or, it can be even impossible e.g., in wireless sensor networks (WSNs) operating under stringent power budget constraints. In other cases such as the Internet or collaborative marketing studies, agents providing private data for e.g., fitting a low-rank preference model, may not be willing to share their training data but only the learning results. Performing the optimization in a centralized fashion raises robustness concerns as well, since the central processor represents an isolated point of failure.

Several customized iterative algorithms have been proposed to solve instances of (P1), and have been shown effective in tackling low- to medium-size problems; see e.g., [4, 14, 18]. However, most algorithms require computation of singular values per iteration and become prohibitively expensive when dealing with high-dimensional data as argued in [23]. In a similar vein, stochastic gradient algorithms were recently developed for large-scale problems entailing regularization with the nuclear norm [23, 6]. Even though iterations in [23] are highly paralellizable, they are not applicable to networks of arbitrary topology. The aforementioned reasons motivate well developing reduced-complexity *decentralized* algorithms for nuclear-norm minimization.

Network data model. Consider N networked agents capable of performing some local computations, as well as exchanging messages among directly connected neighbors. An agent should be understood as an abstract entity, e.g., a sensor in a WSN, or a router monitoring Internet traffic. The network is modeled as an undirected graph $G(\mathcal{N}, \mathcal{L})$, where the set of nodes $\mathcal{N} := \{1, \ldots, N\}$ corresponds to the network agents, and the edges (links) in $\mathcal{L} := \{1, \ldots, L\}$ represent pairs of agents that can communicate. Agent $n \in \mathcal{N}$ communicates with its single-hop neighboring peers in $\mathcal{J}_n$, and the size of the neighborhood will be henceforth denoted by $|\mathcal{J}_n|$. To ensure that the data from an arbitrary agent can eventually percolate through the entire network, it is assumed that graph G is connected; i.e., there exists a (possibly) multi-hop path connecting any two agents.

Decentralized matrix completion. With reference to the matrix completion task in (P1), in the network setting envisioned here each agent $n \in \mathcal{N}$ acquires a few incomplete and noise-corrupted rows of matrix $\mathbf{Y} \in \mathbb{R}^{L \times T}$. Specifically, the local data available to agent n is matrix $\mathcal{P}_{\Omega_n}(\mathbf{Y}_n)$, where $\mathbf{Y}_n \in \mathbb{R}^{L_n \times T}$, $\sum_{n=1}^{N} L_n = L$, and $\mathbf{Y} := \left[\mathbf{Y}_1^\top, \ldots, \mathbf{Y}_N^\top\right]^\top = \mathbf{X} + \mathbf{V}$. The index pairs in Ω_n are those in Ω for which the row index matches the rows of $\mathbf{Y}$ observed by agent n. With regards to the decision variables, partition also $\mathbf{X} := \left[\mathbf{X}_1^\top, \ldots, \mathbf{X}_N^\top\right]^\top \in \mathbb{R}^{L \times T}$ similar to $\mathbf{Y}$, where $\mathbf{X}_n \in \mathbb{R}^{L_n \times T}$, $n = 1, \ldots, N$. Agents collaborate to form the wanted estimator (P1) in a decentralized fashion, which can be equivalently rewritten as

$$\text{(P1)} \qquad \min_{\mathbf{X}} \sum_{n=1}^{N} \left[\frac{1}{2} \|\mathcal{P}_{\Omega_n}(\mathbf{Y}_n - \mathbf{X}_n)\|_F^2 + \frac{\lambda}{N} \|\mathbf{X}\|_* \right].$$

Our objective is to develop a decentralized matrix-completion (DMC) algorithm based on in-network processing of the locally available data. The described setup naturally suggests three features that the algorithm should exhibit: (f1) agent $n \in \mathcal{N}$ should obtain an estimate of $\mathbf{X}_n$, which coincides with the corresponding solution of the centralized estimator (P1) that uses the entire data $\mathcal{P}_\Omega(\mathbf{Y})$; (f2) processing per agent should be kept as simple as possible; and (f3) the overhead for inter-agent communications should be affordable and confined to single-hop neighborhoods.

To facilitate reducing the computational complexity and memory storage requirements of the decentralized algorithm sought, it is henceforth assumed that the decision variable $\mathbf{X}$ in (P1) has rank at most ρ. For instance, empirical analysis of real Internet traffic data has revealed that OD flow traffic matrices typically have rank$[\mathbf{X}] \in [5, 8]$; hence, one can safely choose $\rho = 10$ [24]. In addition, recall that the rank of the solution $\hat{\mathbf{X}}$ in (P1) is controlled by the choice of λ, and can be made small enough for sufficiently large λ. As argued next, the smaller the value of ρ, the more efficient the algorithm becomes.

Because rank$(\hat{\mathbf{X}}) \leq \rho$, (P1)'s search space is effectively reduced and one can factorize the decision variable as $\mathbf{X} = \mathbf{L}\mathbf{Q}^\top$, where $\mathbf{L}$ and $\mathbf{Q}$ are $L \times \rho$ and $T \times \rho$ matrices, respectively. Adopting this reparametrization of $\mathbf{X}$ in (P1) one obtains the following equivalent optimization problem

$$\text{(P2)} \qquad \min_{\{\mathbf{L},\mathbf{Q}\}} \sum_{n=1}^{N} \left[\frac{1}{2} \|\mathcal{P}_{\Omega_n}(\mathbf{Y}_n - \mathbf{L}_n\mathbf{Q}^\top)\|_F^2 + \frac{\lambda}{N} \|\mathbf{L}\mathbf{Q}^\top\|_* \right]$$

which is non-convex due to the bilinear terms $\mathbf{L}_n\mathbf{Q}^\top$, and $\mathbf{L} := \left[\mathbf{L}_1^\top, \ldots, \mathbf{L}_N^\top\right]^\top$. The number of variables is reduced from LT in (P1), to $\rho(L+T)$ in (P2). The savings can be significant when ρ is in the order of a few dozens, and both L and T are large. Problem (P2) is still not amenable to decentralized implementation due to: (i) the non-separable nuclear norm present in the cost function; and (ii) the global variable $\mathbf{Q}$ coupling the per-agent summands.

Leveraging (1.4), the following reformulation of (P2) provides an important first step towards obtaining a decentralized estimator:

$$\text{(P3)} \qquad \min_{\{\mathbf{L},\mathbf{Q}\}} \sum_{n=1}^{N} \left[\frac{1}{2} \|\mathcal{P}_{\Omega_n}(\mathbf{Y}_n - \mathbf{L}_n\mathbf{Q}^\top)\|_F^2 + \frac{\lambda}{2N} \left\{ N\|\mathbf{L}_n\|_F^2 + \|\mathbf{Q}\|_F^2 \right\} \right].$$

Building on (1.4) and since rank$(\hat{\mathbf{X}}) \leq \rho$, it readily follows that the separable Frobenius-norm regularization in (P3) comes with no loss of optimality, meaning that (P1) and (P3) admit identical solutions. This equivalence ensures that by finding the global minimum of (P3) [which can have significantly fewer variables than (P1)], one can recover the optimal solution of (P1). However, since (P3) is non-convex, it may have stationary points which need not be globally optimal. Interestingly, the next proposition offers a global optimality certificate for the stationary points of (P3). For the detailed proof, see [7, Appendix A].

Proposition 1. *Let* $\{\bar{\mathbf{L}}, \bar{\mathbf{Q}}\}$ *be a stationary point of (P3). If* $\|\mathcal{P}_\Omega(\mathbf{Y} - \bar{\mathbf{L}}\bar{\mathbf{Q}}^\top)\| \leq \lambda$ *(no subscript in* $\|.\|$ *signifies spectral norm), then* $\hat{\mathbf{X}} = \bar{\mathbf{L}}\bar{\mathbf{Q}}^\top$ *is the globally optimal solution of (P1).*

To decompose the cost function in (P3), in which summands are coupled through the global variables $\mathbf{Q}$, introduce auxiliary variables $\{\mathbf{Q}_n\}_{n=1}^N$ representing local estimates of $\mathbf{Q}$ per agent n. These local estimates are utilized to form the separable *constrained* minimization problem

$$\text{(P4)} \qquad \min_{\{\mathbf{L}_n,\mathbf{Q}_n\}} \sum_{n=1}^{N} \left[\frac{1}{2} \|\mathcal{P}_{\Omega_n}(\mathbf{Y}_n - \mathbf{L}_n \mathbf{Q}_n^{\top})\|_F^2 + \frac{\lambda}{2}\|\mathbf{L}_n\|_F^2 + \frac{\lambda}{2N}\|\mathbf{Q}_n\|_F^2 \right]$$
$$\text{s. t.} \qquad \mathbf{Q}_n = \mathbf{Q}_m, \quad m \in \mathcal{J}_n, \ n \in \mathcal{N}.$$

Clearly, (P3) and (P4) are equivalent optimization problems because the graph G is assumed to be connected. The equivalence should be understood in the sense that $\hat{\mathbf{Q}}_1 = \hat{\mathbf{Q}}_2 = \ldots = \hat{\mathbf{Q}}_N = \hat{\mathbf{Q}}$, where $\{\hat{\mathbf{Q}}_n\}_{n\in\mathcal{N}}$ and $\hat{\mathbf{Q}}$ are the optimal solutions of (P4) and (P3), respectively. Of course, the corresponding estimates of $\mathbf{L}$ will coincide as well. 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 (P3). To arrive at the desired decentralized algorithm, it is convenient to re-parameterize the consensus constraints in (P4) as

$$\mathbf{Q}_n = \bar{\mathbf{F}}_n^m, \ \mathbf{Q}_m = \tilde{\mathbf{F}}_n^m, \ \text{and } \bar{\mathbf{F}}_n^m = \tilde{\mathbf{F}}_n^m, \ m \in \mathcal{J}_n, \ n \in \mathcal{N} \tag{1.5}$$

where $\{\bar{\mathbf{F}}_n^m, \tilde{\mathbf{F}}_n^m\}_{n\in\mathcal{N}}^{m\in\mathcal{J}_n}$ are auxiliary optimization variables that will be eventually eliminated.

