Pattern Recognition 97 (2020) 106992

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

L_1-Subspace Tracking for Streaming Data

Ying Liu[a,*], Konstantinos Tountas[b], Dimitris A. Pados[b], Stella N. Batalama[b], Michael J. Medley[c]

[a] Department of Computer Science and Engineering, Santa Clara University, Santa Clara, CA 95053, United States
[b] Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431, United States
[c] Department of Engineering, The State University of New York Polytechnic Institute, Utica, NY 13502, United States

ARTICLE INFO

Article history:
Received 9 January 2019
Revised 23 July 2019
Accepted 31 July 2019
Available online 5 August 2019

Keywords:
Dimensionality reduction
Eigenvector decomposition
Internet-of-Things
L_1-norm
Outliers
Principal-component analysis
Subspace learning

ABSTRACT

High-dimensional data usually exhibit intrinsic low-rank structures. With tremendous amount of streaming data generated by ubiquitous sensors in the world of Internet-of-Things, fast detection of such low-rank pattern is of utmost importance to a wide range of applications. In this work, we present an L_1-subspace tracking method to capture the low-rank structure of streaming data. The method is based on the L_1-norm principal-component analysis (L_1-PCA) theory that offers outlier resistance in subspace calculation. The proposed method updates the L_1-subspace as new data are acquired by sensors. In each time slot, the conformity of each datum is measured by the L_1-subspace calculated in the previous time slot and used to weigh the datum. Iterative weighted L_1-PCA is then executed through a refining function. The superiority of the proposed L_1-subspace tracking method compared to existing approaches is demonstrated through experimental studies in various application fields.

1. Introduction

Principal-component analysis (PCA) is a prevalent method for dimensionality reduction and subspace learning. Conventional L_2-norm-based principal-component analysis (L_2-PCA), however, is easily affected by "outlier" values that are numerically distant from the nominal low-rank signal. To deal with the problem of outliers in subspace approximation, there are extensive studies in robust PCA methods. In the pioneer works [1–4] subspace learning is performed under an L_1-error minimization criterion, or its variants. The robust PCA (RPCA), a.k.a. principal-component pursuit (PCP) developed in [5] performs low-rank and sparse decomposition by minimizing a weighted sum of the nuclear-norm of the low-rank component and the L_1-norm of the sparse component. More recently, the robust PCA idea is adopted in DECOLOR [6], which in addition uses Markov random-field (MRF) modeling to improve the accuracy of detecting contiguous outliers. The method in [7] recast the L_1-error minimization problem into a weighted L_2-error minimization problem. By properly choosing the weights of data samples, the formulated L_2-error minimization problem is equivalent to the robust L_1-error minimization problem, and can be solved efficiently via singular-value decomposition (SVD). The method in [8] learns the low-rank representation for data by utilizing locality and similarity information among data, using graph-based manifold analysis. In [9] a fast, iterative algorithm is proposed for outlier resistant two-dimensional PCA based on the Frobenius-norm with respect to the spatial (attribute) dimensions and the 1-norm for the summation over different data points. Authors in [10] propose to jointly select useful features and enhance the robustness of PCA by the relaxation of the orthogonality constraints on the transform matrix.

Another line of research performs robust subspace learning by maximizing the L_1-norm of the data projected onto the pursued subspace [11–15]. The pursued principal components are called L_1 principal components. The work in [11] presented a suboptimal iterative algorithm for the computation of one L_1 principal component and [12] presented an iterative algorithm for suboptimal joint computation of $d \geq 1$ L_1 principal components. In [13], for the first time in the literature, algorithms for exact calculation of L_1 principal components are developed. Later, suboptimal algorithms were developed in [14] and [15] for fast computation of the L_1 principal components. This L_1-PCA method has been successfully applied to a wide range of research fields such as direction-of-arrival (DoA) estimation [16] and robust face recognition [17]. Besides, compressed-sensed-domain L_1-PCA methods were developed for low-rank background scene and sparse foreground moving objects extraction from compressed-sensed surveillance video sequences [18]. In [19], a reweighted L_1-PCA algorithm (L_1-IRW)

* Corresponding author.
E-mail address: yliu15@scu.edu (Y. Liu).

https://doi.org/10.1016/j.patcog.2019.106992
0031-3203/

was developed to refine the pursued L_1-subspace in an iterative manner.

Nevertheless, existing L_1-PCA methods in [11–19] are batch algorithms designed for fixed data ensemble. As sensors keep acquiring streaming data, it is essential to update the calculated subspace with new information and track the potential gradual change of data's low-rank pattern. Robust subspace tracking has been developed to tackle this problem. A computationally efficient incremental PCA algorithm is developed in [20] for adaptive background modeling and active object recognition. The authors in [21] propose an incremental non-negative matrix factorization scheme for online processing of large data sets. The scheme incrementally updates its factors by appropriately reflecting the influence of each data sample on the factorization. In [22], an online robust PCA method is proposed that alternates between standard L_2-PCA for updating PCs and probabilistic selection of the new samples which alleviates the impact of outliers. The online robust PC (OR-PCA) method proposed in [23] reformulates the objective function of PCP [5] by decomposing the nuclear norm into an explicit product of two low-rank matrices, which can be solved by a stochastic optimization algorithm. The method in [24] adopted similar approach to that in [23], and applied it to track the low-rank structure of traffic flow volume in backbone networks. An adaptive projected subgradient method based algorithm is proposed in [25], introducing a cost function properly calculated for each time instance and searching for the set of points which score zero loss. Outlier detection and correction of corrupted data is employed to purify data. The Grassmann Averages method proposed in [26] formulates subspace estimation as the computation of the average of subspaces spanned by data samples, which is scalable to large datasets and robust to outliers. The Grassmannian robust adaptive subspace tracking algorithm (GRASTA) [27] is an efficient and robust online algorithm for tracking subspaces from highly incomplete information. It uses a robust ℓ_1-norm cost function to impose sparsity on the outliers which is formulated as an augmented Lagrangian function, then the subspace and the outlier are estimated by the alternating direction method of multiplier (ADMM). The method in [28] adopts a similar approach to [27], and it also maintains the proximity of the updated subspace to the previous subspace estimate. The recursive projected compressive sensing (ReProCS) algorithm was developed in [29], [30] which performs online robust subspace estimation, and was further extended to a practically usable version (Prac-ReProCS) [31]. This series of studies addressed the problem of recursively recovering sparse and correlated signals in the presence of low-rank and correlated noise. The methods were successfully applied to the scenario of separating a slowly changing video background from correlated moving foreground objects/regions. In [32], an online mixture of Gaussians (MoG) low-rank matrix factorization method (OMoGMF) is proposed for robust video background subtraction. It modeled the foreground as a MoG and the model is also regularized by the learned foreground/background in previous frames. The model can be formulated as a concise probabilistic maximum a posteriori probability (MAP) model, and can be readily solved by the expectation-maximization (EM) algorithm. In [33], a union of subspaces tracking algorithm is proposed for online anomaly detection. The observed data samples are assumed to have a Gaussian mixture model whose covariance matrices each are dominated by a low-rank component. The online discriminative multi-task tracker [34] is proposed with structured and weighted low rank regularization.

In this paper, we propose an L_1-subspace tracking method. As new data are successively acquired over time, the procedure updates the L_1-subspace to capture the underlying low-rank data structure. In each time slot, nominal compliance of each sample in the current processing window is inferred by its relative distance to the L_1-subspace calculated in the previous time slot and translated to a "weight". Samples with larger weights tend to be nominal samples and samples with smaller weights are more likely to be the outliers. Iterative weighted L_1-PCA is then carried out via a refining function. The function alternatively updates the bits associated with the pursued L_1-subspace and the sample weights. Upon convergence, the function returns a refined L_1-subspace for the current time slot. The whole procedure has the merits of outlier suppression through sample weighting and processing acceleration through a warm-start bit-flipping technique.

The remainder of this paper is organized as follows. In Section 2, we introduce necessary background on L_1-PCA. In Section 3, the proposed L_1-subspace tracking algorithm is developed. In Section 4, the effectiveness of the proposed algorithm is demonstrated through four experiments: (i) synthetic data example, (ii) moving objects detection from streaming surveillance videos, (iii) robust online cooperative spectrum sensing in a cognitive radio network, and (iv) DoA tracking in wireless communications. Computational complexity is analyzed in Section 5. Finally, we draw conclusions and discuss future work in Section 6.

2. Background of L_1 principal-component analysis

2.1. L_1-PCA and its solvers

Consider N real-valued samples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ of dimension D ($N < D$) that form the $D \times N$ data matrix

$$\mathbf{X} = [\mathbf{x}_1 \;\; \mathbf{x}_2 \;\; \ldots \mathbf{x}_N]. \tag{1}$$

In conventional L_2-PCA, one seeks to describe (approximate) data matrix $\mathbf{X}$ by a rank-r product $\mathbf{PQ}^{\mathrm{T}}$ where $\mathbf{P} \in \mathbb{R}^{D\times r}$, $\mathbf{Q} \in \mathbb{R}^{N\times r}$, $r \leq N$. Given data matrix $\mathbf{X}$, L_2-PCA minimizes the sum of the element-wise squared error between the original matrix $\mathbf{X}$ and its rank-r representation $\mathbf{PQ}^T$ in the form of Problem $\mathcal{P}_1^{L_2}$ defined below,

$$\mathcal{P}_1^{L_2} : (\mathbf{P}_{L_2}, \mathbf{Q}_{L_2}) = \arg \min_{\substack{\mathbf{P} \in \mathbb{R}^{D\times r}, \mathbf{P}^T\mathbf{P} = \mathbf{I}_r \\ \mathbf{Q} \in \mathbb{R}^{N\times r}}} \|\mathbf{X} - \mathbf{PQ}^{\mathrm{T}}\|_2, \tag{2}$$

where $\mathbf{I}_r$ is an $r \times r$ identity matrix, and matrix $\mathbf{P}$ has r orthonormal columns. Problem $\mathcal{P}_1^{L_2}$ is equivalent to the following two problems,

$$\mathcal{P}_2^{L_2} : \mathbf{P}_{L_2} = \arg \min_{\substack{\mathbf{P} \in \mathbb{R}^{D\times r} \\ \mathbf{P}^T\mathbf{P} = \mathbf{I}_r}} \|\mathbf{X} - \mathbf{PP}^{\mathrm{T}}\mathbf{X}\|_2, \tag{3}$$

and

$$\mathcal{P}_3^{L_2} : \mathbf{P}_{L_2} = \arg \max_{\substack{\mathbf{P} \in \mathbb{R}^{D\times r} \\ \mathbf{P}^T\mathbf{P} = \mathbf{I}_r}} \|\mathbf{X}^{\mathrm{T}}\mathbf{P}\|_2, \tag{4}$$

for which the solution is given by the r dominant left singular vectors of the original data matrix $\mathbf{X}$.

Nevertheless, by minimizing the sum of squared errors, L_2 principal component calculation becomes sensitive to extreme error value occurrences caused by the presence of outlying samples in the data matrix (samples that are numerically distant from the nominal data, appear only few times in the data matrix, and are not to appear under normal system operation upon design). Motivated by this observed drawback of L_2 subspace signal processing, subspace-decomposition based on L_1-norm maximization was proposed for robustness. Replacing the L_2-norm in $\mathcal{P}_3^{L_2}$ by L_1-norm, the so-called L_1-PCA calculates principal components in the form of

$$\mathcal{P}^{L_1} : \mathbf{P}_{L_1} = \arg \max_{\substack{\mathbf{P} \in \mathbb{R}^{D\times r} \\ \mathbf{P}^T\mathbf{P} = \mathbf{I}_r}} \|\mathbf{X}^{\mathrm{T}}\mathbf{P}\|_1. \tag{5}$$

Since the L_1-norm metric is less likely to exaggerate the contribution of outliers on the data projection, $\mathbf{P}_{L_1}$ in (5) is likely to be closer to the true nominal rank-r subspace than L_2-PCA. The r columns of $\mathbf{P}_{L_1}$ in (5) are the so-called r L_1 principal components that describe the rank-r subspace in which $\mathbf{X}$ lies. As shown in [13], exact calculation of the L_1 principal components in $\mathcal{P}^{L_1}$ can be recast as a combinatorial problem. In short, when the rank of the nominal signal is $r = 1$, $\mathcal{P}^{L_1}$ reduces to

$$\mathbf{p}_{L_1} = \arg \max_{\substack{\mathbf{p} \in \mathbb{R}^D \\ \|\mathbf{p}\|_2 = 1}} \|\mathbf{X}^{\mathrm{T}}\mathbf{p}\|_1, \tag{6}$$

which can be reformulated as

$$\max_{\substack{\mathbf{p} \in \mathbb{R}^D \\ \|\mathbf{p}\|_2 = 1}} \|\mathbf{X}^T\mathbf{p}\|_1 = \max_{\substack{\mathbf{p} \in \mathbb{R}^D \\ \|\mathbf{p}\|_2 = 1}} \max_{\mathbf{b} \in \{\pm 1\}^N} \mathbf{b}^T\mathbf{X}^T\mathbf{p} = \max_{\mathbf{b} \in \{\pm 1\}^N} \max_{\substack{\mathbf{p} \in \mathbb{R}^D \\ \|\mathbf{p}\|_2 = 1}} \mathbf{p}^T\mathbf{X}\mathbf{b}$$
$$= \max_{\mathbf{b} \in \{\pm 1\}^N} \|\mathbf{X}\mathbf{b}\|_2 = \max_{\mathbf{b} \in \{\pm 1\}^N} (\mathbf{b}^T\mathbf{X}^T\mathbf{X}\mathbf{b})^{1/2}. \tag{7}$$