Alternating-direction method of multipliers. To tackle the constrained minimization problem (P4), associate dual variables $\bar{\mathbf{D}}_n^m$ and $\tilde{\mathbf{D}}_n^m$ with the consensus constraints in (1.5). Next introduce the quadratically *augmented* Lagrangian function

$$\begin{aligned}\mathcal{L}_c(\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3, \mathcal{M}) &= \sum_{n=1}^{N} \left[\frac{1}{2} \|\mathcal{P}_{\Omega_n}(\mathbf{Y}_n - \mathbf{L}_n \mathbf{Q}_n^{\top})\|_F^2 + \frac{\lambda}{2N}\{N\|\mathbf{L}_n\|_F^2 + \|\mathbf{Q}_n\|_F^2\} \right] \\ &+ \sum_{n=1}^{N} \sum_{m\in\mathcal{J}_n} \left\{ \langle \bar{\mathbf{C}}_n^m, \mathbf{Q}_n - \bar{\mathbf{F}}_n^m \rangle + \langle \tilde{\mathbf{C}}_n^m, \mathbf{Q}_m - \tilde{\mathbf{F}}_n^m \rangle \right\} \\ &+ \frac{c}{2} \sum_{n=1}^{N} \sum_{m\in\mathcal{J}_n} \left\{ \|\mathbf{Q}_n - \bar{\mathbf{F}}_n^m\|_F^2 + \|\mathbf{Q}_m - \tilde{\mathbf{F}}_n^m\|_F^2 \right\}\end{aligned} \tag{1.6}$$

where c is a positive penalty coefficient, and the primal variables are split into three groups $\mathcal{V}_1 := \{\mathbf{Q}_n\}_{n=1}^N$, $\mathcal{V}_2 := \{\mathbf{L}_n\}_{n=1}^N$, and $\mathcal{V}_3 := \{\bar{\mathbf{F}}_n^m, \tilde{\mathbf{F}}_n^m\}_{n\in\mathcal{N}}^{m\in\mathcal{J}_n}$. For notational convenience, collect all multipliers in $\mathcal{M} := \{\bar{\mathbf{C}}_n^m, \tilde{\mathbf{C}}_n^m,\}_{n\in\mathcal{N}}^{m\in\mathcal{J}_n}$. The remaining constraints in (1.5), namely $C_V := \{\bar{\mathbf{F}}_n^m = \tilde{\mathbf{F}}_n^m, \ m \in \mathcal{J}_n, \ n \in \mathcal{N}\}$, have not been dualized.

To minimize (P4) in a decentralized fashion, a variation of the alternating-direction method of multipliers (ADMM) will be adopted here. The ADMM is an iterative augmented Lagrangian method especially well-suited for parallel processing [20, 21], which has been proven successful to tackle the optimization tasks encountered e.g., with decentralized estimation problems [22, 25, 26]. The proposed solver entails an iterative procedure comprising four steps per iteration $k = 1, 2, \ldots$

[S1] Update dual variables for all $n \in \mathcal{N}$, $m \in \mathcal{J}_n$:

$$\bar{\mathbf{C}}_n^m[k] = \bar{\mathbf{C}}_n^m[k-1] + \mu(\mathbf{Q}_n[k] - \bar{\mathbf{F}}_n^m[k]) \tag{1.7}$$

$$\tilde{\mathbf{C}}_n^m[k] = \tilde{\mathbf{C}}_n^m[k-1] + \mu(\mathbf{Q}_m[k] - \tilde{\mathbf{F}}_n^m[k]) \tag{1.8}$$

[S2] Update the first group of primal variables:

$$\mathcal{V}_1[k+1] = \arg\min_{\mathcal{V}_1} \mathcal{L}_c\left(\mathcal{V}_1, \mathcal{V}_2[k], \mathcal{V}_3[k], \mathcal{M}[k]\right). \tag{1.9}$$

[S3] Update the second group of primal variables:

$$\mathcal{V}_2[k+1] = \arg\min_{\mathcal{V}_2} \mathcal{L}_c\left(\mathcal{V}_1[k+1], \mathcal{V}_2, \mathcal{V}_3[k], \mathcal{M}[k]\right). \tag{1.10}$$

[S4] Update the auxiliary primal variables:

$$\mathcal{V}_3[k+1] = \arg\min_{\mathcal{V}_3 \in C_V} \mathcal{L}_c\left(\mathcal{V}_1[k+1], \mathcal{V}_2[k+1], \mathcal{V}_3, \mathcal{M}[k]\right). \tag{1.11}$$

This four-step procedure implements a block-coordinate descent method with dual variable updates. At each step of minimizing the augmented Lagrangian, the variables not being updated are treated as fixed and are substituted with their most up-to-date values. Different from ADMM, the alternating-minimization step here generally cycles over three groups of primal variables $\mathcal{V}_1$-$\mathcal{V}_3$ (cf. two groups in ADMM [20]). In [S1], $\mu > 0$ is the step size of the subgradient ascent iterations on the dual problem. While it is common in ADMM implementations to select $\mu = c$, a distinction between the step size and the penalty parameter is made explicit here in the interest of generality.

Reformulating the estimator (P1) to its equivalent form (P4) renders the augmented Lagrangian in (1.6) highly decomposable. The separability comes in two flavors, both with respect to the variable groups $\mathcal{V}_1$-$\mathcal{V}_3$, as well as across the network agents $n \in \mathcal{N}$. This in turn leads to highly parallelized, simplified recursions corresponding to the aforementioned four steps. Specifically, it is shown in [7, Appendix B] that if the multipliers are initialized to zero, [S1]-[S4] constitute the DMC algorithm tabulated under Algorithm 1. Careful inspection of Algorithm 1 reveals that the inherently redundant auxiliary variables and multipliers $\{\bar{\mathbf{F}}_n^m, \tilde{\mathbf{F}}_n^m, \tilde{\mathbf{C}}_n^m\}$ have been eliminated. Agent n does not need to *separately* keep track of all its non-redundant multipliers $\{\bar{\mathbf{C}}_n^m\}_{m\in\mathcal{J}_n}$, but only to update their respective (scaled) sums $\mathbf{O}_n[k] := 2\sum_{m\in\mathcal{J}_n} \bar{\mathbf{C}}_n^m[k]$. To derive Algorithm 1 it is useful to recognize that linearity of $\mathcal{P}_{\Omega_n}$ implies that $\text{vec}(\mathcal{P}_{\Omega_n}(\mathbf{Z})) = \mathbf{\Omega}_n \text{vec}(\mathbf{Z})$, where $\mathbf{\Omega}_n \in \{0,1\}^{L_nT\times L_nT}$ is a diagonal matrix.

Computational and communication cost. The per-agent computational complexity of the DMC algorithm is dominated by repeated inversions of $\rho\times\rho$ and $\rho L_n \times \rho L_n$ matrices to obtain $\mathbf{E}_n[k+1]$ and $\mathbf{D}_n[k+1]$, respectively, and matrix multiplications to update $\mathbf{Q}_n[k+1]$ and $\mathbf{L}_n[k+1]$. Notice that $\mathbf{E}_n[k+1] \in \mathbb{R}^{\rho T\times\rho T}$ has block-diagonal structure with blocks of size $\rho\times\rho$. Overall, the per-iteration complexity across the network is upper bounded by $O(\rho^3 NT)$, which grows linearly with