The optimal solution for (7) can be obtained by exhaustive search in the N-dimensional space of the binary antipodal vector $\mathbf{b}$ with complexity $\mathcal{O}(2^{N-1})$. Denote this optimal solution as $\mathbf{b}^{\mathrm{opt}}$, then the pursued principal component is given by

$$\mathbf{p}_{L_1} = \frac{\mathbf{X}\mathbf{b}^{\mathrm{opt}}}{\|\mathbf{X}\mathbf{b}^{\mathrm{opt}}\|_2}. \tag{8}$$

When the rank of the nominal data is $r > 1$, problem $\mathcal{P}^{L_1}$ can be recast into [13]

$$\max_{\substack{\mathbf{P} \in \mathbb{R}^{D\times r} \\ \mathbf{P}^{\mathrm{T}}\mathbf{P} = \mathbf{I}_r}} \|\mathbf{X}^{\mathrm{T}}\mathbf{P}\|_1 = \max_{\substack{\mathbf{P} \in \mathbb{R}^{D\times r} \\ \mathbf{P}^{\mathrm{T}}\mathbf{P} = \mathbf{I}_r}} \max_{\mathbf{B} \in \{\pm 1\}^{N\times r}} \mathrm{tr}(\mathbf{P}^{\mathrm{T}}\mathbf{X}\mathbf{B})$$
$$= \max_{\substack{\mathbf{P} \in \mathbb{R}^{D\times r} \\ \mathbf{P}^{\mathrm{T}}\mathbf{P} = \mathbf{I}_r}} \max_{\mathbf{B} \in \{\pm 1\}^{N\times r}} \mathrm{tr}(\mathbf{B}\mathbf{P}^{\mathrm{T}}\mathbf{X}) = \max_{\mathbf{B} \in \{\pm 1\}^{N\times r}} \|\mathbf{X}\mathbf{B}\|_* \tag{9}$$

where $\|\cdot\|_*$ stands for nuclear norm. By Proposition 4 of Markopoulos et al. [13], to find exactly the optimal L_1-norm projection operator $\mathbf{P}_{L_1}$ in (9) we can perform the following steps:

1) Solve (9) to obtain the optimal binary matrix $\mathbf{B}^{\mathrm{opt}} \in \{\pm 1\}^{N\times r}$.
2) Perform singular value decomposition (SVD) on $\mathbf{X}\mathbf{B}^{\mathrm{opt}}$, such that $\mathbf{X}\mathbf{B}^{\mathrm{opt}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$.
3) Return $\mathbf{P}_{L_1} = [\mathbf{U}]_{:,1:r}\mathbf{V}^{\mathrm{T}}$.[1]

2.2. Sub-optimal bit-flipping algorithm for L_1-PCA

From (7), we know that finding the optimal rank-1 L_1-subspace for a data matrix $\mathbf{X} \in \mathbb{R}^{D\times N}$ is equivalent to finding the optimal binary antipodal vector $\mathbf{b}^{\mathrm{opt}} \in \{\pm 1\}^N$. In [14], a fast bit-flipping (BF) algorithm was proposed to solve (7). The BF algorithm for rank $r = 1$ starts with an initial binary vector $\mathbf{b}^{(0)} \in \{\pm 1\}^N$, and iteratively produces a sequence of new binary vectors $\mathbf{b}^{(k)}$ (iteration index $k = 1, 2, \ldots$), in which $\mathbf{b}^{(k+1)}$ differs from $\mathbf{b}^{(k)}$ only in a single bit position, selected so as to achieve the highest increase of the quadratic value $\mathbf{b}^T\mathbf{X}^T\mathbf{X}\mathbf{b}$ in (7). Upon convergence, the BF algorithm generates a suboptimal binary vector $\mathbf{b}^c$ (superscript c stands for "at convergence"), then a suboptimal solution for the L_1 principal component $\mathbf{p}_{L_1}$ is obtained by

$$\mathbf{p}_{L_1} = \frac{\mathbf{X}\mathbf{b}^c}{\left\|\mathbf{X}\mathbf{b}^c\right\|_2}. \tag{10}$$

The above BF algorithm is extended in [15] for the calculation of $r > 1$ L_1 principal components. From (9), we know that the problem of finding r principal components is equivalent to finding a binary matrix $\mathbf{B} \in \{\pm 1\}^{N\times r}$ that maximizes $\|\mathbf{X}\mathbf{B}\|_*$. The corresponding BF algorithm starts with an initial binary matrix $\mathbf{B}^{(0)} \in \{\pm 1\}^{N\times r}$ and iteratively produces a sequence of new binary matrices $\mathbf{B}^{(k)}$ (iteration index $k = 1, 2, \ldots$), in which $\mathbf{B}^{(k+1)}$ differs from $\mathbf{B}^{(k)}$ only in a single bit position, selected so as to achieve the highest increase of $\|\mathbf{X}\mathbf{B}\|_*$ in (9). The associated suboptimal solution for the rank-r L_1-subspace $\mathbf{P}_{L_1}$ can be obtained by performing the following steps: 1) Run BF with input $\mathbf{B}^{(0)}$ to obtain a suboptimal binary matrix $\mathbf{B}^c \in \{\pm 1\}^{N\times r}$; 2) Perform SVD on $\mathbf{X}\mathbf{B}^c$ such that $\mathbf{X}\mathbf{B}^c = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$; and 3) Return $\mathbf{P}_{L_1} = [\mathbf{U}]_{:,1:r}\mathbf{V}^T$.

2.3. Iterative re-weighted L_1-PCA

Our preliminary study [19] proposed the iterative re-weighted L_1-PCA (L_1-IRW) that generates a sequence of improved rank-r L_1-subspaces $\mathbf{P}_{L_1}^{(k)} \in \mathbb{R}^{D\times r}$, with iteration index $k = 0, 1, \ldots$ for a fixed data matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D\times N}$. Initially, the batch L_1-subspace of (5) is obtained with the BF algorithm and denoted as $\mathbf{P}_{L_1}^{(0)}$. Then, $\mathbf{P}_{L_1}^{(0)}$ is iteratively updated. In the kth iteration, the L_1-subspace calculated in the $(k-1)$th iteration $\mathbf{P}_{L_1}^{(k-1)}$ is available. The conformity of the nth data sample $\mathbf{x}_n$ is measured by the L_2 error between $\mathbf{x}_n$ and its rank-r surrogate

$$d_n^{(k)} = \|\mathbf{x}_n - \mathbf{P}_{L_1}^{(k-1)}\mathbf{P}_{L_1}^{(k-1)^{\mathrm{T}}}\mathbf{x}_n\|_2, \quad n = 1, \ldots, N. \tag{11}$$

We expect large $d_n^{(k)}$ if $\mathbf{x}_n$ is an "outlier" and small $d_n^{(k)}$ if $\mathbf{x}_n$ is a nominal sample. Then the sample weight is defined as the inverse of the distance

$$w_n^{(k)} \triangleq (d_n^{(k)})^{-1}, \quad n = 1, \ldots, N, \tag{12}$$

followed by normalization,

$$\widetilde{w}_n^{(k)} = \frac{w_n^{(k)}}{\sum_{n=1}^{N} w_n^{(k)}}, \quad n = 1, \ldots, N. \tag{13}$$

When computing the L_1-subspace, data samples with larger weight should contribute more and samples with smaller weight should be suppressed such that the resulting calculated L_1-subspace is more accurate. Hence, in [19] we proposed that each data sample $\mathbf{x}_n$ is weighed by $\widetilde{w}_n^{(k)}$. We form a weight matrix

$$\widetilde{\mathbf{W}}^{(k)} \triangleq \begin{bmatrix} \widetilde{w}_1^{(k)} & 0 & 0 & \ldots \\ 0 & \widetilde{w}_2^{(k)} & 0 & \ldots \\ & \vdots & & \\ 0 & 0 & \ldots & \widetilde{w}_N^{(k)} \end{bmatrix} \tag{14}$$

and update the L_1-subspace by solving

$$\mathbf{P}_{L_1}^{(k)} = \arg \max_{\substack{\mathbf{P} \in \mathbb{R}^{D\times r} \\ \mathbf{P}^T\mathbf{P} = \mathbf{I}_r}} \|(\mathbf{X}\widetilde{\mathbf{W}}^{(k)})^{\mathrm{T}}\mathbf{P}\|_1 \tag{15}$$

with the BF algorithm. The approach automatically suppresses outliers in each iteration, resulting in a sequence of refined L_1-subspaces. When $\|\mathbf{w}^{(k)} - \mathbf{w}^{(k-1)}\|_2 < \epsilon$, where $\mathbf{w}^{(k)} \triangleq [w_1^{(k)}, w_2^{(k)}, \ldots, w_N^{(k)}]^T$ and $\epsilon > 0$ is a predefined threshold, the algorithm converges at a suboptimal rank-r subspace. It was demonstrated that this iterative sample re-weighting technique leads to more robust subspace estimation [19] than the original one-time L_1-subspace calculation.

3. Proposed L_1-subspace tracking

In this work, we propose an L_1-subspace tracking algorithm for streaming data. The proposed scheme is based on the BF technique [14], [15] and the preliminary study on L_1-IRW [19].

[1] $[\mathbf{U}]_{:,1:r}$ stands for the first r columns of matrix $\mathbf{U}$.

The problem statement is the following. At time slot $t-1$, the data matrix is $\mathbf{X}_{t-1}=\{\mathbf{x}_{t-1,n}\}_{n=1}^{N}\in\mathbb{R}^{D\times N}$, where $\mathbf{x}_{t-1,n}\in\mathbb{R}^{D}$ represents the nth sample (column) of the data matrix at time-slot $t-1$. Hereafter we use the same notation. Assume we obtained the following quantities at time-slot $t-1$: rank-1 subspace $\mathbf{p}_{t-1}^{c}\in\mathbb{R}^{D}$ (or rank-r subspace $\mathbf{P}_{t-1}^{c}\in\mathbb{R}^{D\times r}$), the suboptimal binary vector $\mathbf{b}_{t-1}^{c}\in\{\pm1\}^{N}$ (or binary matrix $\mathbf{B}_{t-1}^{c}\in\{\pm1\}^{N\times r}$), and the weight matrix $\mathbf{W}_{t-1}^{c}=\mathrm{diag}\{w_{t-1,1}^{c},\ldots,w_{t-1,N}^{c}\}$, where $w_{t-1,n}^{c}$ stands for the unnormalized weight at convergence for $\mathbf{x}_{t-1,n}$, $n=1,2,\ldots,N$.[2] At time-slot t, $t\geq 2$, a new datum $\mathbf{x}\in\mathbb{R}^{D}$ is acquired, and we aim at updating $\mathbf{p}_{t-1}^{c}$ ($\mathbf{P}_{t-1}^{c}$) to obtain $\mathbf{p}_{t}^{c}$ ($\mathbf{P}_{t}^{c}$). A straight-forward method is to incorporate the new datum in the old data matrix $\mathbf{X}_{t-1}$ and re-run L_1-IRW. Nevertheless, as more data are acquired, the size of the data matrix keeps increasing. Moreover, L_1-IRW with BF starts from an arbitrarily initialized binary vector (or matrix), resulting in slow convergence speed.

Instead, our task is to find $\mathbf{p}_{t}^{c}$ ($\mathbf{P}_{t}^{c}$) without solving L_1-IRW in (15) from scratch but by exploiting the results $\mathbf{p}_{t-1}^{c}$ ($\mathbf{P}_{t-1}^{c}$), $\mathbf{b}_{t-1}^{c}$ ($\mathbf{B}_{t-1}^{c}$), and $\mathbf{W}_{t-1}^{c}$ obtained in the previous time slot. In the following subsections, we elaborate the rank-1 and rank-r ($r>1$) cases in detail.

3.1. Rank-1 L_1-subspace tracking

The proposed rank-1 L_1-subspace tracking is outlined in Algorithm 1. The inputs are $\mathbf{X}_{t-1}$, $\mathbf{p}_{t-1}^{c}$, $\mathbf{W}_{t-1}^{c}$, $\mathbf{b}_{t-1}^{c}$, the new datum $\mathbf{x}$, and two positive parameters $0<\beta<1$ and $\epsilon\ll 1$. The outputs are the solutions at convergence for time-slot t: $\mathbf{p}_{t}^{c}$, $\mathbf{b}_{t}^{c}$, and $\mathbf{W}_{t}^{c}$. Algorithm 1 is executed every time a new datum is acquired, updating the rank-1 L_1-subspace on-the-fly.

Algorithm 1 Rank-1 L_1-subspace tracking.

Input: $\mathbf{X}_{t-1}\in\mathbb{R}^{D\times N}$, $\mathbf{p}_{t-1}^{c}\in\mathbb{R}^{D}$, $\mathbf{W}_{t-1}^{c}\in\mathbb{R}^{N\times N}$, $\mathbf{b}_{t-1}^{c}\in\{\pm1\}^{N}$,$\mathbf{x}\in\mathbb{R}^{D}$, $0<\beta<1$, $\epsilon\ll 1$.

$\mathbf{p}_{t}^{(0)}=\mathbf{p}_{t-1}^{c}$.
Remove $\mathbf{x}_{t-1,i}$ from $\mathbf{X}_{t-1}$ to obtain $\mathbf{X}_{t-1/i}\in\mathbb{R}^{D\times(N-1)}$.
$\mathbf{X}_{t}\leftarrow[\mathbf{X}_{t/N}=\mathbf{X}_{t-1/i}\ \ \mathbf{x}_{t,N}=\mathbf{x}]$.
$\mathbf{W}_{t/N}^{(0)}\leftarrow\mathbf{W}_{t-1/i}^{c}$, $\mathbf{b}_{t/N}^{(0)}\leftarrow\mathbf{b}_{t-1/i}^{c}$.
$d_{t,N}^{(0)}\leftarrow\|\mathbf{x}_{t,N}-\mathbf{p}_{t}^{(0)}\mathbf{p}_{t}^{(0)^{T}}\mathbf{x}_{t,N}\|_2$.
$w_{t,N}^{(0)}\leftarrow(d_{t,N}^{(0)})^{-1}$.
$\widetilde{w}_{t,n}^{(0)}\leftarrow w_{t,n}^{(0)}/\sum_{n=1}^{N}w_{t,n}^{(0)}$, $n=1,\ldots,N$.
$\widetilde{\mathbf{W}}_{t}^{(0)}\leftarrow\mathrm{diag}\{\widetilde{w}_{t,1}^{(0)},\ldots,\widetilde{w}_{t,N}^{(0)}\}$.
$(\mathbf{p}_{t}^{c},\mathbf{b}_{t}^{c},\mathbf{W}_{t}^{c})\leftarrow$ RANK1-L_1REFINE$(\mathbf{X}_{t},\mathbf{p}_{t}^{(0)},\mathbf{W}_{t}^{(0)},\widetilde{\mathbf{W}}_{t}^{(0)},\mathbf{b}_{t/N}^{(0)},\beta,\epsilon)$.

Output: $\mathbf{p}_{t}^{c}$, $\mathbf{b}_{t}^{c}$, $\mathbf{W}_{t}^{c}$.

To avoid processing an enlarging data matrix, we adopt a fixed processing window of N data samples. Before appending the new datum $\mathbf{x}$ to data matrix $\mathbf{X}_{t-1}$, one old datum is removed from $\mathbf{X}_{t-1}$. Assume that the ith datum is removed, $1\leq i\leq N$, and $\mathbf{X}_{t-1/i}\triangleq\{\mathbf{x}_{t-1,n}\}_{\substack{n=1\\ n\neq i}}^{N}\in\mathbb{R}^{D\times(N-1)}$ denotes the sub-matrix of $\mathbf{X}_{t-1}$ that excludes the ith sample (column). Specifically, the index of the datum to remove is selected by the following criterion:

$$i=\arg\min_{1\leq n\leq N}w_{t-1,n}^{c}. \tag{16}$$

In this way, we discard the datum with the minimum weight. Such a datum is more likely to be the outlier than other samples, therefore its removal purifies the current data matrix. Then, we append the new datum $\mathbf{x}$ to $\mathbf{X}_{t-1/i}$ and form the new data matrix at time t, $\mathbf{X}_{t}\triangleq[\mathbf{X}_{t/N}\ \ \mathbf{x}_{t,N}]\in\mathbb{R}^{D\times N}$, in which the first $N-1$ columns are $\mathbf{X}_{t/N}=\mathbf{X}_{t-1/i}$, and the last column is the new datum $\mathbf{x}_{t,N}=\mathbf{x}$.

With the new data matrix $\mathbf{X}_{t}$, we re-formulate the kth iteration of L_1-IRW in (15) as

$$\mathbf{p}_{t}^{(k)}=\arg\max_{\substack{\mathbf{p}\in\mathbb{R}^{D}\\ \|\mathbf{p}\|_2=1}}\|(\mathbf{X}_{t}\widetilde{\mathbf{W}}_{t}^{(k)})^{T}\mathbf{p}\|_1, \tag{17}$$

where $\widetilde{\mathbf{W}}_{t}^{(k)}$ is the normalized weight matrix for $\mathbf{X}_{t}$ at iteration k. Instead of solving (17) from scratch as the original L_1-IRW [19] does, we aim at intelligently running the iterations and updating $\mathbf{p}_{t}^{(k)}$ till it converges to $\mathbf{p}_{t}^{c}$. The key issue is to utilize the available information: $\mathbf{p}_{t-1}^{c}$, $\mathbf{b}_{t-1}^{c}$, and $\mathbf{W}_{t-1}^{c}$.

First, with similar derivation in (7), the problem in (17) is equivalent to finding the binary vector

$$\mathbf{b}_{t}^{(k)}=\arg\max_{\mathbf{b}\in\{\pm1\}^{N}}\mathbf{b}^{T}\widetilde{\mathbf{W}}_{t}^{(k)^{T}}\mathbf{X}_{t}^{T}\mathbf{X}_{t}\widetilde{\mathbf{W}}_{t}^{(k)}\mathbf{b}, \tag{18}$$

or equivalently,

$$\max_{\mathbf{b}_{t}^{(k)}\in\{\pm1\}^{N}}\Big\{\mathbf{b}_{t/N}^{(k)^{T}}\widetilde{\mathbf{W}}_{t/N}^{(k)^{T}}\mathbf{X}_{t/N}^{T}\mathbf{X}_{t/N}\widetilde{\mathbf{W}}_{t/N}^{(k)}\mathbf{b}_{t/N}^{(k)} + 2b_{t,N}^{(k)}\widetilde{w}_{t,N}^{(k)}\mathbf{x}_{t,N}^{T}\mathbf{X}_{t/N}\widetilde{\mathbf{W}}_{t/N}^{(k)}\mathbf{b}_{t/N}^{(k)}+(b_{t,N}^{(k)})^{2}(\widetilde{w}_{t,N}^{(k)})^{2}\mathbf{x}_{t,N}^{T}\mathbf{x}_{t,N}\Big\}, \tag{19}$$

where the objective function in (18) is decomposed into three terms, and the pursued binary vector $\mathbf{b}_{t}^{(k)}=[b_{t,1}^{(k)},\ldots,b_{t,N}^{(k)}]^{T}\in\{\pm1\}^{N}$ is decomposed into two parts: $\mathbf{b}_{t/N}^{(k)}=[b_{t,1}^{(k)},\ldots,b_{t,N-1}^{(k)}]^{T}\in\{\pm1\}^{N-1}$ associated with the old data $\mathbf{X}_{t/N}$, and the last bit $b_{t,N}^{(k)}$ associated with the new datum $\mathbf{x}_{t,N}$. The normalized weight matrix $\widetilde{\mathbf{W}}_{t}^{(k)}=\mathrm{diag}\{\widetilde{w}_{t,1}^{(k)},\ldots,\widetilde{w}_{t,N}^{(k)}\}$ is also decomposed into two parts: $\widetilde{\mathbf{W}}_{t/N}^{(k)}=\mathrm{diag}\{\widetilde{w}_{t,1}^{(k)},\ldots,\widetilde{w}_{t,N-1}^{(k)}\}$ associated with $\mathbf{X}_{t/N}$ and $\widetilde{w}_{t,N}^{(k)}$ associated with $\mathbf{x}_{t,N}$.

Second, initialize the rank-1 subspace in (17) by $\mathbf{p}_{t}^{(0)}=\mathbf{p}_{t-1}^{c}$. Since the partial data $\mathbf{X}_{t/N}$ of the current time-slot t are indeed the partial data $\mathbf{X}_{t-1/i}$ from time-slot $t-1$, for which the convergent weights and binary bits associated with $\mathbf{p}_{t-1}^{c}$ are available, we can then initialize the weights and binary bits for $\mathbf{X}_{t/N}$ as $\mathbf{W}_{t/N}^{(0)}=\mathbf{W}_{t-1/i}^{c}$ and $\mathbf{b}_{t/N}^{(0)}=\mathbf{b}_{t-1/i}^{c}$, respectively. For the new datum $\mathbf{x}_{t,N}$, its distance to $\mathbf{p}_{t}^{(0)}$ and its weight are initialized by $d_{t,N}^{(0)}=\|\mathbf{x}_{t,N}-\mathbf{p}_{t}^{(0)}\mathbf{p}_{t}^{(0)^{T}}\mathbf{x}_{t,N}\|_2$ and $w_{t,N}^{(0)}=(d_{t,N}^{(0)})^{-1}$. Subsequently, the normalized weights of all samples and the normalized weight matrix are initialized as $\widetilde{w}_{t,n}^{(0)}=w_{t,n}^{(0)}/\sum_{n=1}^{N}w_{t,n}^{(0)}$, $1\leq n\leq N$, and $\widetilde{\mathbf{W}}_{t}^{(0)}=\mathrm{diag}\{\widetilde{w}_{t,1}^{(0)},\ldots,\widetilde{w}_{t,N}^{(0)}\}$. The above initialization is followed by a rank-1 L_1-subspace refining function RANK1-L_1REFINE described in Algorithm 2.

In Algorithm 2, the inputs of the function RANK1-L_1REFINE are $\mathbf{X}_{t}$, $\mathbf{p}_{t}^{(0)}$, $\mathbf{W}_{t}^{(0)}$, $\widetilde{\mathbf{W}}_{t}^{(0)}$, $\mathbf{b}_{t/N}^{(0)}$, parameters $0<\beta<1$ and $\epsilon\ll 1$. The outputs of the function are the solutions at convergence $\mathbf{p}_{t}^{c}$, $\mathbf{b}_{t}^{c}$, $\mathbf{W}_{t}^{c}$ for time-slot t.

It first initializes the single bit $b_{t,N}^{(k)}$ associated with the new datum by maximizing the objective function in (19)

$$\begin{aligned}b_{t,N}^{(0)}&=\arg\max_{b\in\{\pm1\}}\{\mathbf{b}_{t/N}^{(0)^{T}}\widetilde{\mathbf{W}}_{t/N}^{(0)^{T}}\mathbf{X}_{t/N}^{T}\mathbf{X}_{t/N}\widetilde{\mathbf{W}}_{t/N}^{(0)}\mathbf{b}_{t/N}^{(0)}\\&\quad+2b\widetilde{w}_{t,N}^{(0)}\mathbf{x}_{t,N}^{T}\mathbf{X}_{t/N}\widetilde{\mathbf{W}}_{t/N}^{(0)}\mathbf{b}_{t/N}^{(0)}+b^{2}(\widetilde{w}_{t,N}^{(0)})^{2}\mathbf{x}_{t,N}^{T}\mathbf{x}_{t,N}\}\\&=\mathrm{sgn}\{\widetilde{w}_{t,N}^{(0)}\mathbf{x}_{t,N}^{T}\mathbf{X}_{t/N}\widetilde{\mathbf{W}}_{t/N}^{(0)}\mathbf{b}_{t/N}^{(0)}\}.\end{aligned} \tag{20}$$

Then the initial full binary vector can be formed as $\mathbf{b}_{t}^{(0)}=[\mathbf{b}_{t/N}^{(0)};b_{t,N}^{(0)}]\in\{\pm1\}^{N}$. Subsequently, the L_1-subspace is iteratively refined. In the kth iteration, BF is first executed to solve (18) for $\mathbf{b}_{t}^{(k)}$ where the initial normalized weight matrix and binary vector are

[2] Initially at $t=1$, these quantities $\mathbf{p}_{1}^{c}$ ($\mathbf{P}_{1}^{c}$), $\mathbf{b}_{1}^{c}$ ($\mathbf{B}_{1}^{c}$), and $\mathbf{W}_{1}^{c}$ are obtained by L_1-IRW in (15) with the initial data matrix $\mathbf{X}_{1}$.

Algorithm 2 Function $(\mathbf{p}_t^c, \mathbf{b}_t^c, \mathbf{W}_t^c) \leftarrow$ RANK1-L_1REFINE$(\mathbf{X}_t, \mathbf{p}_t^{(0)}, \mathbf{W}_t^{(0)}, \widetilde{\mathbf{W}}_t^{(0)}, \mathbf{b}_{t/N}^{(0)}, \beta, \epsilon)$.

Input: $\mathbf{X}_t \in \mathbb{R}^{D\times N}$, $\mathbf{p}_t^{(0)} \in \mathbb{R}^D$, $\mathbf{W}_t^{(0)} \in \mathbb{R}^{N\times N}$, $\widetilde{\mathbf{W}}_t^{(0)} \in \mathbb{R}^{N\times N}$, $\mathbf{b}_{t/N}^{(0)} \in \{\pm 1\}^{N-1}$,$\beta$, ϵ.