Algorithm 1 : DMC algorithm per agent $n \in \mathcal{N}$

```
Input $\mathbf{Y}_n, \Omega_n, \lambda, c, \mu$
Initialize $\mathbf{O}[0] = \mathbf{0}_{T\times\rho}$, and $\mathbf{L}_n[1]$, $\mathbf{Q}_n[1]$ at random
for $k = 1, 2, \ldots$ do
    Receive $\{\mathbf{Q}_m[k]\}$ from neighbors $m \in \mathcal{J}_n$
    [S1] Update local dual variables:
    $\mathbf{O}_n[k] = \mathbf{O}_n[k-1] + \mu \sum_{m\in\mathcal{J}_n}(\mathbf{Q}_n[k] - \mathbf{Q}_m[k])$
    [S2] Update first group of local primal variables:
    $\mathbf{E}_n[k+1] = \left\{(\mathbf{I}_T \otimes \mathbf{L}_n^\top[k])\boldsymbol{\Omega}_n(\mathbf{I}_T \otimes \mathbf{L}_n[k]) + (\lambda/N + 2c|\mathcal{J}_n|)\mathbf{I}_{\rho T}\right\}^{-1}$
    $\mathbf{G}_n[k+1] := (\mathbf{I}_T \otimes \mathbf{L}_n^\top[k])\boldsymbol{\Omega}_n \mathrm{vec}(\mathbf{Y}_n) - \mathrm{vec}(\mathbf{O}_n^\top[k]) + c\,\mathrm{vec}(\sum_{m\in\mathcal{J}_n}(\mathbf{Q}_n^\top[k] + \mathbf{Q}_m^\top[k]))$
    $\mathbf{Q}_n^\top[k+1] = \mathrm{unvec}(\mathbf{E}_n[k+1]\mathbf{G}_n[k+1])$
    [S3] Update second group of local primal variables:
    $\mathbf{D}_n[k+1] := \left\{(\mathbf{Q}_n^\top[k+1] \otimes \mathbf{I}_{L_n})\boldsymbol{\Omega}_n(\mathbf{Q}_n[k+1] \otimes \mathbf{I}_{L_n}) + \lambda \mathbf{I}_{\rho L_n}\right\}^{-1}$
    $\mathbf{L}_n[k+1] = \mathrm{unvec}\,(\mathbf{D}_n[k+1]\,(\mathbf{Q}_n^\top[k+1] \otimes \mathbf{I}_{L_n})\boldsymbol{\Omega}_n \mathrm{vec}(\mathbf{Y}_n))$
    Broadcast $\{\mathbf{Q}_n[k+1]\}$ to neighbors $m \in \mathcal{J}_n$
end for
Return $\mathbf{Q}_n, \mathbf{L}_n$
```

the network size. This is affordable since in practice ρ is typically small for a number of applications of interest (cf. the low-rank assumption). In addition, L_n, the number of row vectors acquired per agent, and T, the number of time instants for data collection, can be controlled by the designer to accommodate a prescribed maximum computational complexity. One can also benefit from the decomposability of (1.9) and (1.10) across rows of $\mathbf{L}$ and $\mathbf{Q}$, respectively, and parallelize the row updates. This way, one only needs to invert $\rho \times \rho$ matrices.

On a per-iteration basis, network agents communicate their updated local estimates $\mathbf{Q}_n[k]$ only with their neighbors, in order to carry out the updates of primal and dual variables during the next iteration. In terms of communication cost, $\mathbf{Q}_n[k]$ is a $T \times \rho$ matrix and its transmission does not incur significant overhead for small values of ρ. Observe that the dual variables $\mathbf{O}_n[k]$ need not be exchanged, and the overall communication cost does not depend on the network size N.

Convergence and optimality. When employed to solve non-convex problems such as (P4), ADMM (or its variant used here) 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, (P4) is bi-convex and gives rise to the strictly convex optimization subproblems (1.9)-(1.11), which admit unique closed-form solutions per iteration. This observation and the linearity of the constraints endow Algorithm 1 with good convergence properties – extensive numerical tests in [7] demonstrate that this is indeed the case. The following proposition proved in [7, Appendix C] asserts that upon convergence, Algorithm 1 attains consensus and global optimality thus yielding the performance of the centralized estimator (P1).

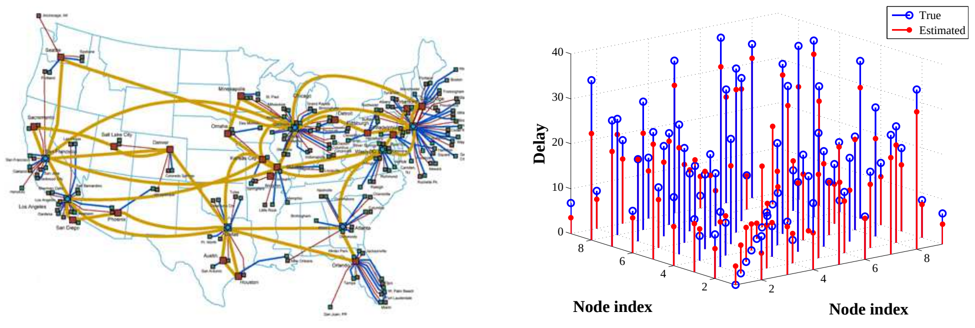

FIGURE 1.1 **Internet2 end-to-end delay prediction.**

(Left) Topology of the Internet-2 backbone network. (Right) Predicted and true end-to-end delays of the Internet2 network when only 20% of the paths are randomly sampled.

Proposition 2. *If the sequence of iterates $\{\mathbf{Q}_n[k], \mathbf{L}_n[k]\}_{n\in\mathcal{N}}$ generated by Algorithm 1 converge to $\{\bar{\mathbf{Q}}_n, \bar{\mathbf{L}}_n\}_{n\in\mathcal{N}}$, and G is connected, then: i) $\bar{\mathbf{Q}}_n = \bar{\mathbf{Q}}_m$ $n, m \in \mathcal{N}$; and ii) if $\|\mathcal{P}_\Omega(\mathbf{Y} - \bar{\mathbf{L}}\bar{\mathbf{Q}}_1^\top)\| \leq \lambda$, then $\hat{\mathbf{X}} = \bar{\mathbf{L}}\bar{\mathbf{Q}}_1^\top$, where $\hat{\mathbf{X}}$ is the global optimum of (P1).*

1.3.1 INTERNET DELAY CARTOGRAPHY

End-to-end network latency information is critical towards enforcing quality-of-service constraints in many Internet applications. However, probing all pairwise delays becomes infeasible in large-scale networks. If one collects the end-to-end latencies of source-sink pairs (i, j) in a delay matrix $\mathbf{X} := [x_{i,j}] \in \mathbb{R}^{N\times N}$, strong dependencies among path delays render $\mathbf{X}$ low-rank [27]. This is mainly because the paths with nearby end nodes often overlap and share common bottleneck links. This property of $\mathbf{X}$ along with the decentralized-processing requirements of large-scale networks, motivates well the adoption of the DMC algorithm for network-wide path latency prediction. Given the n-th row of $\mathbf{X}$ is partially available to agent n, the goal is to impute the missing delays through agent collaboration.

End-to-end flow latencies are collected from the operation of Internet2 backbone network during Aug. 18–22, 2011 [28]. The Internet2 network (Fig. 1.1 (Left)) comprises $N = 9$ agents, $L = 26$ links, and $F = 81$ flows. Spectral analysis of the delay matrix reveals that the first four singular values are markedly dominant, demonstrating that $\mathbf{X}$ is low rank. A fraction of the entries in $\mathbf{X}$ are purposely dropped to yield an incomplete delay matrix $\mathcal{P}_\Omega(\mathbf{X})$. After running the DMC algorithm, the true and predicted latencies are depicted in Fig. 1.1 (Right) (for 20% missing data). The relative prediction error is around 10%.

1.4 STREAMING ANALYTICS

Extracting latent low-dimensional structure from high-dimensional data is of paramount importance in timely inference tasks encountered with big data analytics. However, the collection of massive amounts of data far outweigh the ability of modern computers to store and analyze them in a batch fashion. In addition, in practice (possibly incomplete) observations are acquired sequentially in time which motivates updating previously obtained estimates rather than re-computing new ones from scratch each time a new datum becomes available. In this context, the present section permeates benefits from rank minimization to scalable imputation of missing data, via tracking low-dimensional subspaces and unraveling latent structure from incomplete streaming data.

Subspace tracking has a long history in signal processing. An early noteworthy representative is the projection approximation subspace tracking (PAST) algorithm [29]; see also [30]. Recently, an algorithm (termed GROUSE) for tracking subspaces from incomplete observations was put forth in [31], based on incremental gradient descent iterations on the Grassmannian manifold of subspaces. Recent analysis has shown that GROUSE can converge locally at an expected linear rate [32], and that it is tightly related to the incremental SVD algorithm [33]. PETRELS is a second-order recursive LS type algorithm, that extends the seminal PAST iterations to handle missing data [34]. As noted in [35], the performance of GROUSE is limited by the existence of barriers in the search path on the Grassmanian, which may lead to GROUSE iterations being trapped at local minima; see also [34]. Lack of regularization in PETRELS can also lead to unstable (even divergent) behaviors, especially when the amount of missing data is large. Accordingly, the convergence results for PETRELS are confined to the full-data setting where the algorithm boils down to PAST [34]. Relative to all aforementioned works, the algorithmic framework for online matrix completion presented here offers provable convergence and theoretical performance guarantees in a stationary setting, and is flexible to accommodate tensor streaming data models as well.

Streaming data model. Consider a sequence of high-dimensional data vectors, which are corrupted with additive noise and some of their entries may be missing. At time instant t, the incomplete streaming observations are modeled as

$$\mathcal{P}_{\omega_t}(\mathbf{y}_t) = \mathcal{P}_{\omega_t}(\mathbf{x}_t + \mathbf{v}_t), \quad t = 1, 2, \ldots \tag{1.12}$$

where $\mathbf{x}_t \in \mathbb{R}^L$ is the signal of interest, and $\mathbf{v}_t$ stands for the noise. The set $\omega_t \subset \{1, 2, \ldots, L\}$ contains the indices of available observations, while the corresponding sampling operator $\mathcal{P}_{\omega_t}(\cdot)$ sets the entries of its vector argument not in ω_t to zero, and keeps the rest unchanged; note that $\mathcal{P}_{\omega_t}(\mathbf{y}_t) \in \mathbb{R}^L$. Depending on the application, these acquired vectors could e.g., correspond to (vectorized) images, link traffic measurements collected across physical links of a computer network, or, movie ratings provided by Netflix users. Suppose that the sequence $\{\mathbf{x}_t\}_{t=1}^{\infty}$ lives in a *low-*

dimensional ($\ll L$) linear subspace $\mathcal{L}_t$, which is allowed to change slowly over time. Given the incomplete observations $\{\mathcal{P}_{\omega_\tau}(\mathbf{y}_\tau)\}_{\tau=1}^t$, ensuing sections deal with online (adaptive) estimation of $\mathcal{L}_t$, and reconstruction of $\mathbf{x}_t$ as a byproduct. The reconstruction here involves imputing the missing elements, and denoising the observed ones.