1: Initialize $b_{t,N}^{(0)}$ by (20).
2: $\mathbf{b}_t^{(0)} \leftarrow [\mathbf{b}_{t/N}^{(0)}; b_{t,N}^{(0)}]$.
for $k = 1, 2, \ldots,$ **do**
3: $\mathbf{b}_t^{(k)} \leftarrow \mathrm{BF}(\mathbf{b}_t^{(k-1)}, \mathbf{X}_t, \widetilde{\mathbf{W}}_t^{(k-1)})$.
4: $\mathbf{p}_t^{(k)} \leftarrow \mathbf{X}_t\widetilde{\mathbf{W}}_t^{(k-1)}\mathbf{b}_t^{(k)} / \|\mathbf{X}_t\widetilde{\mathbf{W}}_t^{(k-1)}\mathbf{b}_t^{(k)}\|_2$.
5: $d_{t,n}^{(k)} \leftarrow \|\mathbf{x}_{t,n} - \mathbf{p}_t^{(k)}\mathbf{p}_t^{(k)^T}\mathbf{x}_{t,n}\|_2$, $1 \le n \le N$.
6: $u_{t,n}^{(k)} \leftarrow (d_{t,n}^{(k)})^{-1}$, update $w_{t,n}^{(k)}$ by (22), $1 \le n \le N$.
7: Check stopping criterion: if $\|\mathbf{w}_t^{(k)} - \mathbf{w}_t^{(k-1)}\|_2 < \epsilon$, then $\mathbf{p}_t^c \leftarrow \mathbf{p}_t^{(k)}$, $\mathbf{b}_t^c \leftarrow \mathbf{b}_t^{(k)}$, $\mathbf{w}_t^c \leftarrow \mathbf{w}_t^{(k)}$, $\mathbf{W}_t^c \leftarrow \mathbf{W}_t^{(k)}$. Exit.
8: $\widetilde{w}_{t,n}^{(k)} \leftarrow w_{t,n}^{(k)} / \sum_{n=1}^{N} w_{t,n}^{(k)}$, $\widetilde{\mathbf{W}}_t^{(k)} \leftarrow \mathrm{diag}\{\widetilde{w}_{t,1}^{(k)}, \ldots, \widetilde{w}_{t,N}^{(k)}\}$.
end for
Output: $\mathbf{p}_t^c$, $\mathbf{b}_t^c$, $\mathbf{W}_t^c$.

$\widetilde{\mathbf{W}}_t^{(k-1)}$ and $\mathbf{b}_t^{(k-1)}$, respectively. Then the updated $\mathbf{b}_t^{(k)}$ is utilized to update $\mathbf{p}_t^{(k)}$ in a similar way as in (10), followed by sample distance update as in (11) and sample weight update. Specifically, we modify the weight update formula (12) as follows to guarantee a convergent weight sequence. Once the distance $d_{t,n}^{(k)}$ is obtained, we define

$$u_{t,n}^{(k)} \triangleq (d_{t,n}^{(k)})^{-1} \tag{21}$$

and update the weight $w_{t,n}^{(k)}$ by

$$w_{t,n}^{(k)} = \begin{cases} w_{t,n}^{(k-1)}(1-\beta^k), & \text{if } u_{t,n}^{(k)} < w_{t,n}^{(k-1)}(1-\beta^k), \\ u_{t,n}^{(k)}, & \text{if } w_{t,n}^{(k-1)}(1-\beta^k) \le u_{t,n}^{(k)} \le w_{t,n}^{(k-1)}(1+\beta^k), \\ w_{t,n}^{(k-1)}(1+\beta^k), & \text{if } u_{t,n}^{(k)} > w_{t,n}^{(k-1)}(1+\beta^k) \end{cases} \tag{22}$$

where $0 < \beta < 1$ is a pre-defined parameter, and β^k is the kth power of β. Intuitively, we avoid updating the weights too aggressively by restricting the new weight $w_{t,n}^{(k)}$ to be within a small neighborhood of the weight in the previous iteration $w_{t,n}^{(k-1)}$. The size of the neighborhood depends on β. The convergence of the weight sequence can be verified by

$$\lim_{k\to\infty} \beta^k = 0, \tag{23}$$

$$\lim_{k\to\infty} (w_{t,n}^{(k)} - w_{t,n}^{(k-1)}) = 0. \tag{24}$$

Then we check the stopping criterion: if the L_2-distance between the updated weight vector $\mathbf{w}_t^{(k)} \triangleq [w_{t,1}^{(k)}, w_{t,2}^{(k)}, \ldots, w_{t,N}^{(k)}]^T$ and the previous one $\mathbf{w}_t^{(k-1)} \triangleq [w_{t,1}^{(k-1)}, w_{t,2}^{(k-1)}, \ldots, w_{t,N}^{(k-1)}]^T$ is less than ϵ, we exit the RANK1-L_1REFINE function and returns the convergent quantities $\mathbf{p}_t^c = \mathbf{p}_t^{(k)}$, $\mathbf{b}_t^c = \mathbf{b}_t^{(k)}$, $\mathbf{w}_t^c = \mathbf{w}_t^{(k)}$, $\mathbf{W}_t^c = \mathbf{W}_t^{(k)}$. Otherwise, we update the normalized weights and weight matrix in step 8 and continue with the iterations.

3.2. *Rank-*r *(*r > 1*)* L_1*-subspace tracking*

We extend the rank-1 L_1-subspace tracking to rank-r $(r > 1)$ L_1-subspace tracking in Algorithm 3. In contrast to Algorithm 1, the inputs of Algorithm 3 are: the rank-r subspace $\mathbf{P}_{t-1}^c \in \mathbb{R}^{D\times r}$, and the binary matrix $\mathbf{B}_{t-1}^c \in \{\pm 1\}^{N\times r}$ obtained at time-slot $t-1$. The outputs are: the rank-r subspace $\mathbf{P}_t^c \in \mathbb{R}^{D\times r}$, the binary matrix

Algorithm 3 Rank-r $(r > 1)$ L_1-subspace tracking.

Input: $\mathbf{X}_{t-1} \in \mathbb{R}^{D\times N}$, $\mathbf{P}_{t-1}^c \in \mathbb{R}^{D\times r}$, $\mathbf{W}_{t-1}^c \in \mathbb{R}^{N\times N}$, $\mathbf{B}_{t-1}^c \in \{\pm 1\}^{N\times r}$,$\mathbf{x} \in \mathbb{R}^D$, $0 < \beta < 1$, $\epsilon \ll 1$.

1: $\mathbf{P}_t^{(0)} = \mathbf{P}_{t-1}^c$.
2: Remove $\mathbf{x}_{t-1,i}$ from $\mathbf{X}_{t-1}$ to obtain$\mathbf{X}_{t-1/i} \in \mathbb{R}^{D\times(N-1)}$.
3: $\mathbf{X}_t \leftarrow [\mathbf{X}_{t/N} = \mathbf{X}_{t-1/i} \quad \mathbf{x}_{t,N} = \mathbf{x}]$.
4: $\mathbf{W}_{t/N}^{(0)} \leftarrow \mathbf{W}_{t-1/i}^c$, $\mathbf{B}_{t/N}^{(0)} \leftarrow \mathbf{B}_{t-1/i}^c$.
5: $d_{t,N}^{(0)} \leftarrow \|\mathbf{x}_{t,N} - \mathbf{P}_t^{(0)}\mathbf{P}_t^{(0)^T}\mathbf{x}_{t,N}\|_2$.
6: $w_{t,N}^{(0)} \leftarrow (d_{t,N}^{(0)})^{-1}$.
7: $\widetilde{w}_{t,n}^{(0)} \leftarrow w_{t,n}^{(0)} / \sum_{n=1}^{N} w_{t,n}^{(0)}$, $n = 1, \ldots, N$.
8: $\widetilde{\mathbf{W}}_t^{(0)} \leftarrow \mathrm{diag}\{\widetilde{w}_{t,1}^{(0)}, \ldots, \widetilde{w}_{t,N}^{(0)}\}$.
9: $(\mathbf{P}_t^c, \mathbf{B}_t^c, \mathbf{W}_t^c) \leftarrow$ RANKr-L_1REFINE$(\mathbf{X}_t, \mathbf{P}_t^{(0)}, \mathbf{W}_t^{(0)}, \widetilde{\mathbf{W}}_t^{(0)}, \mathbf{B}_{t/N}^{(0)}, \beta, \epsilon)$.

Output: $\mathbf{P}_t^c \in \mathbb{R}^{D\times r}$, $\mathbf{B}_t^c \in \{\pm 1\}^{N\times r}$, $\mathbf{W}_t^c \in \mathbb{R}^{N\times N}$.

$\mathbf{B}_t^c \in \{\pm 1\}^{N\times r}$, and the weight matrix $\mathbf{W}_t^c \in \mathbb{R}^{N\times N}$ at convergence at time-slot t. Steps 1 to 3 initialize $\mathbf{P}_t^{(0)}$ and construct $\mathbf{X}_t$ in a similar way as in Algorithm 1. While Algorithm 1 iteratively solves (17), Algorithm 3 iteratively solves the following problem

$$\mathbf{P}_t^{(k)} = \arg \max_{\substack{\mathbf{P}\in\mathbb{R}^{D\times r} \\ \mathbf{P}^T\mathbf{P} = \mathbf{I}_r}} \|(\mathbf{X}_t\widetilde{\mathbf{W}}_t^{(k)})^T\mathbf{P}\|_1. \tag{25}$$

Following the derivation in (9), the above maximization problem is equivalent to

$$\mathbf{P}_t^{(k)} = \arg \max_{\substack{\mathbf{P}\in\mathbb{P}^{D\times r} \\ \mathbf{P}^T\mathbf{P} = \mathbf{I}_r}} \max_{\mathbf{B}\in\{\pm 1\}^{N\times r}} \mathrm{tr}(\mathbf{B}\mathbf{P}^T\mathbf{X}_t\widetilde{\mathbf{W}}_t^{(k)}), \tag{26}$$

which is equivalent to solving

$$\mathbf{B}_t^{(k)} = \arg \max_{\mathbf{B}\in\{\pm 1\}^{N\times r}} \|\mathbf{X}_t\widetilde{\mathbf{W}}_t^{(k)}\mathbf{B}\|_*. \tag{27}$$

In (26), the optimal nth row of binary matrix $\mathbf{B}$ is given by $\mathbf{b}_n^{\mathrm{opt}} = \mathrm{sgn}\{\mathbf{P}^T\mathbf{x}_{t,n}\widetilde{w}_{t,n}^{(k)}\} \in \{\pm 1\}^r$, that is, projecting the weighted nth data sample onto the rank-r subspace $\mathbf{P}$, followed by a sign operation, which means that the nth row of $\mathbf{B}$ ($\mathbf{b}_n \in \{\pm 1\}^r$) is associated with the nth data sample $\mathbf{x}_{t,n}$. Hence, in rank-r L_1-subspace tracking, the partial data matrix $\mathbf{X}_{t/N}$ are associated with the partial binary matrix $\mathbf{B}_{t/N}$, which represents an $(N-1)\times r$ binary matrix formed by removing the Nth row of $\mathbf{B}_t$. In step 4 of Algorithm 3, $\mathbf{B}_{t/N}$ is initialized by $\mathbf{B}_{t-1/i}^c$, the convergent binary matrix from time-slot $t-1$ excluding the ith row. Steps 5–8 are similar to those in Algorithm 1, which initialize the weights and weight matrix for data samples. In step 9, the rank-r L_1-subspace refining function RANKr-L_1REFINE is called to generate the rank-r subspace $\mathbf{P}_t^c \in \mathbb{R}^{D\times r}$, the binary matrix $\mathbf{B}_t^c \in \{\pm 1\}^{N\times r}$, and the weight matrix $\mathbf{W}_t^c \in \mathbb{R}^{N\times N}$ at convergence.

The function RANKr-L_1REFINE is outlined in Algorithm 4. The inputs and outputs are similar to those in Algorithm 3. It first initializes the Nth row of $\mathbf{B}_t$ (denoted as a column vector $\mathbf{b}_{t,N} \in \{\pm 1\}^r$, the r binary bits associated with the new datum) as

$$\mathbf{b}_{t,N}^{(0)} = \arg \max_{\mathbf{b}\in\{\pm 1\}^r} \|\mathbf{X}_t\mathbf{B}_t^{(0)}\|_* = \left\| [\mathbf{X}_{t/N} \quad \mathbf{x}_{t,N}] \begin{bmatrix} \mathbf{B}_{t/N}^{(0)} \\ \mathbf{b}^T \end{bmatrix} \right\|_*. \tag{28}$$

Then the initial full binary matrix can be formed as $\mathbf{B}_t^{(0)} = [\mathbf{B}_{t/N}^{(0)}; \mathbf{b}_{t,N}^{(0)^T}]$. Subsequently, the L_1-subspace $\mathbf{P}_t^{(k)}$ is iteratively refined. In the kth iteration, BF is first executed (step 3) to solve (27) for $\mathbf{B}_t^{(k)}$ where the initial binary matrix is $\mathbf{B}_t^{(k-1)}$. Then the updated $\mathbf{B}_t^{(k)}$ is utilized to update $\mathbf{P}_t^{(k)}$ (steps 4 and 5). Steps 6–9 are

Algorithm 4 Function $(\mathbf{P}_t^c, \mathbf{B}_t^c, \mathbf{W}_t^c) \leftarrow$ RANKr-L_1REFINE$(\mathbf{X}_t, \mathbf{P}_t^{(0)}, \mathbf{W}_t^{(0)}, \widetilde{\mathbf{W}}_t^{(0)}, \mathbf{B}_{t/N}^{(0)}, \beta, \epsilon)$.

Input: $\mathbf{X}_t \in \mathbb{R}^{D\times N}$, $\mathbf{P}_t^{(0)} \in \mathbb{R}^D$, $\mathbf{W}_t^{(0)} \in \mathbb{R}^{N\times N}$, $\widetilde{\mathbf{W}}_t^{(0)} \in \mathbb{R}^{N\times N}$, $\mathbf{B}_{t/N}^{(0)} \in \{\pm 1\}^{(N-1)\times r}$,$\beta$, ϵ.

Initialize $\mathbf{b}_{t,N}^{(0)}$ by (28).

$\mathbf{B}_t^{(0)} \leftarrow [\mathbf{B}_{t/N}^{(0)}; \mathbf{b}_{t,N}^{(0)^T}]$.

for $k = 1, 2, \ldots,$ **do**

$\mathbf{B}_t^{(k)} \leftarrow \text{BF}(\mathbf{B}_t^{(k-1)}, \mathbf{X}_t, \widetilde{\mathbf{W}}_t^{(k-1)})$.

$(\mathbf{U}_t^{(k)}, \mathbf{S}_t^{(k)}, \mathbf{V}_t^{(k)}) \leftarrow \text{SVD}(\mathbf{X}_t \widetilde{\mathbf{W}}_t^{(k-1)} \mathbf{B}_t^{(k)})$.

$\mathbf{P}_t^{(k)} \leftarrow [\mathbf{U}_t^{(k)}]_{:,1:r} \mathbf{V}_t^{(k)^T}$.

$d_{t,n}^{(k)} \leftarrow \|\mathbf{x}_{t,n} - \mathbf{P}_t^{(k)} \mathbf{P}_t^{(k)^T} \mathbf{x}_{t,n}\|_2$, $1 \le n \le N$.

$u_{t,n}^{(k)} \leftarrow (d_{t,n}^{(k)})^{-1}$, update $w_{t,n}^{(k)}$ by (22), $1 \le n \le N$.

Check stopping criterion: if $\|\mathbf{w}_t^{(k)} - \mathbf{w}_t^{(k-1)}\|_2 < \epsilon$, then $\mathbf{P}_t^c \leftarrow \mathbf{P}_t^{(k)}$, $\mathbf{B}_t^c \leftarrow \mathbf{B}_t^{(k)}$, $\mathbf{w}_t^c \leftarrow \mathbf{w}_t^{(k)}$, $\mathbf{W}_t^c \leftarrow \mathbf{W}_t^{(k)}$. Exit.

$\widetilde{w}_{t,n}^{(k)} \leftarrow w_{t,n}^{(k)} / \sum_{n=1}^{N} w_{t,n}^{(k)}$, $\widetilde{\mathbf{W}}_t^{(k)} \leftarrow \text{diag}\{\widetilde{w}_{t,1}^{(k)}, \ldots, \widetilde{w}_{t,N}^{(k)}\}$.

end for

Output: $\mathbf{P}_t^c$, $\mathbf{B}_t^c$, $\mathbf{W}_t^c$.

the same as steps 5–8 in Algorithm 2, with $\mathbf{P}_t^{(k)}, \mathbf{P}_t^c, \mathbf{B}_t^{(k)}$, and $\mathbf{B}_t^c$ replacing $\mathbf{p}_t^{(k)}, \mathbf{p}_t^c, \mathbf{b}_t^{(k)}$, and $\mathbf{b}_t^c$.

4. Applications and experimental studies

In this section, we assess the effectiveness of the proposed L_1-subspace tracking algorithm through four experimental studies: (i) synthetic data example, (ii) moving objects detection from streaming surveillance videos, (iii) robust online cooperative spectrum sensing in a cognitive radio network, and (iv) DoA tracking. We compare the proposed method (named "L_1-Tracking") with the batch L_1-PCA [13] (named "L_1-Batch"), the L_1-IRW [19], the batch L_2-PCA (named "L_2-Batch"), the GRASTA [27], the PracReProCS [31], and the OMoGMF [32] schemes, in terms of performance and execution time. All the experiments in this work were implemented on a personal computer with i7 CPU and 16G RAM.

4.1. Synthetic data example

We create a synthetic data example to evaluate the performance of the proposed online L_1-subspace tracking algorithm in controlled conditions and to assess the impact of design parameters N and β.

We generate random streaming measurements of $\mathbf{x}_t$, $t = 1, 2, \ldots$ from a rank-4 subspace in $\mathbb{R}^{100}$, spanned by the columns of a random matrix $\mathbf{P}_{\text{true}} \in \mathbb{R}^{100\times 4}$ that has orthonormal columns. The measurement $\mathbf{x}_t$ is corrupted by outliers with probability 0.3, that is, $\mathbf{x}_t = \mathbf{P}_{\text{true}}\mathbf{a}_t + \mathbf{s}_t$, where $\mathbf{a}_t$ are Gaussian random vectors in $\mathbb{R}^4$, and $\mathbf{s}_t \in \mathbb{R}^{100}$ are outlier vectors with nonzero Gaussian random coefficients in 50% of their entries. We apply the proposed L_1-Tracking to estimate the underlying rank-4 subspace $\mathbf{P}_{\text{true}}$, and evaluate the performance in terms of the subspace estimation error between the updated subspace $\mathbf{P}_t^{(k)}$ and the true subspace $\mathbf{P}_{\text{true}}$, which is defined as [28]

$$\text{Error}_t = \frac{\|\mathbf{P}_t^{(k)}\mathbf{P}_t^{(k)^\dagger} - \mathbf{P}_{\text{true}}\mathbf{P}_{\text{true}}^\dagger\|_F}{\|\mathbf{P}_{\text{true}}\mathbf{P}_{\text{true}}^\dagger\|_F}, \tag{29}$$

where $\dagger$ and $\|\cdot\|_F$ stands for the pseudo-inverse and the Frobenius-norm of a matrix.

Fig. 1 (a) shows the subspace estimation error versus the number of new data samples, for different processing-window size $N = 20, 30$, and 40. The subspace is initialized with N data samples, and is updated as $\mathbf{P}_t^c \in \mathbb{R}^{100\times 4}$ at the arrival of every new datum $\mathbf{x}_t$, $t = 1, 2, \ldots, 200$. The weight update parameter is fixed at $\beta = 0.5$. It is observed that for all N values, the subspace estimation error decreases as the subspace is being updated with new data samples. In particular, a smaller N value leads to faster convergence rate.

In Fig. 1(b), for a fixed processing-window size $N = 20$, the subspace estimation error when updating the subspace at time-slot $t = 15$ using the 15th new datum is evaluated versus the number of iterations k for weight update for various values of design parameter $\beta = 0.1, 0.3, 0.5$, and 0.7. As specified in Algorithms 2 and 4, $0 < \beta < 1$ is an input parameter in the L_1-subspace refining functions RANK1-L_1REFINE and RANKr-L_1REFINE. As described in the weight update Eq. (22), the new weight $w_{t,n}^{(k)}$ is confined to a small neighborhood centered at the weight in the previous iteration $w_{t,n}^{(k-1)}$, and $\beta^k w_{t,n}^{(k-1)}$ is the radius of the neighborhood. Since $0 < \beta < 1$, the weight update procedure is guaranteed to converge by Eqs. (23) and (24). When $\beta \to 0$, the neighborhood is infinitely small and the weight update terminates after one iteration. In addition, a larger β leads to slow convergence while a smaller β leads to fast convergence, which is demonstrated in Fig. 1(b). In Fig. 1(b), as expected, $\beta = 0.1$ leads to fast convergence and the resulting $\mathbf{P}_t^c$ is still far away from $\mathbf{P}_{\text{true}}$, while $\beta = 0.5$ and $\beta = 0.7$ lead to slower convergence rate but they achieve lower subspace estimation error. The estimation error Error_t at convergence for different β values are labeled on the curves.

4.2. Moving objects detection from streaming surveillance videos

Consider a sequence of surveillance video frames $\mathbf{X}_t \in \mathbb{R}^{m\times n}$ with frame resolution of $m \times n$ pixels and time index $t = 1, 2, \ldots$. A typical surveillance video sequence is consisted of a background scene that can be modeled as a low-rank component, and sparse foreground moving objects superimposed on the background scene that are regarded as the outliers. For security monitoring, the objective is to extract the moving objects.

Each video frame $\mathbf{X}_t$ is vectorized as $\mathbf{x}_t \in \mathbb{R}^D$, $D = m \times n$ via column concatenation. We model the background scene as a low-rank component $\mathbf{z}_t \in \mathbb{R}^D$ and the foreground moving objects as a sparse component $\mathbf{s}_t \in \mathbb{R}^D$. Hence,

$$\mathbf{x}_t = \mathbf{z}_t + \mathbf{s}_t, \quad t = 1, 2, \ldots. \tag{30}$$

Consider a group of N frames, the matrix-form representation is

$$\mathbf{X} = \mathbf{Z} + \mathbf{S}, \tag{31}$$

where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D\times N}$, $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_N] \in \mathbb{R}^{D\times N}$, and $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_N] \in \mathbb{R}^{D\times N}$. To extract the low-rank background, a simple method is to run rank-1 L_1-Batch on $\mathbf{X}$ and obtain the L_1-subspace $\mathbf{p}_{L_1} \in \mathbb{R}^D$, or to run L_1-IRW and obtain the L_1-subspace $\mathbf{p}_{L_1}^c \in \mathbb{R}^D$ at convergence. Afterwards, the background can be approximated by $\widehat{\mathbf{Z}} = \mathbf{p}_{L_1}\mathbf{p}_{L_1}^T\mathbf{X}$ (or $\widehat{\mathbf{Z}} = \mathbf{p}_{L_1}^c\mathbf{p}_{L_1}^{cT}\mathbf{X}$) and the foreground can be extracted as $\widehat{\mathbf{S}} = \mathbf{X} - \widehat{\mathbf{Z}}$. For our proposed L_1-Tracking, we initialize the rank-1 subspace $\mathbf{p}_0^c$ with the initial $N = 8$ frames using L_1-IRW. Subsequently, we update the subspace $\mathbf{p}_t^c \in \mathbb{R}^D$ at the arrival of every new frame $\mathbf{x}_t$, $t = 1, 2, \ldots$. We keep the processing window at $N = 8$.

We first test the proposed L_1-Tracking, the L_1-IRW, and the L_1-Batch algorithms on a subset of 80 frames from the *Lobby* video sequence. Each frame is of 128×160 pixels. This is a challenging video sequence since there is illumination change in the background. The processing window is $N = 8$ for all schemes in comparison. Fig. 2 displays the background and foreground extracted at multiple distinct time slots $t = 10, 13, 30, 51, 54$ by the proposed

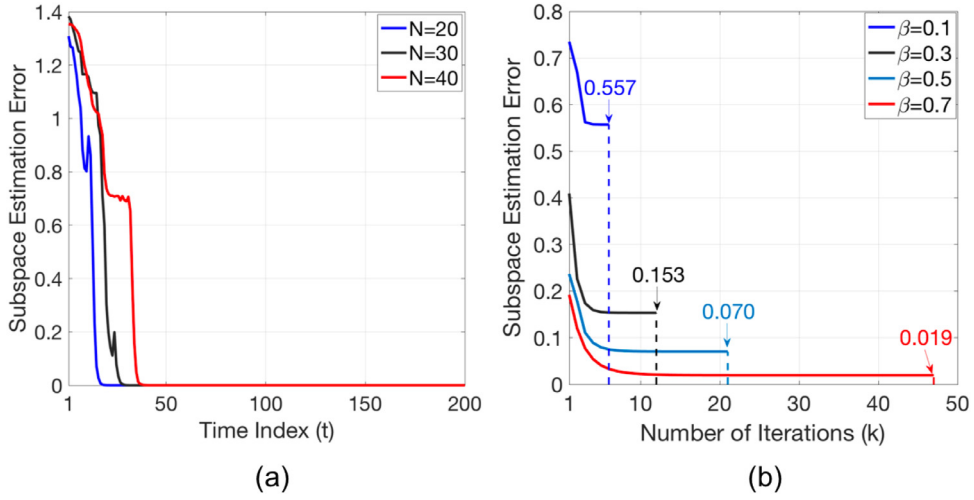

Fig. 1. (a) Subspace estimation error with different processing-window size $N = 20, 30,$ and 40. The sample-weight update parameter is fixed at $\beta = 0.5$. (b) Subspace estimation error with different sample-weight update parameter $\beta = 0.1$ to 0.7. The processing-window size is fixed at $N = 20$.