Online matrix completion. Collect the indices of available observations up to time t in the set $\Omega_t := \cup_{\tau=1}^t \omega_\tau$, and the actual batch of observations in the matrix $\mathcal{P}_{\Omega_t}(\mathbf{Y}_t) := [\mathcal{P}_{\omega_1}(\mathbf{y}_1), \ldots, \mathcal{P}_{\omega_t}(\mathbf{y}_t)] \in \mathbb{R}^{L \times t}$. Likewise, introduce matrix $\mathbf{X}_t$ containing the signal of interest. Since $\mathbf{x}_t$ lies in a low-dimensional subspace, $\mathbf{X}_t$ is (approximately) a *low-rank* matrix. A natural estimator leveraging the low rank property of $\mathbf{X}_t$ attempts to fit the incomplete data $\mathcal{P}_{\Omega_t}(\mathbf{Y}_t)$ to $\mathbf{X}_t$ in the LS sense, and minimize the rank of $\mathbf{X}_t$. This motivates recovering $\mathbf{X}_t$ by solving (P1).

Scalable imputation algorithms for streaming observations should effectively overcome the following challenges: (c1) the problem size can easily become quite large, since the number of optimization variables Lt grows with time; (c2) existing batch iterative solvers for (P1) typically rely on costly SVD computations per iteration; see e.g., [14]; and (c3) (columnwise) nonseparability of the nuclear-norm challenges online processing when new columns $\{\mathcal{P}_{\omega_t}(\mathbf{y}_t)\}$ arrive sequentially in time. To limit the computational complexity and memory storage requirements of the algorithm sought, it is henceforth assumed that the dimensionality of the underlying time-varying subspace $\mathcal{L}_t$ is bounded by a known quantity ρ. Accordingly, it is natural to require $\text{rank}(\hat{\mathbf{X}}_t) \leq \rho$. Because $\text{rank}(\hat{\mathbf{X}}_t) \leq \rho$, one can factorize the matrix decision variable as $\mathbf{X} = \mathbf{L}\mathbf{Q}^\top$, where $\mathbf{L}$ and $\mathbf{Q}$ are $L \times \rho$ and $t \times \rho$ matrices, respectively. Such a bilinear decomposition suggests $\mathcal{L}_t$ is spanned by the columns of the tall matrix $\mathbf{L}$, while the rows of $\mathbf{Q}$ are the projections of $\{\mathbf{x}_t\}$ onto $\mathcal{L}_t$.

Leveraging once more the separable nuclear-norm regularization in (1.4), a possible adaptive counterpart to (P1) is the exponentially-weighted LS (EWLS) estimator found by minimizing the empirical cost

$$\text{(P5)} \qquad \min_{\{\mathbf{L},\mathbf{Q}\}} \sum_{\tau=1}^{t} \theta^{t-\tau} \left[\frac{1}{2} \left\| \mathcal{P}_{\omega_\tau}(\mathbf{y}_\tau - \mathbf{L}\mathbf{q}_\tau) \right\|_2^2 + \frac{\bar{\lambda}_t}{2} \|\mathbf{L}\|_F^2 + \frac{\lambda_t}{2} \|\mathbf{q}_\tau\|_2^2 \right]$$

where $\mathbf{Q} := [\mathbf{q}_1, \ldots, \mathbf{q}_t]$, $\bar{\lambda}_t := \lambda_t / \sum_{\tau=1}^t \theta^{t-\tau}$, and $0 < \theta \leq 1$ is the so-termed forgetting factor. When $\theta < 1$, data in the distant past are exponentially downweighted, which facilitates tracking in nonstationary environments. In the case of infinite memory ($\theta = 1$) and for $\lambda_t = \lambda$, the formulation (P5) coincides with the batch estimator (P1). This is the reason for the time-varying factor $\bar{\lambda}_t$ weighting $\|\mathbf{L}\|_F^2$.

Towards deriving a real-time, computationally efficient, and recursive solver of (P5), an alternating-minimization (AM) method is adopted in which iterations coincide with the time-scale t of data acquisition. Per time instant t, a new datum $\{\mathcal{P}_{\omega_t}(\mathbf{y}_t)\}$ is drawn and $\mathbf{q}_t$ is estimated via

$$\mathbf{q}[t] = \arg\min_{\mathbf{q}} \left[\frac{1}{2} \|\mathcal{P}_{\omega_t}(\mathbf{y}_t - \mathbf{L}[t-1]\mathbf{q})\|_2^2 + \frac{\lambda_t}{2} \|\mathbf{q}\|_2^2 \right] \qquad (1.13)$$

Algorithm 2 : Alternating LS for subspace tracking from incomplete observations

```
**input** $\{\mathcal{P}_{\omega_\tau}(\mathbf{y}_\tau), \omega_\tau\}_{\tau=1}^{\infty}$, $\{\lambda_\tau\}_{\tau=1}^{\infty}$, and $\theta$.
**initialize** $\mathbf{G}_l[0] = \mathbf{0}_{\rho\times\rho}$, $\mathbf{s}_l[0] = \mathbf{0}_\rho$, $l = 1, ..., L$, and $\mathbf{L}[0]$ at random.
**for** $t = 1, 2, \ldots$ **do**
    $\mathbf{D}[t] = \left(\lambda_t \mathbf{I}_\rho + \mathbf{L}^\top[t-1]\boldsymbol{\Omega}_t \mathbf{L}[t-1]\right)^{-1} \mathbf{L}^\top[t-1]$.
    $\mathbf{q}[t] = \mathbf{D}[t]\mathcal{P}_{\omega_t}(\mathbf{y}_t)$.
    $\mathbf{G}_l[t] = \theta \mathbf{G}_l[t-1] + \omega_{l,t}\mathbf{q}[t]\mathbf{q}[t]^\top, \quad l = 1, \ldots, L$.
    $\mathbf{s}_l[t] = \theta \mathbf{s}_l[t-1] + \omega_{l,t} y_{l,t}\mathbf{q}[t], \quad l = 1, \ldots, L$.
    $\mathbf{l}_l[t] = \left(\mathbf{G}_l[t] + \lambda_t \mathbf{I}_\rho\right)^{-1} \mathbf{s}_l[t], \quad l = 1, ..., L$.
    **return** $\hat{\mathbf{x}}_t := \mathbf{L}[t]\mathbf{q}[t]$.
**end for**
```

which is an ℓ_2-norm regularized LS (ridge-regression) problem. It admits the closed-form solution

$$\mathbf{q}[t] = \left(\lambda_t \mathbf{I}_\rho + \mathbf{L}^\top[t-1]\boldsymbol{\Omega}_t \mathbf{L}[t-1]\right)^{-1} \mathbf{L}^\top[t-1]\mathcal{P}_{\omega_t}(\mathbf{y}_t) \tag{1.14}$$

where diagonal matrix $\boldsymbol{\Omega}_t \in \{0,1\}^{L\times L}$ is such that $[\boldsymbol{\Omega}_t]_{l,l} = 1$ if $l \in \omega_t$, and is zero elsewhere. In the second step of the AM scheme, the updated subspace matrix $\mathbf{L}[t]$ is obtained by minimizing (P5) with respect to $\mathbf{L}$, while the optimization variables $\{\mathbf{q}_\tau\}_{\tau=1}^{t}$ are fixed and take the values $\{\mathbf{q}[\tau]\}_{\tau=1}^{t}$, namely

$$\mathbf{L}[t] = \arg\min_{\mathbf{L}} \left[\frac{\lambda_t}{2}\|\mathbf{L}\|_F^2 + \sum_{\tau=1}^{t} \theta^{t-\tau}\frac{1}{2}\|\mathcal{P}_{\omega_\tau}(\mathbf{y}_\tau - \mathbf{L}\mathbf{q}[\tau])\|_2^2\right]. \tag{1.15}$$

Notice that (1.15) decouples over the rows of $\mathbf{L}$ which are obtained in parallel via

$$\mathbf{l}_l[t] = \arg\min_{\mathbf{l}} \left[\frac{\lambda_t}{2}\|\mathbf{l}\|_2^2 + \sum_{\tau=1}^{t} \theta^{t-\tau}\omega_{l,\tau}(y_{l,\tau} - \mathbf{l}^\top \mathbf{q}[\tau])^2\right], \tag{1.16}$$

for $l = 1, \ldots, L$, where $\omega_{l,\tau}$ denotes the l-th diagonal entry of $\boldsymbol{\Omega}_\tau$. For $\theta = 1$ and fixed $\lambda_t = \lambda$, $\forall t$, subproblems (1.16) can be efficiently solved via recursive LS (RLS) [36]. Upon defining $\mathbf{s}_l[t] := \sum_{\tau=1}^{t} \theta^{t-\tau}\omega_{l,\tau}y_{l,\tau}\mathbf{q}[\tau]$, $\mathbf{H}_l[t] := \sum_{\tau=1}^{t} \theta^{t-\tau}\omega_{l,\tau}\mathbf{q}[\tau]\mathbf{q}^\top[\tau] + \lambda_t \mathbf{I}_\rho$, and $\mathbf{M}_l[t] := \mathbf{H}_l^{-1}[t]$, one updates

$$\mathbf{s}_l[t] = \mathbf{s}_l[t-1] + \omega_{l,t} y_{l,t}\mathbf{q}[t]$$

$$\mathbf{M}_l[t] = \mathbf{M}_l[t-1] - \omega_{l,t}\frac{\mathbf{M}_l[t-1]\mathbf{q}[t]\mathbf{q}^\top[t]\mathbf{M}_l[t-1]}{1 + \mathbf{q}^\top[t]\mathbf{M}_l[t-1]\mathbf{q}[t]}$$

and forms $\mathbf{l}_l[t] = \mathbf{M}_l[t]\mathbf{s}_l[t]$, for $l = 1, \ldots, L$.

However, for $0 < \theta < 1$ the regularization term $(\lambda_t/2)\|\mathbf{l}\|_2^2$ in (1.16) makes it impossible to express $\mathbf{H}_l[t]$ in terms of $\mathbf{H}_l[t-1]$ plus a rank-one correction. Hence, one cannot resort to the matrix inversion lemma and update $\mathbf{M}_l[t]$ with quadratic complexity only. Based on direct inversion of each $\mathbf{H}_l[t]$, the alternating LS algorithm for subspace tracking from incomplete data is tabulated under Algorithm 2.

Before moving on to reduced-complexity subspace trackers it is worth commenting that the basic idea of performing online rank-minimization leveraging the separable nuclear-norm regularization was first introduced in [6], in the context of unveiling network traffic anomalies. Since then, the approach has gained popularity in real-time non-negative matrix factorization for singing voice separation from its music accompaniment [37], and online robust PCA [38], to name a few examples.

Low-complexity stochastic-gradient subspace updates. To further reduce Algorithm's 2 computational complexity in updating the subspace $\mathbf{L}[t]$, here we develop lightweight algorithms which better suit big data applications. To this end, the basic AM framework is retained, and the update for $\mathbf{q}[t]$ will be identical [cf. (1.14)]. However, instead of exactly solving an unconstrained quadratic program per iteration to obtain $\mathbf{L}[t]$ [cf. (1.15)], the subspace estimates will be obtained via stochastic-gradient descent (SGD) iterations. As shown later on, these updates can be traced to inexact solutions of a certain quadratic program related to (1.15).

For $\theta = 1$, it is shown in Section 1.4.1 that Algorithm 1's subspace estimate $\mathbf{L}[t]$ is obtained by minimizing the empirical cost function $\hat{C}_t(\mathbf{L}) = (1/t)\sum_{\tau=1}^{t} f_\tau(\mathbf{L})$, where

$$f_t(\mathbf{L}) := \frac{1}{2}\|\mathcal{P}_{\omega_t}(\mathbf{y}_t - \mathbf{L}\mathbf{q}[t])\|_2^2 + \frac{\lambda}{2t}\|\mathbf{L}\|_F^2 + \frac{\lambda}{2}\|\mathbf{q}[t]\|_2^2, \quad t = 1, 2, \ldots \tag{1.17}$$

By the law of large numbers, if $\{\mathcal{P}_{\omega_t}(\mathbf{y}_t)\}_{t=1}^{\infty}$ are stationary, solving $\min_{\mathbf{L}} \lim_{t\to\infty} \hat{C}_t(\mathbf{L})$ yields the desired minimizer of the *expected* cost $\mathbb{E}[C_t(\mathbf{L})]$, where the expectation is taken with respect to the unknown probability distribution of the data. A standard approach to achieve this same goal – typically with reduced computational complexity – is to drop the expectation (or the sample averaging operator for that matter), and update the subspace via SGD; see e.g., [36]

$$\mathbf{L}[t] = \mathbf{L}[t-1] - (\mu[t])^{-1}\nabla f_t(\mathbf{L}[t-1]) \tag{1.18}$$

where $(\mu[t])^{-1}$ is the step size, and $\nabla f_t(\mathbf{L}) = -\mathcal{P}_{\omega_t}(\mathbf{y}_t - \mathbf{L}\mathbf{q}[t])\mathbf{q}^\top[t] + (\lambda/t)\mathbf{L}$. The subspace update $\mathbf{L}[t]$ is nothing but the minimizer of a second-order approximation $Q_{\mu[t],t}(\mathbf{L}, \mathbf{L}[t-1])$ of $f_t(\mathbf{L})$ around the previous subspace estimate $\mathbf{L}[t-1]$, where

$$Q_{\mu,t}(\mathbf{L}_1, \mathbf{L}_2) := f_t(\mathbf{L}_2) + \langle \mathbf{L}_1 - \mathbf{L}_2, \nabla f_t(\mathbf{L}_2)\rangle + \frac{\mu}{2}\|\mathbf{L}_1 - \mathbf{L}_2\|_f^2.$$

To tune the step size, the backtracking rule is adopted, whereby the non-increasing step size sequence $\{(\mu[t])^{-1}\}$ decreases geometrically at certain iterations to guarantee the quadratic function $Q_{\mu[t],t}(\mathbf{L}, \mathbf{L}[t-1])$ majorizes $f_t(\mathbf{L})$ at the new update $\mathbf{L}[t]$. Other choices of the step size are discussed in Section 1.4.1. Different from Algorithm 2, no matrix inversions are involved in the update of the subspace $\mathbf{L}[t]$. In the context of adaptive filtering, first-order SGD algorithms such as (1.17) are known to converge slower than RLS. This is expected since RLS can be shown to be an instance of Newton's (second-order) optimization method [36, Ch. 4].

Building on the increasingly popular *accelerated* gradient methods for batch smooth optimization [39, 40], the idea here is to speed-up the learning rate of the estimated subspace (1.18), without paying a penalty in terms of computa-

Algorithm 3 : Online SGD for subspace tracking from incomplete observations

```
input $\{\mathcal{P}_{\omega_\tau}(\mathbf{y}_\tau), \omega_\tau\}_{\tau=1}^{\infty}, \rho, \lambda, \eta > 1$.
initialize $\mathbf{L}[0]$ at random, $\mu[0] > 0$, $\tilde{\mathbf{L}}[1] := \mathbf{L}[0]$, and $k[1] := 1$.
for $t = 1, 2, \dots$ do
    $\mathbf{D}[t] = \left(\lambda\mathbf{I}_\rho + \mathbf{L}^\top[t-1]\mathbf{\Omega}_t\mathbf{L}[t-1]\right)^{-1}\mathbf{L}^\top[t-1]$
    $\mathbf{q}[t] = \mathbf{D}[t]\mathcal{P}_{\omega_t}(\mathbf{y}_t)$
    Find the smallest nonnegative integer $i[t]$ such that with $\bar{\mu} := \eta^{i[t]}\mu[t-1]$
        $f_t(\tilde{\mathbf{L}}[t] - (1/\bar{\mu})\nabla f_t(\tilde{\mathbf{L}}[t])) \leq Q_{\bar{\mu},t}(\tilde{\mathbf{L}}[t] - (1/\bar{\mu})\nabla f_t(\tilde{\mathbf{L}}[t]), \tilde{\mathbf{L}}[t])$
    holds, and set $\mu[t] = \eta^{i[t]}\mu[t-1]$.
    $\mathbf{L}[t] = \tilde{\mathbf{L}}[t] - (1/\mu[t])\nabla f_t(\tilde{\mathbf{L}}[t])$.
    $k[t+1] = \frac{1+\sqrt{1+4k^2[t]}}{2}$.
    $\tilde{\mathbf{L}}[t+1] = \mathbf{L}[t] + \left(\frac{k[t]-1}{k[t+1]}\right)(\mathbf{L}[t] - \mathbf{L}[t-1])$.
end for
return $\hat{\mathbf{x}}[t] := \mathbf{L}[t]\mathbf{q}[t]$.
```

tional complexity per iteration. The critical difference between standard gradient algorithms and the so-termed Nesterov's variant, is that the accelerated updates take the form $\mathbf{L}[t] = \tilde{\mathbf{L}}[t] - (\mu[t])^{-1}\nabla f_t(\tilde{\mathbf{L}}[t])$, which relies on a judicious linear combination $\tilde{\mathbf{L}}[t-1]$ of the previous pair of iterates $\{\mathbf{L}[t-1], \mathbf{L}[t-2]\}$. Specifically, the choice $\tilde{\mathbf{L}}[t] = \mathbf{L}[t-1] + \frac{k[t-1]-1}{k[t]}(\mathbf{L}[t-1] - \mathbf{L}[t-2])$, where $k[t] = \left[1 + \sqrt{4k^2[t-1]+1}\right]/2$, has been shown to significantly accelerate batch gradient algorithms resulting in convergence rate no worse than $O(1/k^2)$; see e.g., [40] and references therein. Using this acceleration technique in conjunction with a backtracking stepsize rule [41], a fast online SGD algorithm for imputing missing entries is tabulated under Algorithm 3. Clearly, a standard (non accelerated) SGD algorithm with backtracking step size rule is subsumed as a special case, when $k[t] = 1$, $t = 1, 2, \dots$. In this case, complexity is $O(|\omega_t|\rho^2)$ mainly due to update of $\mathbf{q}_t$, while the accelerated algorithm incurs an additional cost $O(P\rho)$ for the subspace extrapolation step.