L_1-Tracking, the L_1-IRW [19], and the regular L_1-Batch [13] methods. Fig. 2.(a) shows the original frames, where the background is bright for $t = 10, 13,$ and is dark for the remaining three frames. In frames $t = 10, 30, 51, 54,$ a person appears in the scene as a moving object. From Fig. 2.(b1), we observe that the proposed L_1-Tracking successfully recovers the background and adapts to the illumination change. The corresponding extracted foreground in grayscale is displayed in Fig. 2.(c1), and the binary masks for the detected moving objects are displayed in Fig. 2.(d1). In contrast, the L_1-IRW and L_1-Batch cannot accurately recover the background scenes (Fig. 2.(b2)(b3)), which cause "ghost" phenomenon in the extracted grayscale foreground scenes (Fig. 2.(c2)(c3)), and the binary masks (Fig. 2.(d3)).

Besides, we run the proposed L_1-Tracking algorithm on the complete Lobby video sequence of 1546 frames, and compare its receiver operating characteristic (ROC) curve with those generated by the GRASTA [27], PracReProCS [31], and OMoGMF [32] algorithms. For a fair comparison, for all four schemes we use the first $N = 8$ frames for subspace initialization, and the remaining 1538 frames for online subspace update. For fast convergence, the sample weight update parameter is set as $\beta = 0.5$ for the proposed L_1-Tracking. For OMoGMF, a mixture of 2 Gaussians is used to model the foreground. As shown in Fig. 3(a), our proposed L_1-Tracking achieves the highest true positive rate (TPR) under the same false positive rate (FPR) compared to the other three schemes. Fig. 3(b) shows the accumulated execution time for all four subspace tracking methods as the new frame index increases. Compared to PracReProCS, the proposed L_1-Tracking, the GRASTA, and the OMoGMF methods have significant saving in execution time.

4.3. Robust cooperative spectrum sensing in cognitive radio networks

Radio frequency spectrum is a scarce resource in wireless communications due to the ever-increasing wireless channel users. Spectrum-sensing cognitive radio is a technique that allows secondary users to detect the idle spectrum and share the wireless channel with primary users in an opportunistic manner [35]. We consider the robust cooperative spectrum sensing problem in a cognitive radio network (CRN) when malicious attacks exist [36,37]. The CRN in Fig. 4 consists of a primary user (PU), multiple secondary users (SUs) and a fusion center. The PU transmits signals on the wireless channel and the SUs monitor the PU's status (presence or absence). At time-slot t, the received PU signal power (dB) at the mth SU can be expressed as

$$y_{m,t} = \Gamma_t + \alpha 10\log_{10}(d_0/d_m) + o_{m,t} \quad \text{dB}, \tag{32}$$

where Γ_t is the PU transmission power (in dB) at time-slot t, α is the path-loss exponent, d_0 is the reference distance, and d_m is the distance between the PU and the mth SU which is measured prior via geo-location database. The parameters Γ_t and α are unknown at the fusion center and need to be estimated. When attackers attack the mth SU at time t, the received signal $y_{m,t}$ has an extra additive component $o_{m,t}$ (dB), which is considered as the outlier. All SUs send their sensed signal $y_{m,t}$ to the fusion center. The objective of the fusion center is to recover the transmission power Γ_t, $t = 1, 2, \ldots$ reported by the SUs, compare it with a threshold and determine whether the PU exists or not.

Consider M SUs and sensing time slots $t = 1, 2, \ldots, N$, define matrices $\mathbf{H}_{M\times 2} \triangleq [\begin{smallmatrix}1 & 1 & \cdots & 1\\ \beta_1 & \beta_2 & \cdots & \beta_M\end{smallmatrix}]^T$ and $\mathbf{X}_{2\times N} \triangleq [\begin{smallmatrix}\Gamma_1 & \Gamma_2 & \cdots & \Gamma_N\\ \alpha & \alpha & \cdots & \alpha\end{smallmatrix}]$, in which $\beta_m \triangleq 10\log_{10}(d_0/d_m)$, then the data at the fusion center collected in the period of N time slots can be modeled as $\mathbf{Y} = \mathbf{HX} + \mathbf{O} \in \mathbb{R}^{M\times N}$, where the (m, t)th entry of the outlier matrix $\mathbf{O}$ is $o_{m,t}$. Define $\mathbf{L} \triangleq \mathbf{HX}$, then $\mathbf{Y} = \mathbf{L} + \mathbf{O}$. Since $\text{rank}(\mathbf{H}) \le 2$ and $\text{rank}(\mathbf{X}) \le 2$, we have $\text{rank}(\mathbf{L}) \le 2$.

To solve the power estimation problem in the presence of outlier $\mathbf{O}$, we can apply L_1-Batch to data matrix $\mathbf{Y}$ and estimate the rank-2 subspace $\mathbf{P}_{L_1} \in \mathbb{R}^{M\times 2}$ in which the low-rank matrix $\mathbf{L}$ lies, that is,

$$\mathbf{P}_{L_1} = \arg \max_{\substack{\mathbf{P}\in\mathbb{R}^{M\times 2}\\ \mathbf{P}^T\mathbf{P}=\mathbf{I}}} \|\mathbf{Y}^T\mathbf{P}\|_1. \tag{33}$$

Then $\mathbf{L}$ and $\mathbf{X}$ can be recovered by $\widehat{\mathbf{L}} = \mathbf{P}_{L_1}\mathbf{P}_{L_1}^T\mathbf{Y}$ and $\widehat{\mathbf{X}} = \mathbf{H}^{\dagger}\widehat{\mathbf{L}}$[3], respectively. In the sequel, the PU transmission power (in dB) can be obtained from the first row of $\widehat{\mathbf{X}}$, which is $\widehat{\mathbf{\Gamma}} = [\widehat{\Gamma}_1, \widehat{\Gamma}_2, \ldots, \widehat{\Gamma}_N]^T$.

In our study, we run 100 independent experiments, and each experiment has $N_{\text{total}} = 60$ snapshots. The reference distance d_0 is $20m$, pathloss coefficient is set to $\alpha = 4$, and $M = 40$ SUs are deployed. The PU transmission power is uniformly distributed between 1000 Watt and 1100 Watt, such that $30\text{dB} \le \Gamma_t \le 30.4139\text{dB}$, and the distance between the PU and the mth SU is uniformly distributed between 5km and 6km. We fix the attack amplitude from all attackers to be 20dB, and 8% of the sensed signals are

[3] $\mathbf{H}\dagger$ is the pseudo-inverse of $\mathbf{H}$.

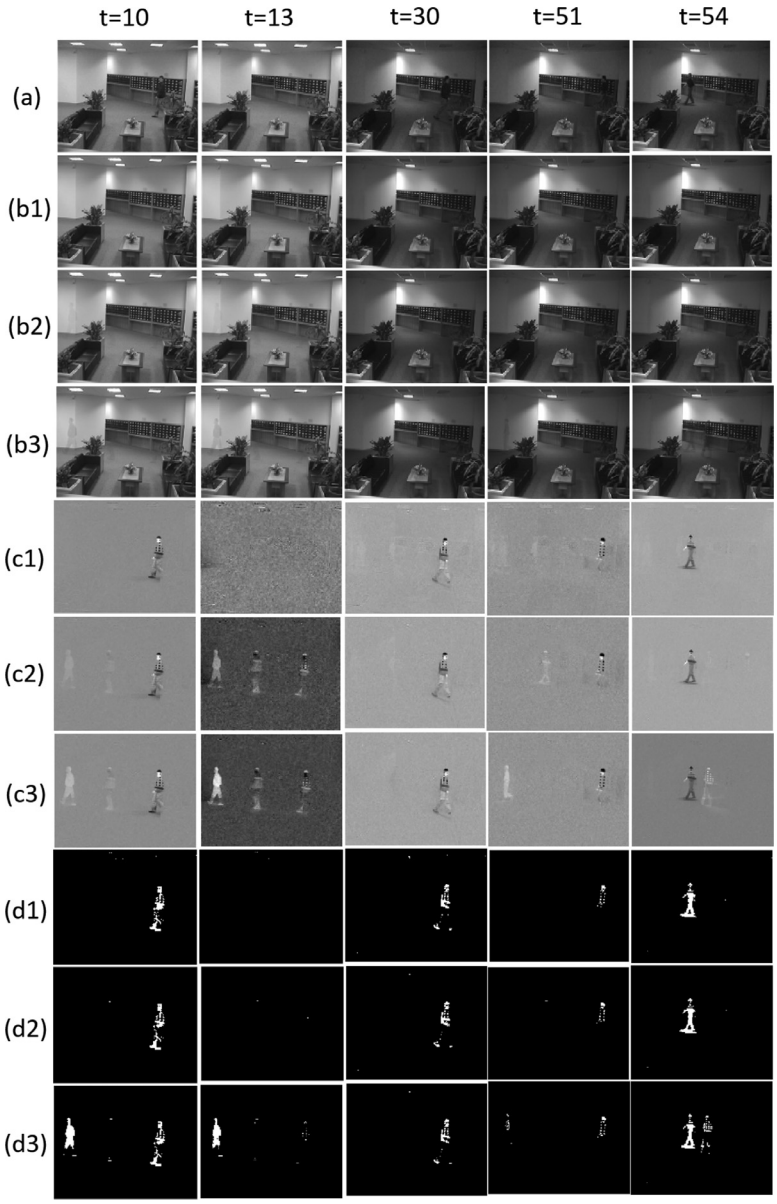

Fig. 2. The subset of *Lobby* sequence (80 frames): (a) Original frame of time slot $t = 10, 13, 30, 51,$ and 54; reconstructed background by (b1) proposed L_1-Tracking, (b2) L_1-IRW, and (b3) L_1-Batch; gray-scale extracted moving objects by (c1) proposed L_1-Tracking, (c2) L_1-IRW, and (c3) L_1-Batch; and binary mask by (d1) proposed L_1-Tracking, (d2) L_1-IRW, and (d3) L_1-Batch.

attacked randomly. We compare the performance of our proposed L_1-Tracking with L_1-Batch, L_1-IRW, GRASTA [27], PracReProCS [31], and OMoGMF [32]. The weight update parameter of the proposed L_1-Tracking is set as $\beta = 0.5$. For L_1-Batch and L_1-IRW, the 60 snapshots are divided into 6 groups of $N = 10$ snapshots, and an independent L_1-Batch or L_1-IRW subspace is computed for each group, followed by power estimation. For the proposed L_1-Tracking, we initialize the L_1-subspace and the associated binary bit matrix with the initial $N = 10$ snapshots. Then we keep the processing window size at $N = 10$, with every collected new snapshot $\mathbf{y}_t \in \mathbb{R}^M$, we update the L_1-subspace $\mathbf{P}_t^c \in \mathbb{R}^{M\times 2}$, $t = N+1, N+2, \ldots, N_{\text{total}}$. Correspondingly, $\widehat{\ell}_t = \mathbf{P}_t^c \mathbf{P}_t^{c\ T} \mathbf{y}_t$, $\widehat{\mathbf{x}}_t = \mathbf{H}^{\dagger}\widehat{\ell}_t = [\widehat{\Gamma}_t, \widehat{\alpha}_t]^T$. The recovered PU transmission power at time-slot t is $\widehat{\Gamma}_t$.

The power estimation error over a period of $N = 10$ time slots is calculated as the following:

$$\sigma_N = \|\widehat{\mathbf{\Gamma}} - \mathbf{\Gamma}\|_2 / \|\mathbf{\Gamma}\|_2, \tag{34}$$

Table 1
The average power estimation error and accumulated subspace tracking time of six algorithms in comparison.

	Proposed L_1-Tracking	L_1-IRW	L_1-Batch
Accumulated Time (s) ($N_{\text{total}} = 60$)	0.376	26.719	0.174
Average Power Estimation Error (σ_N^{ave})	**0.372**	**0.335**	0.793
	OMoGMF	PracReProCS	GRASTA
Accumulated Time (s) ($N_{\text{total}} = 60$)	0.054	0.656	0.019
Average Power Estimation Error (σ_N^{ave})	0.761	1.525	6.947

where $\widehat{\mathbf{\Gamma}}$ is the estimated PU transmission power (in dB). For all schemes in comparison, we calculate the average σ_N (denoted as σ_N^{ave}) with $N_{\text{total}} = 60$ snapshots and 100 experiments.

Table 1 shows the accumulated subspace update time measured in seconds and the average power estimation error for the

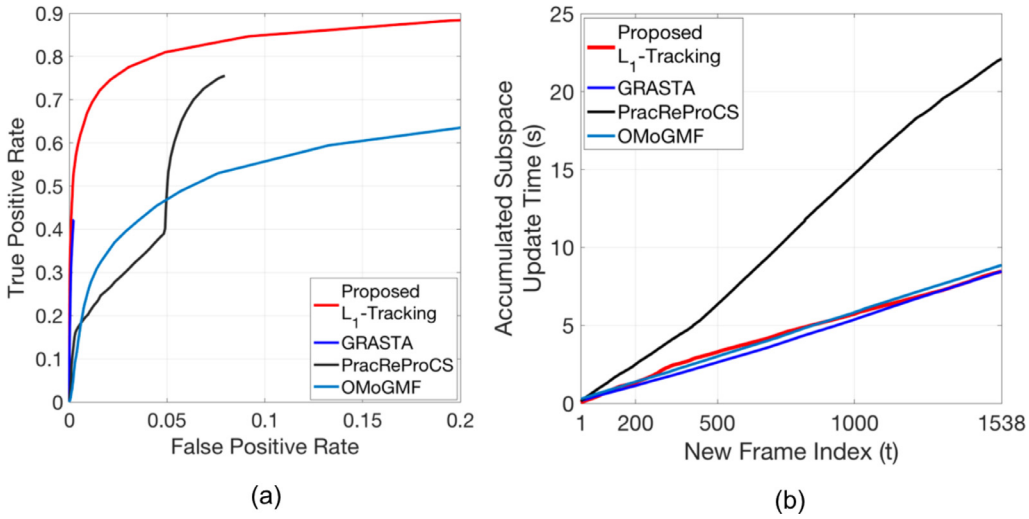

Fig. 3. Comparison studies of the proposed L_1-Tracking, the GRASTA [27], the PracReProCS [31], and the OMoGMF [32] algorithms on the complete *Lobby* sequence (1546 frames): (a) the ROC curves; (b) the accumulated subspace update time in seconds versus the new frame index t.

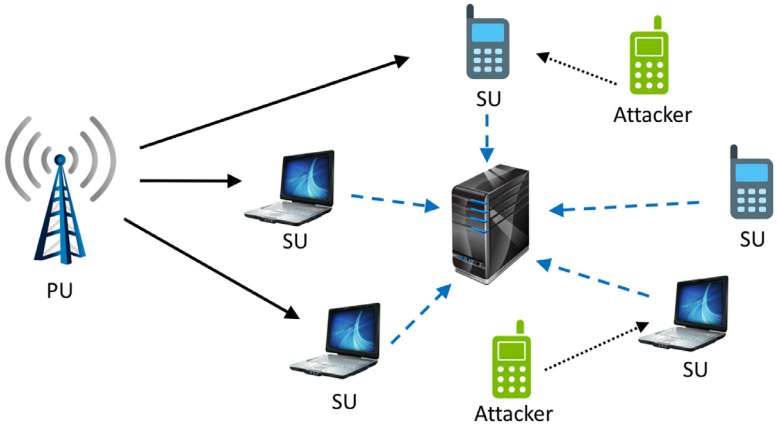

Fig. 4. Cooperative cognitive radio network structure.

six algorithms in comparison. The two lowest average power estimation error are highlighted in red, which are offered by the proposed L_1-Tracking and the L_1-IRW methods. Although L_1-IRW slightly outperforms L_1-Tracking in power estimation error, its huge processing time is inappropriate for real-time scenarios. On the other hand, although the GRASTA and OMoGMF algorithms excel in subspace update speed, their power estimation error resulted from inaccurate subspace estimation is much higher than the proposed L_1-Tracking scheme.

4.4. Direction-of-arrival tracking

A core technical problem in wireless communications and radar applications is the problem of estimating the direction-of-arrival (DoA) of incoming signals [38,39]. Our signal model is similar to those in [13] and [16]. We consider a receiver equipped with a uniform linear array (ULA) of M antenna elements, and d is the spacing between adjacent antenna elements. For an incoming far-field signal with angle-of-arrival $\theta_i \in (-\frac{\pi}{2}, \frac{\pi}{2}]$ and wavelength λ_c, the complex-domain array response vector is defined as

$$\mathbf{s}_{\theta_i} \triangleq \left[1,\ e^{-j\frac{2\pi d \sin\theta_i}{\lambda_c}}, \ldots, e^{-j\frac{(M-1)2\pi d \sin\theta_i}{\lambda_c}}\right]^T \in \mathbb{C}^M. \tag{35}$$

To satisfy the Nyquist spatial sampling theorem, d is chosen to be half the signal wavelength $d = \frac{1}{2}\lambda_c$. For simplicity, we define

$$f_i \triangleq 0.5\sin(-\theta_i), \tag{36}$$

then the array response vector becomes

$$\mathbf{s}_{f_i} = [1,\ e^{j1\cdot 2\pi f_i},\ e^{j2\cdot 2\pi f_i}, \ldots, e^{j(M-1)\cdot 2\pi f_i}]^T \in \mathbb{C}^M. \tag{37}$$

In our signal model, the ULA takes snapshots of two incoming signals (targets) with angles-of-arrival θ_1 and θ_2, and the associated f_1 and f_2 can be obtained by (36). The number of antenna elements is $M = 20$. The snapshot at time-slot t is expressed as

$$\mathbf{x}_t = A_1\mathbf{s}_{f_1} + A_2\mathbf{s}_{f_2} + \mathbf{n}_t, t = 1, 2, \ldots, \tag{38}$$

where A_1, A_2 are the received-signal amplitudes, and $\mathbf{n}_t \sim \mathcal{CN}(\mathbf{0}_M, \sigma^2\mathbf{I}_M)$ is additive white complex Gaussian noise. Therefore, the nominal signal lies in a rank-2 subspace formed by $\mathbf{s}_{f_1}$ and $\mathbf{s}_{f_2}$. We assume that the signal-to-noise ratio (SNR) of the two signals is $\text{SNR}_1 = 10\log_{10}\frac{A_1^2}{\sigma^2}\text{dB} = 4\text{dB}$ and $\text{SNR}_2 = 10\log_{10}\frac{A_2^2}{\sigma^2}\text{dB} = 5\text{dB}$. For the first ten snapshots $\mathbf{x}_t$, $t = 1, \ldots, 10$, f_1 and f_2 are fixed at 0.2 and 0.3, respectively, and $\mathbf{x}_5$ is corrupted by an interferer signal $\mathbf{x}_J = A_J\mathbf{s}_J$ with $f_J = 0.4$ and amplitude $A_J = A_2$, that is,

$$\mathbf{x}_5 = A_1\mathbf{s}_{f_1} + A_2\mathbf{s}_{f_2} + \mathbf{x}_J + \mathbf{n}_5. \tag{39}$$

Then, starting from $t = 11$, due to gradual change of θ_1 and θ_2, f_1 and f_2 become linearly time varying [40]. They start at 0.2 and 0.3, cross at 0.25, and finish at 0.3 and 0.2 over a span of 1000 snapshots. This causes the gradual change of the underlying rank-2 signal subspace. Besides, the same interferer signal $\mathbf{x}_J$ corrupts $\mathbf{x}_t$ with probability $p = 0.3$ for $t = 11, 12, \ldots, 1010$.

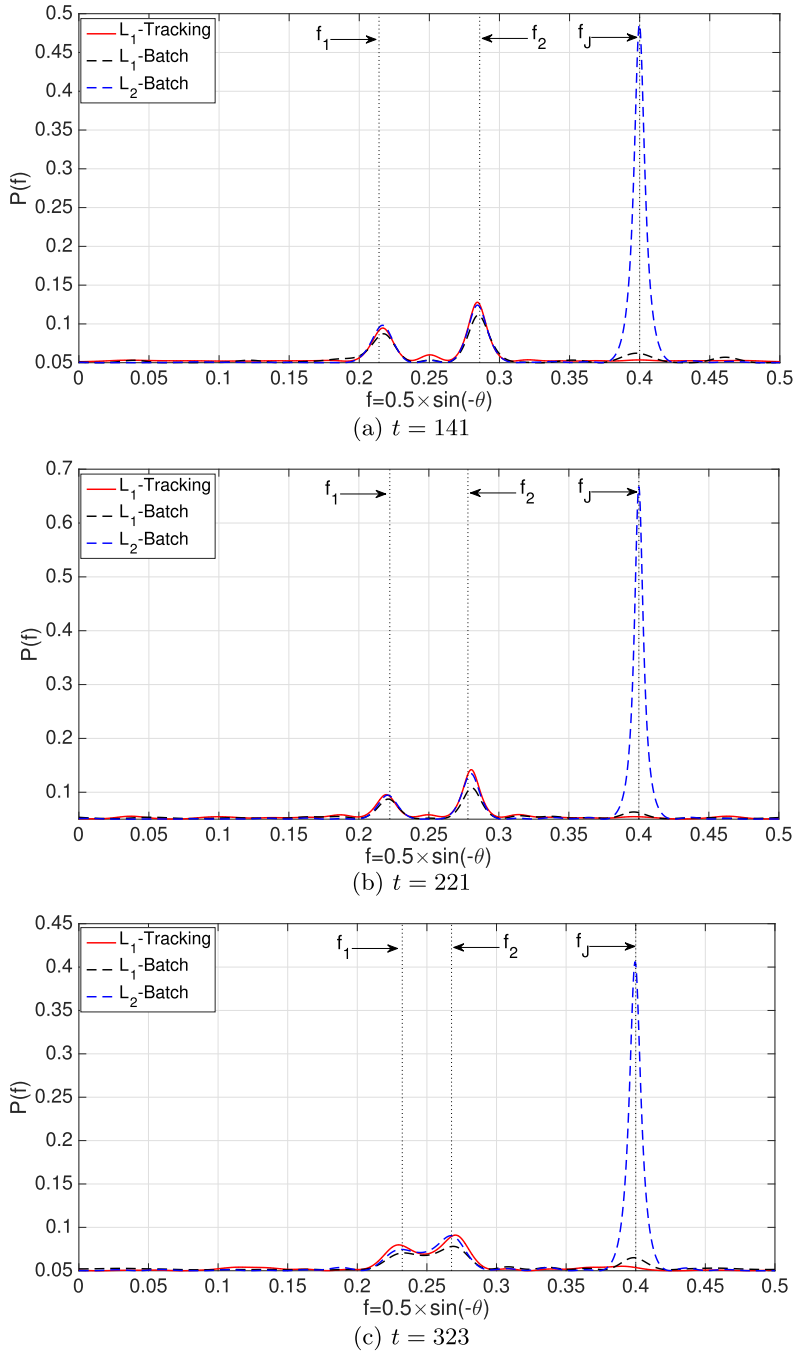

Fig. 5. MUSIC spectra with rank-2 subspaces at time-slot (a) $t = 141$, (b) $t = 221$, and (c) $t = 323$.

Our objective is to track the slowly changing rank-2 subspace formed by the two incoming signals, that is, to track the varying angles-of-arrival θ_1 and θ_2, or equivalently, to track the varying f_1 and f_2. For each snapshot, we create a real-valued version $\widetilde{\mathbf{x}}_t = [\mathrm{Re}\{\mathbf{x}_t\}; \mathrm{Im}\{\mathbf{x}_t\}] \in \mathbb{R}^{2M}$ by $\mathrm{Re}\{\cdot\}$, $\mathrm{Im}\{\cdot\}$ part concatenation. For our proposed L_1-Tracking, we initialize the rank-2 L_1-subspace by the initial $N = 10$ snapshots, using the L_1-IRW scheme. Subsequently, we update the L_1-subspace $\mathbf{P}_t^c \in \mathbb{R}^{2M\times 2}$ at the arrival of every two new snapshots.

We compared the proposed scheme with L_1-Batch and L_2-Batch. The processing window is fixed at $N = 10$ for all three schemes. For L_1-Batch and L_2-Batch, we re-calculate a new rank-2 subspace for every other time slot. At time slot t, a data matrix is formed by $\widetilde{\mathbf{X}}_t = [\widetilde{\mathbf{x}}_{t-N+1}, \widetilde{\mathbf{x}}_{t-N+2}, \ldots, \widetilde{\mathbf{x}}_t] \in \mathbb{R}^{2M\times N}$, on which the batch rank-2 L_1-PCA (9) and L_2-PCA (4) are performed to obtain $\mathbf{P}_{t,L_1} \in \mathbb{R}^{2M\times 2}$ and $\mathbf{P}_{t,L_2} \in \mathbb{R}^{2M\times 2}$, respectively. For performance evaluation, we plot for all three schemes the MUSIC spectrum [13]:

$$P(f) \triangleq \frac{1}{\widetilde{\mathbf{s}}_f^T(\mathbf{I}_{2M} - \mathbf{P}\mathbf{P}^T)\widetilde{\mathbf{s}}_f}, \tag{40}$$

where $\widetilde{\mathbf{s}}_f = [\mathrm{Re}\{\mathbf{s}_f\}; \mathrm{Im}\{\mathbf{s}_f\}] \in \mathbb{R}^{2M}$, $\mathbf{P} \in \mathbb{R}^{2M\times 2}$ is the learned rank-2 subspace, and $\mathbf{P} = \mathbf{P}_t^c$, $\mathbf{P} = \mathbf{P}_{t,L_1}$, $\mathbf{P} = \mathbf{P}_{t,L_2}$ for the proposed L_1-Tracking, L_1-Batch, and L_2-Batch, respectively. For successful DoA estimation schemes, the MUSIC spectrum shall show high peaks at nominal DoAs f_1 and f_2, and suppresses other signals.

In Fig. 5, we plot the MUSIC spectra for all three schemes at time-slot $t = 141, 221,$ and 323, respectively. The true f_1, f_2 and f_J are indicated by the vertical dotted lines in the figures. We

observe that as time elapses, L_1-Batch and the proposed L_1-Tracking algorithms are able to track the changing rank-2 subspaces and show peaks very close to the two nominal signals f_1 and f_2, while L_2-Batch MUSIC spectrum is severely contaminated by the interferer signal at f_J. Besides, the proposed L_1-Tracking outperforms L_1-Batch since it well suppresses the interferer at f_J, while L_1-Batch still shows a small peak at f_J. Further, the proposed L_1-Tracking accelerates the DoA tracking speed. In our experiment, the average subspace update time for the proposed L_1-Tracking algorithm is 0.0313 s per snapshot, and that for the L_1-Batch algorithm is 0.0514 s per snapshot.