Computational cost. Careful inspection of Algorithm 2 reveals that the main computational burden stems from $\rho \times \rho$ inversions to update the subspace matrix $\mathbf{L}[t]$. The per iteration complexity for performing the inversions is $O(|\omega_t|\rho^3)$ (which could be further reduced if one leverages also the symmetry of $\mathbf{G}_l[t]$), while the cost for the rest of operations is $O(|\omega_t|\rho^2)$. The overall cost of the algorithm per iteration can thus be safely estimated as $O(|\omega_t|\rho^3)$, which can be affordable since ρ is typically small (cf. the low rank assumption). In addition, for the infinite memory case $\theta = 1$ where the RLS update is employed, the overall cost is further reduced to $O(|\omega_t|\rho^2)$. The first-order Algorithm 3 reduces this cost by order ρ, that is in the same order as GROUSE and PETRELS, which incur costs of $O(P\rho + |\omega_t|\rho^2)$ and $O(|\omega_t|\rho^2)$, respectively.

1.4.1 PERFORMANCE GUARANTEES

This section studies the performance of the proposed first- and second-order online algorithms for the infinite memory special case; that is $\theta = 1$. In the sequel, to make the analysis tractable the following assumptions are adopted:
(a1) Processes $\{\omega_t, \mathcal{P}_{\omega_t}(\mathbf{y}_t)\}_{t=1}^{\infty}$ are independent and identically distributed (i.i.d.);
(a2) Sequence $\{\mathcal{P}_{\omega_t}(\mathbf{y}_t)\}_{t=1}^{\infty}$ is uniformly bounded; and
(a3) Iterates $\{\mathbf{L}[t]\}_{t=1}^{\infty}$ lie in a compact set.

To clearly delineate the scope of the analysis, it is worth commenting on (a1)-(a3) and the factors that influence their satisfaction. Regarding (a1), the acquired data is assumed statistically independent across time as it is customary when studying the stability and performance of online (adaptive) algorithms [36]. While independence is required for tractability, (a1) may be grossly violated because the observations $\{\mathcal{P}_{\omega_t}(\mathbf{y}_t)\}$ are correlated across time (cf. the fact that $\{\mathbf{x}_t\}$ lies in a low-dimensional subspace). Still, in accordance with the adaptive filtering folklore e.g., [36], as $\theta \to 1$ or $(\mu[t])^{-1} \to 0$ the upshot of the analysis based on i.i.d. data extends accurately to the pragmatic setting whereby the observations are correlated. Uniform boundedness of $\mathcal{P}_{\omega_t}(\mathbf{y}_t)$ [cf. (a2)] is natural in practice as it is imposed by the data acquisition process. The bounded subspace requirement in (a3) is a technical assumption that simplifies the analysis, and has been corroborated via extensive computer simulations [5].

Convergence of the second-order algorithm. Convergence of the iterates generated by Algorithm 2 (with $\theta = 1$) is established first. Upon defining

$$g_t(\mathbf{L}, \mathbf{q}) := \frac{1}{2}\|\mathcal{P}_{\omega_t}(\mathbf{y}_t - \mathbf{L}\mathbf{q})\|_2^2 + \frac{\lambda_t}{2}\|\mathbf{q}\|_2^2$$

in addition to $\ell_t(\mathbf{L}) := \min_{\mathbf{q}} g_t(\mathbf{L}, \mathbf{q})$, Algorithm 2 aims at minimizing the following *average* cost function at time t

$$C_t(\mathbf{L}) := \frac{1}{t}\sum_{\tau=1}^{t} \ell_\tau(\mathbf{L}) + \frac{\lambda_t}{2t}\|\mathbf{L}\|_F^2. \tag{1.19}$$

Normalization (by t) ensures that the cost function does not grow unbounded as time evolves. For any finite t, (1.19) is essentially identical to the batch estimator in (P3) up to a scaling, which does not affect the value of the minimizer. Note that as time evolves, minimization of C_t becomes increasingly complex computationally. Hence, at time t the subspace estimate $\mathbf{L}[t]$ is obtained by minimizing the *approximate* cost function

$$\hat{C}_t(\mathbf{L}) = \frac{1}{t}\sum_{\tau=1}^{t} g_\tau(\mathbf{L}, \mathbf{q}[\tau]) + \frac{\lambda_t}{2t}\|\mathbf{L}\|_F^2 \tag{1.20}$$

in which $\mathbf{q}[t]$ is obtained based on the prior subspace estimate $\mathbf{L}[t-1]$ after solving $\mathbf{q}[t] = \arg\min_{\mathbf{q}} g_t(\mathbf{L}[t-1], \mathbf{q})$ [cf. (1.13)]. Obtaining $\mathbf{q}[t]$ this way resembles the projection approximation adopted in [29]. Since $\hat{C}_t(\mathbf{L})$ is a smooth convex quadratic function, the minimizer $\mathbf{L}[t] = \arg\min_{\mathbf{L}} \hat{C}_t(\mathbf{L})$ is the solution of the linear equation $\nabla \hat{C}_t(\mathbf{L}[t]) = \mathbf{0}_{L\times\rho}$.

So far, it is apparent that since $g_t(\mathbf{L}, \mathbf{q}[t]) \geq \min_{\mathbf{q}} g_t(\mathbf{L}, \mathbf{q}) = \ell_t(\mathbf{L})$, the approximate cost function $\hat{C}_t(\mathbf{L}[t])$ overestimates the target cost $C_t(\mathbf{L}[t])$, for $t = 1, 2, \ldots$. However, it is not clear whether the subspace iterates $\{\mathbf{L}[t]\}_{t=1}^{\infty}$ converge, and most importantly, how well can they optimize the target cost function C_t. The good news is that $\hat{C}_t(\mathbf{L}[t])$ asymptotically approaches $C_t(\mathbf{L}[t])$, and the subspace iterates null $\nabla C_t(\mathbf{L}[t])$ as well, both as $t \to \infty$. This result is summarized in the next proposition.

Proposition 3. *Under (a1)–(a3) and $\theta = 1$ in Algorithm 2, if $\lambda_t = \lambda$ and $\lambda_{\min}[\nabla^2 \hat{C}_t(\mathbf{L})] \geq c$ for some $c > 0$, then $\lim_{t\to\infty} \nabla C_t(\mathbf{L}[t]) = \mathbf{0}_{L\times\rho}$ almost surely (a.s.), i.e., the subspace iterates $\{\mathbf{L}[t]\}_{t=1}^{\infty}$ asymptotically fall into the stationary point set of the batch problem (P3).*

It is worth noting that the pattern and the amount of missing data, summarized in the sampling sets $\{\omega_t\}$, play a key role towards satisfying the Hessian's positive semi-definiteness condition. In fact, random misses are desirable since the Hessian $\nabla^2 \hat{C}_t(\mathbf{L}) = \frac{\lambda}{t}\mathbf{I}_{L\rho} + \frac{1}{t}\sum_{\tau=1}^{t}(\mathbf{q}[\tau]\mathbf{q}^\top[\tau]) \otimes \mathbf{\Omega}_\tau$ is more likely to satisfy $\nabla^2 \hat{C}_t(\mathbf{L}) \geq c\mathbf{I}_{L\rho}$, for some $c > 0$.

The proof of Proposition 3 is inspired by [42] which establishes convergence of an online dictionary learning algorithm using the theory of martingale sequences. Details can be found in [5], and in a nutshell the proof procedure proceeds in the following two main steps:

(S1) Establish that the approximate cost sequence $\{\hat{C}_t(\mathbf{L}[t])\}$ asymptotically converges to the target cost sequence $\{C_t(\mathbf{L}[t])\}$. To this end, it is first proved that $\{\hat{C}_t(\mathbf{L}[t])\}_{t=1}^{\infty}$ is a quasi-martingale sequence, and hence convergent a.s. This relies on the fact that $g_t(\mathbf{L}, \mathbf{q}[t])$ is a *tight* upper bound approximation of $\ell_t(\mathbf{L})$ at the previous update $\mathbf{L}[t-1]$, namely, $g_t(\mathbf{L}, \mathbf{q}[t]) \geq \ell_t(\mathbf{L})$, $\forall \mathbf{L} \in \mathbb{R}^{L\times\rho}$, and $g_t(\mathbf{L}[t-1], \mathbf{q}[t]) = \ell_t(\mathbf{L}[t-1])$.

(S2) Under certain regularity assumptions on g_t, establish that convergence of the cost sequence $\{\hat{C}_t(\mathbf{L}[t]) - C_t(\mathbf{L}[t])\} \to 0$ yields convergence of the gradients $\{\nabla \hat{C}_t(\mathbf{L}[t]) - \nabla C_t(\mathbf{L}[t])\} \to 0$, which subsequently results in $\lim_{t\to\infty} \nabla C_t(\mathbf{L}[t]) = \mathbf{0}$.

Optimality. Beyond convergence to stationary points of (P3), one may ponder whether the online estimator offers performance guarantees of the batch nuclear-norm regularized estimator (P1), for which stable/exact recovery results are well documented e.g., in [14, 9]. Specifically, given the learned subspace $\bar{\mathbf{L}}[t]$ and the corresponding $\bar{\mathbf{Q}}[t]$ [obtained via (1.13)] over a time window of size t, is $\{\hat{\mathbf{X}}[t] := \bar{\mathbf{L}}[t]\bar{\mathbf{Q}}^\top[t]\}$ an optimal solution of (P1) as $t \to \infty$? This in turn requires asymptotic analysis of the optimality conditions for (P1) and (P3), and a positive answer is established in the next proposition whose proof is available in [5].

Proposition 4. *Consider the subspace iterates $\{\mathbf{L}[t]\}$ generated by either Algorithm 2* *(with $\theta = 1$), or Algorithm 3. If there exists a subsequence $\{\mathbf{L}[t_k], \mathbf{Q}[t_k]\}$ for which (c1) $\lim_{k\to\infty} \nabla C_{t_k}(\mathbf{L}[t_k]) = \mathbf{0}_{L\times\rho}$ a.s., and (c2) $\frac{1}{\sqrt{t_k}}\sigma_{\max}[\mathcal{P}_{\Omega_{t_k}}(\mathbf{Y}_{t_k} - \mathbf{L}[t_k]\mathbf{Q}^\top[t_k])] \leq \frac{\lambda_{t_k}}{\sqrt{t_k}}$ hold, then the sequence $\{\mathbf{X}[k] = \mathbf{L}[t_k]\mathbf{Q}^\top[t_k]\}$ satisfies the optimality conditions for (P1) [normalized by t_k] as $k \to \infty$ a.s.*

Regarding condition (c1), even though it holds for a time invariant rank-controlling parameter λ as per Proposition 3, numerical tests indicate that it still holds true for the time-varying case; see [5, Remark 2] for guidelines on the choice of λ_t. Under (a2) and (a3) one has $\sigma_{\max}[\mathcal{P}_{\Omega_t}(\mathbf{Y}_t - \mathbf{L}[t]\mathbf{Q}^\top[t])] \approx O(\sqrt{t})$, which implies that the quantity on the left-hand side of (c2) cannot grow unbounded. Moreover, upon choosing $\lambda_t \approx O(\sqrt{t})$ the term in the right-hand side of (c2) will not vanish, which suggests that the qualification condition can indeed be satisfied [5]. Effective heuristic rules are devised in [9, 5] for tunning λ.

1.4.2 REAL-TIME NETWORK TRAFFIC MONITORING

Accurate estimation of OD flow traffic in the backbone of large-scale IP networks is of paramount importance for proactive network security and management tasks [43]. Several experimental studies have demonstrated that OD flow traffic exhibits low rank, mainly due to common temporal patterns across OD flows, and periodic trends across time [2]. However, due to the massive number of OD pairs and the high volume of traffic, measuring the traffic of all possible OD flows is impossible for all practical purposes [2, 43]. Only the traffic level for a small fraction of OD flows can be measured via the NetFlow protocol [2].

Aggregate OD-flow traffic is collected from operation of the Internet2 during December $8 - 28$, 2003 containing 121 OD pairs [28]. The measured OD flows contain spikes (anomalies), which are discarded to end up with a anomaly-free data stream $\{\mathbf{y}_t\} \in \mathbb{R}^{121}$. The detailed description of the considered dataset can be found in [6]. A fraction π of the entries of $\mathbf{y}_t$ are then randomly sampled to yield the input of Algorithm 2. Evolution of the running-average traffic estimation error is depicted in Fig. 1.2(Left) for different subspace trackers and π values. Evidently, Algorithm 2 outperforms the competing alternatives when λ_t is adaptively tuned as in [5]. When only 25% of the total OD flows are sampled by NetFlow, Fig. 1.2(Right) depicts how Algorithm 2 accurately tracks representative OD flows.

1.4.3 LARGE-SCALE MACHINE LEARNING

The advocated subspace learning framework identifies latent low-dimensional structure in streaming data, and can facilitate large-scale machine learning tasks beyond matrix completion. The scope could be for instance broadened to accommodate large-scale dimensionality reduction and feature extraction from multi-way tensors [44], as well as categorical and finite-alphabet datasets [45]. In addition, the developed subspace trackers can be adopted in conjunction with support vector ma-

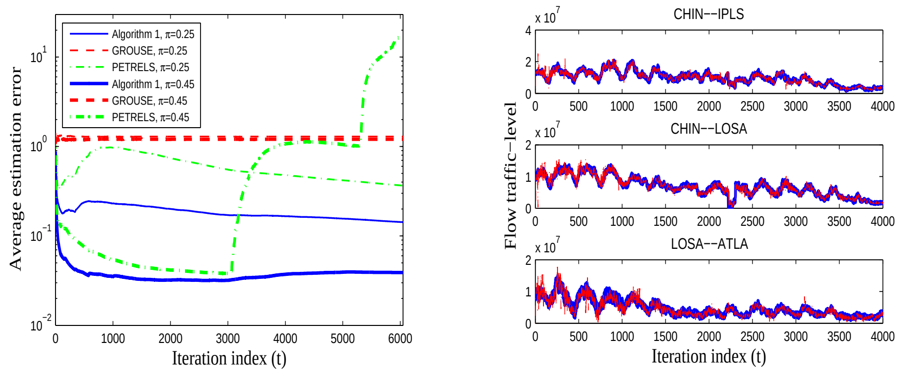

FIGURE 1.2 **Internet2 flow traffic estimation.**

Traffic estimation performance for Internet2 data when $\rho = 10$ and $\theta = 0.95$, and a variable fraction π of data is available. (Left) Average estimation error for $\pi = 0.25$ (thin line) and $\pi = 0.45$ (thick line). (Right) Algorithm 2's estimated (dashed red) versus true (solid blue) OD flow traffic for 75% missing data ($\pi = 0.25$).

chines (SVMs) to classify incomplete data online [46]. These generalizations are briefly summarized next.

Multi-linear decomposition and dimensionality reduction. Many applications involve data indexed by three, or, more variables giving rise to a tensor, instead of just two variables as in the matrix settings dealt with so far. It is not uncommon that one of these variables indexes time [47, 5], and that sizable portions of the data are missing [48, 49]. Examples of time-indexed, incomplete tensor data include: (i) dynamic social networks represented through a temporal sequence of adjacency matrices, while it may be the case that not all pairwise interactions among nodes can be sampled; and (ii) multidimensional nuclear magnetic resonance (NMR) analysis, where missing data are encountered when sparse sampling is used in order to reduce the experimental time. Various data analytics tasks aim at unveiling underlying latent structures, which calls for high-order tensor factorizations even in the presence of missing data [49, 48]. With this objective in mind, [5] puts forth for the first time an online (adaptive) algorithm for decomposing low-rank tensors with missing entries; see also [44] for an adaptive algorithm to obtain parallel factor analysis (PARAFAC) decompositions – a natural extension of the bilinear model in (P5) to multilinear case. The proposed online algorithm offers a viable approach to solving large-scale tensor decomposition (and completion) problems, even if the data is not actually streamed but they are so massive that do not fit in main memory

Sketching of categorical data. With the scale of data growing every day, reducing the dimensionality (a.k.a. sketching) of high-dimensional data has emerged as a task of paramount importance. Critical challenges arise with increasingly ubiquitous datasets comprising incomplete categorical samples. For instance, in movie recommender systems observation $\mathbf{y}_t$ represents the users' categorical ratings (e.g., like/dislike or in a 1 to 5 integer-valued scale) for the t-th movie. Since each user rates only a small fraction of movies, ratings for a sizable portion of movies will be missing. In this context, effective sketching tools were developed in [45] for large-scale categorical data that are incomplete and streaming. Low-dimensional Probit, Tobit and Logit models were considered and learned, using a maximum likelihood approach regularized with a separable surrogate of the nuclear norm as in (1.4). The developed online algorithms are provably convergent and light-weight, while they achieve sublinear regret bounds for finite data streams and asymptotic convergence for infinite data streams.

Classification with absent features. The SVM is a workhorse classification technique that breaks down when some of the features in the input vectors are missing. Consider streaming, high-dimensional data $\mathbf{x}_t$ from two classes, namely C_1 and C_2, and suppose only a small fraction of features is present due to security concerns, or, outliers that render data unreliable. Building on the subspace learning framework discussed in this section, a joint imputation and supervised classification scheme is developed in [46] which operates in two alternating steps upon arrival of a new datum: (i) the algorithm first imputes the missing features based on the learned low-dimensional subspace; and (ii) subsequently adjusts the SVM hyperplane to match the imputed datum to its binary label.

1.5 CONCLUDING SUMMARY

Nowadays machine learning tasks deal with *sheer volumes* of data of possibly *incomplete*, *decentralized*, and *streaming* nature that demands on-the-fly processing for real-time decision making. Conventional inference analytics mine such big data by leveraging their intrinsic parsimony, e.g., via models that include rank and sparsity regularization or priors. Convex nuclear and ℓ_1-norm surrogates are typically adopted and offer well-documented guarantees in recovering informative low-dimensional structure from high-dimensional data. However, the computational complexity of the resulting algorithms tends to scale poorly due to the nuclear norm's entangled structure, which also impedes streaming and decentralized analytics. To mitigate this computational hurdle, this chapter discussed a framework which leverages a bilinear characterization of the nuclear norm to bring separability at the expense of nonconvexity. This challenge notwithstanding, under mild conditions stationary points of the nonconvex program provably coincide with the optimum of the convex counterpart. Using this idea along with the theory of alternating minimization, lightweight algorithms are developed with low communication-overhead for in-network processing. Provably convergent online subspace trackers which are suitable for streaming analytics are developed as well. Remarkably, even under the constraints imposed by decentralized computing and sequential data acquisition, one can still attain the performance offered by the prohibitively-complex batch analytics. While the ideas were presented for a matrix completion problem, the scope of the presented framework can be broadened to be jointly adopted with downstream machine learning tasks such as dimensionality reduction, clustering, classification, multidimensional scaling, and anomaly detection.

ACKNOWLEDGMENTS

The work was supported in part by NSF grants NSF 1500713, 1509040, 1514056, 1711471, and ARO W911NF-15-1-0492.

REFERENCES

1. Slavakis K, Giannakis GB, Mateos G, Modeling and optimization for Big Data analytics. IEEE Signal Processing Magazine 2014; 31(5):18–31.
2. Lakhina A, Papagiannaki K, Crovella M, Diot C, Kolaczyk ED, Taft N, Structural analysis of network traffic flows. In: Proc. of ACM SIGMETRICS, New York, NY, 2004.
3. Mateos G, Rajawat K, Dynamic network cartography: Advances in network health monitoring. IEEE Signal Processing Magazine 2013; 30(3):129–143.
4. Mardani M, Mateos G, Giannakis GB, Recovery of low-rank plus compressed sparse matrices with application to unveiling traffic anomalies. IEEE Transactions on Information Theory 2013; 59:5186–5205.
5. Mardani M, Mateos G, Giannakis GB, Subspace learning and imputation for streaming big data matrices and tensors. IEEE Transactions on Signal Processing 2015; 63:2663 – 2677.
6. Mardani M, Mateos G, Giannakis GB, Dynamic anomalography: Tracking network anomalies via sparsity and low rank. IEEE Journal on Selected Topics Signal Processing 2013; 7:50–66.
7. Mardani M, Mateos G, Giannakis GB, Decentralized sparsity regularized rank minimization: Applications and algorithms. IEEE Transactions on Signal Processing 2013; 61:5374–5388.
8. Wright J, Ganesh A, Min K, Ma Y, Compressive principal component pursuit. In: Proceeding of International Symposium on Information Theory, Cambridge, MA, 2012, pp. 1276–1280.
9. Candès EJ, Plan Y, Matrix completion with noise. Proceedings of the IEEE 2009; 98:925–936.
10. Chistov A, Grigorev D, Complexity of quantifier elimination in the theory of algebraically closed fields. In: Mathematical Foundations of Computer Science, Lecture Notes in Computer Science, vol. 176, Lecture Notes in Computer Science, vol. 176, Springer Berlin / Heidelberg, 1984; pp. 17–31.
11. Natarajan BK, Sparse approximate solutions to linear systems. SIAM Journal on Computing 1995; 24:227–234.
12. Fazel M, Hindi H, Boyd SP, A rank minimization heuristic with application to minimum order system approximation. In: Proc. of American Control Conference, vol. 6, vol. 6, 2001, pp. 4734–4739.
13. Candes EJ, Tao T, Decoding by linear programming. IEEE Transactions on Information Theory 2005; 51(12):4203–4215.

14. Candes EJ, Recht B, Exact matrix completion via convex optimization. Foundation of Computational Mathematics 2009; 9(6):717–722.
15. Candes EJ, Li X, Ma Y, Wright J, Robust principal component analysis? Journal of the ACM 2011; 58(1):1–37.
16. Chandrasekaran V, Sanghavi S, Parrilo PR, Willsky AS, Rank-sparsity incoherence for matrix decomposition. SIAM Journal on Optimization 2011; 21(2):572–596.
17. Mardani M, Mateos G, Giannakis GB, Exact recovery of low-rank plus compressed sparse matrices. In: Statistical Signal Processing Workshop (SSP), 2012 IEEE, IEEE, 2012, pp. 49–52.
18. Recht B, Fazel M, Parrilo PA, Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM Review 2010; 52(3):471–501.
19. Srebro N, Shraibman A, Rank, trace-norm and max-norm. In: Proc. of Learning Theory, 2005, pp. 545–560.
20. Bertsekas DP, Tsitsiklis JN, Parallel and Distributed Computation: Numerical Methods. 2nd ed., Athena-Scientific, 1999.
21. S Boyd N Parikh ECBP, Eckstein J, Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundation Trends in Machine Learning 2010; 3:1–122.
22. Giannakis GB, Ling Q, Mateos G, Schizas ID, Zhu H, Decentralized learning for wireless communications and networking. In: Glowinski R, Osher S, Yin W, editors, Splitting Methods in Communication and Imaging, Science and Engineering, Scientific Computation, Springer New York, 2016; pp. 461–497.
23. Recht B, Ré C, Parallel stochastic gradient algorithms for large-scale matrix completion. Mathematical Programming Computation 2013; 5(2):201–226.
24. Lakhina A, Crovella M, Diot C, Diagnosing network-wide traffic anomalies. In: Proceeding of ACM SIGCOMM, Portland, OR, 2004.
25. Schizas ID, Giannakis GB, Luo ZQ, Distributed estimation using reduced-dimensionality sensor observations. IEEE Transactions on Signal Processing 2007; 55:4284–4299.
26. Mateos G, Bazerque JA, Giannakis GB, Distributed sparse linear regression. IEEE Transactions on Signal Processing 2010 (submitted); .
27. Liao Y, Du W, Geurts P, Leduc G, DMFSGD: A decentralized matrix factorization algorithm for network distance prediction. IEEE/ACM Trans Network 2011; See also arXiv:1201.1174v1 [cs.NI].
28. The Internet Observatory Data Collection 2012; URL: `http://internet2.edu/observatory/archive/data-collections.html`.
29. Yang B, Projection approximation subspace tracking. IEEE Transactions on Signal processing 1995; 43(1):95–107.
30. Yang JF, Kaveh M, Adaptive eigensubspace algorithms for direction or frequency estimation and tracking. IEEE Transactions on Acoustic, Speech, Signal Processing 1988; 36(2):241–251.
31. Balzano L, Nowak R, Recht B, Online identification and tracking of subspaces from highly incomplete information. In: Proc. of Allerton Conference on Communication, Control, and Computing, Monticello, USA, 2010.
32. Balzano L, Wright SJ, Local convergence of an algorithm for subspace identification from partial data. Foundations of Computational Mathematics 2015; 15(5):1279–1314.
33. Balzano L, Wright SJ, On GROUSE and incremental SVD. In: IEEE 5th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), IEEE, 2013, pp. 1–4.

34. Chi Y, Eldar YC, Calderbank R, Petrels: Parallel subspace estimation and tracking by recursive least squares from partial observations. IEEE Transactions on Signal Processing 2013; 61(23):5947–5959.
35. Dai W, Milenkovic O, Kerman E, Subspace evolution and transfer (SET) for low-rank matrix completion. IEEE Transactions on Signal Processing 2011; 59(7):3120–3132.
36. Solo V, Kong X, Adaptive Signal Processing Algorithms: Stability and Performance. Prentice Hall, 1995.
37. Sprechmann P, Bronstein AM, Sapiro G, Real-time online singing voice separation from monaural recordings using robust low-rank modeling. In: Proc. Annual Conference of the International Society for Music Information Retrieval, Porto, Portugal, 2012.
38. Feng J, Xu H, Yan S, Online robust PCA via stochastic optimization. In: Proc. Advances in Neural Information Processing Systems, Lake Tahoe, NV, 2013.
39. Nesterov Y, A method of solving a convex programming problem with convergence rate $o(1/k^2)$. Soviet Mathematics Doklady 1983; 27:372–376.
40. Beck A, Teboulle M, A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal on Imaging Sciences 2009; 2:183–202.
41. Bertsekas DP, Nonlinear Programming. 2nd ed., Athena-Scientific, 1999.
42. J Mairal J Bach JP, Sapiro G, Online learning for matrix factorization and sparse coding. Journal of Machine Learning Research 2010; 11:19–60.
43. Kolaczyk ED, Statistical Analysis of Network Data: Methods and Models. Springer, 2009.
44. Mardani M, Giannakis GB, Ugurbil K, Tracking tensor subspaces with informative random sampling for real-time MR imaging. arXiv preprint arXiv:160904104 2016; .
45. Shen Y, Mardani M, Giannakis GB, Online categorical subspace learning for sketching big data with misses. IEEE Transactions on Signal Processing ; 65(15):4004–4018.
46. Sheikholeslami F, Mardani M, Giannakis GB, Streaming support vector classification of big data with misses. In: Proc. of Asilomar Conf. on Control, Signal and Systems, 2014, pp. 516–520.
47. Nion D, Sidiropoulos ND, Adaptive algorithms to track the PARAFAC decomposition of a third-order tensor. IEEE Transactions on Signal Processing 2009; 57(6):2299–2310.
48. Acar E, Dunlavy DM, Kolda TG, Mrup M, Scalable tensor factorizations for incomplete data. Chemometrics and Intelligent Laboratory Systems 2011; 106(1):41–56.
49. Bazerque JA, Mateos G, Giannakis GB, Rank regularization and Bayesian inference for tensor completion and extrapolation. IEEE Transactions Signal Processing 2013; 61(22):5689–5703.