5. Complexity analysis

In this section, we analyze the theoretical computational complexity in terms of multiplication operations for the proposed L_1-Tracking algorithm, the GRASTA, OMoGMF, and PracReProCS. Our findings are in accordance with the experimental results in Sections 4.2 and 4.3.

We assume that the data sample dimension is D, and the processing window size is N for the proposed L_1-Tracking. The complexity of the proposed rank-1 L_1-Tracking is analyzed in Table 2. At the arrival of the tth new datum $\mathbf{x}_t \in \mathbb{R}^D$, the major computational tasks to update the subspace $\mathbf{p}_t^c \in \mathbb{R}^D$ include: (1) Algorithm 1 Steps 5–7 that calculate the weight for the new datum, the complexity of which is $O(D+N)$; (2) Algorithm 2 Step 3 that executes bit flipping to update $\mathbf{b}_t^{(k)} \in \{\pm 1\}^N$, the complexity of which is $O(DN^2 \times \text{maxFlip})$, where maxFlip represents the number of bit flips for the BF procedure to converge; (3) Algorithm 2 Step 4 that re-calculates the subspace; and (4) Algorithm 2 Step 5 that updates the distance for N data samples. Let maxIter represent the number of sample-weight update iterations, then the total complexity of rank-1 subspace update is $O(DN^2 \times \text{maxFlip} \times \text{maxIter})$.

The complexity of the proposed rank-r ($r>1$) L_1-Tracking is analyzed in Table 3. The major computational tasks include: (1) Algorithm 3 Steps 5–7 that calculate the weight for the new datum and normalize the weights for all N data samples in the current processing window, with complexity $O(2Dr+D+N)$; (2) Algorithm 4 Step 1 that initializes the r bits associated with the new datum by exhaustive search over the 2^r-dimensional binary space, with complexity $O(2^r Dr^2)$; (3) Algorithm 4 Step 3 that executes BF to update the $N \times r$ bit matrix, with complexity $O(DNr^3 \times \text{maxFlip})$. Again maxFlip is the number of bit flips required for the BF to converge; (4) Algorithm 4 Step 4 that performs SVD with complexity $O(Dr^2)$; (5) Algorithm 4 Step 5 that updates the subspace with complexity $O(Dr^2)$; and (6) Algorithm 4 Step 6 that updates the distances for N data samples in the current processing window with complexity $O(DNr)$. Again, let maxIter be the number of iterations for sample weights to converge, then the total complexity for rank-r ($r>1$) case is $O((DNr^3 \times \text{maxFlip} + 2^r Dr^2) \times \text{maxIter})$.

Table 2
The computational complexity of the proposed rank-1 L_1-Tracking described in Algorithms 1 and 2.

Computational tasks	Complexity (Multiplications)
Algo. 1 Steps 5–7	$O(D+N)$
Algo. 2 Step 3	$O(DN^2 \times \text{maxFlip})$
Algo. 2 Step 4	$O(DN)$
Algo. 2 Step 5	$O(DN)$
Algo. 2 Step 6	$O(N)$
Algo. 2 Step 7	$O(N)$
Algo. 2 Step 8	$O(N)$
Total	$O(DN^2 \times \text{maxFlip} \times \text{maxIter})$

Table 3
The computational complexity of the proposed rank-r ($r>1$) L_1-Tracking described in Algorithms 3 and 4.

Computational tasks	Complexity (Multiplications)
Algo. 3 Steps 5–7	$O(2Dr+D+N)$
Algo. 4 Step 1	$O(2^r Dr^2)$
Algo. 4 Step 3	$O(DNr^3 \times \text{maxFlip})$
Algo. 4 Step 4	$O(Dr^2)$
Algo. 4 Step 5	$O(Dr^2)$
Algo. 4 Step 6	$O(DNr)$
Algo. 4 Step 7	$O(N)$
Algo. 4 Step 8	$O(N)$
Algo. 4 Step 9	$O(N)$
Total	$O\left(\left(DNr^3 \times \text{maxFlip} + 2^r Dr^2\right) \times \text{maxIter}\right)$

Empirically, for both rank-1 and rank-r, maxFlip <10 or even equals to 1 with large chances. This is because for rank-1, the BF in the kth iteration is initialized with $\mathbf{b}_t^{(k-1)}$, the bit vector in the $(k-1)$th weight-update iteration; while for rank-r, the BF in the kth iteration is initialized with the bit matrix $\mathbf{B}_t^{(k-1)}$ in the $(k-1)$th weight-update iteration. Such "warm-start" technique significantly accelerates the convergence of the BF procedure.

The GRASTA [27] minimizes a cost function that has an ℓ_1-norm penalty on the sparse outliers. Then the subspace tracking is formulated as minimizing an augmented Lagrangian function, which is solved by alternating between solving for four variables: the rank-r subspace coefficient of length r, the sparse outlier vector of length D, the Lagrange multiplier of length D, and the columns of a $D \times r$ matrix that span the rank-r subspace. The complexity is in the order of $O(r^3+Dr)$, where $O(r^3)$ is the complexity of the inversion of an $r \times r$ matrix in solving for the subspace coefficients, and $O(Dr)$ is the complexity of a matrix-vector multiplication involved in updating all four variables.

In PracReProCS [31], the subspace tracking includes four steps: 1) perpendicular projection of the new datum onto the space orthogonal to the previously estimated rank-r subspace, with complexity $O(D^2)$; 2) sparse outlier vector recovery by ℓ_1-norm minimization, with complexity $O(D^3)$; 3) low-rank component recovery with subtraction operations only; and 4) subspace update by the method of projection PCA, which involves an SVD of complexity $O(DN\min\{D, N\})$, in which the most recent N data samples are utilized.

The OMoGMF [32] deals with the background subtraction problem in video surveillance by modeling the video background as a low-rank component and performs low-rank matrix factorization. More importantly, it models the foreground as a mixture of Gaussians (MoG). The online low-rank subspace learning problem is then tackled by iteratively solving for the MoG parameters, the subspace coefficients, and the subspace. The MoG parameters are solved by the EM algorithm, in which the E-step is of complexity $O(D(r+K))$ where D is the dimension of the datum (a video frame in [32]), r is the subspace rank, and K is the number of components in the MoG model, and the M-step is of complexity $O(DK)$. The subspace coefficients are of size $r \times 1$ for a datum and it is solved by a least squares problem with complexity $O(Dr+r^3)$. Finally, updating the subspace of dimension $D \times r$ has complexity $O(Dr^2)$.

We compare the computational complexity in terms of multiplication operations for the proposed L_1-Tracking, GRASTA, OMoGMF, and PracReProCS algorithms in Table 4. For OMoGMF, "Iter" refers to the number of iterations for the EM algorithm to calculate the MoG parameters. In practice, the rank value is usually $r \ll \min\{D, N\}$. Besides, for the proposed L_1-Tracking, we adopt a small processing-window size N for lower complexity and faster convergence rate according to the synthetic data experiment in Section 4.1, and we use a medium β value for sample-weight update to control maxIter. We also consider the fact that maxFlip <10 or equals to 1 most of the time. With these conditions, it is observed from Table 4 that GRASTA has the lowest complexity,

Table 4
Computational complexity comparison among GRASTA, OMoGMF, the proposed L_1-Tracking, and PracReProCS.

GRASTA	OMoGMF	Proposed L_1-Tracking	PracReProCS
$O(Dr + r^3)$	$O\big((Dr + DK) \times \text{Iter} + Dr^2 + r^3\big)$	$r = 1$: $O(DN^2 \times \text{maxFlip} \times \text{maxIter})$ $r > 1$: $O\big((DNr^3 \times \text{maxFlip} + 2^r Dr^2) \times \text{maxIter}\big)$	$O(D^3 + D^2 + DN \min\{D, N\})$

PracReProCS has the highest complexity due to the ℓ_1-norm minimization adopted to solve for sparse outliers, while our proposed L_1-Tracking and the OMoGMF algorithms have medium complexity. In our experimental studies in Sections 4.2 (video surveillance) and 4.3 (cognitive radio network transmission power estimation), the measured execution time for subspace tracking is in accordance with the complexity analysis in Table 4.

6. Conclusion

In this work, we propose a novel online robust subspace tracking algorithm "L_1-Tracking" based on the L_1-norm principal-component analysis theory. The algorithm effectively captures the intrinsic low-rank structure of streaming data in the presence of observation outliers. It updates the subspace at each time slot with new sensor datum, utilizing the subspace obtained at the previous time slot and a small batch of most recent data samples. It has the merits of data outlier suppression through sample weighting and speed acceleration through a warm-start bit-flipping technique.

The experimental studies on various applications illustrated the superior performance of the proposed algorithm in subspace estimation accuracy. Besides, the theoretical analysis and experimental results demonstrated that the computational complexity of the proposed algorithm is comparable to several state-of-the-art online subspace learning algorithms. Meanwhile, it significantly reduces the processing time compared to the existing iterative re-weighted L_1-subspace (L_1-IRW) calculation. Hence, the proposed method is amenable to streaming and real-time applications.

In terms of future work, it is of particular interest to further investigate the capability of the proposed L_1-Tracking algorithm to process large data set online, such as real-time camera data that is of high dimensionality and has high frame rate. Our experimental study on the Lobby video sequence already illustrates such potential, and it is possible to explore such potential in other fields such as large-scale IoT networks. Besides, to accelerate the subspace tracking speed for high-dimensional streaming data, it is significant to investigate the sub-sampling technique. We will also develop schemes to automatically select proper model parameters, such as the rank value, processing-window size, and the weight-update parameter. Further, currently there is a lack of theoretical analysis on how close the estimated subspace in the proposed algorithm is to the true low-rank subspace of the data. In the future research, we will try to establish a theoretical bound for the subspace estimation error defined in (29).

References

[1] Q. Ke, T. Kanade, Robust Subspace Computation Using L_1 Norm, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2003.
[2] Q. Ke, T. Kanade, Robust L_1 norm factorization in the presence of outliers and missing data by alternative convex programming, in: Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2005, pp. 739–746.
[3] A. Eriksson, A. Van Den Hengel, Efficient computation of robust low-rank matrix approximations in the presence of missing data using the L_1 norm, in: Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2010, pp. 771–778.
[4] L. Yu, M. Zhang, C. Ding, An efficient algorithm for L_1-norm principal component analysis, in: Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), 2012, pp. 1377–1380.
[5] E.J. Candès, X. Li, Y. Ma, J. Wright, Robust principal component analysis? J. ACM (JACM) 58 (3) (2011) Article 11.
[6] X. Zhou, C. Yang, W. Yu, Moving object detection by detecting contiguous outliers in the low-rank representation, IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) 35 (3) (2013) 597–610.
[7] Y.W. Park, D. Klabjan, Iteratively reweighted least squares algorithms for L_1-norm principal component analysis, in: Proc. IEEE 16th Int. Conf. Data Mining (ICDM), 2016, pp. 430–438.
[8] M. Yin, J. Gao, Z. Lin, Laplacian regularized low-rank representation and its applications, IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) 38 (3) (2016) 504–517.
[9] Q. Wang, Q. Gao, X. Gao, F. Nie, Optimal mean two-dimensional principal component analysis with f-norm minimization, J. Pattern Recognit. 68 (2017) 286–294.
[10] S. Yi, Z. Lai, Z. He, Y. Cheung, Y. Liu, Joint sparse principal component analysis, J. Pattern Recognit. 61 (2017) 524–536.
[11] N. Kwak, Principal component analysis based on L_1-norm maximization, IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) 30 (9) (2008) 1672–1680.
[12] F. Nie, H. Huang, C. Ding, D. Luo, H. Wang, Robust principal component analysis with non-greedy L_1-norm maximization, in: Proc. Twenty-Second Int. Joint Conf. Artificial Intelligence, 2011, pp. 1433–1438.
[13] P.P. Markopoulos, G.N. Karystinos, D.A. Pados, Optimal algorithms for L_1-subspace signal processing, IEEE Trans. Signal Process. 62 (19) (2014) 5046–5058.
[14] S. Kundu, P.P. Markopoulos, D.A. Pados, Fast computation of the L_1-principal component of real-valued data, in: Proc. IEEE Int. Conf. Acoustics, Speech and Signal Process. (ICASSP), 2014, pp. 8028–8032.
[15] P.P. Markopoulos, S. Kundu, S. Chamadia, D.A. Pados, Efficient L_1-norm principal-component analysis via bit flipping, IEEE Trans. Signal Process. 65 (16) (2017) 4252–4264.
[16] P.P. Markopoulos, N. Tsagkarakis, D.A. Pados, G.N. Karystinos, Direction finding with L_1-norm subspaces, in: Proc. SPIE Sensing Technology + Applications, Compressive Sensing III, vol. 9109, 2014, p. 91090J.
[17] F. Maritato, Y. Liu, S. Colonnese, D.A. Pados, Cloud-assisted individual L_1-pca face recognition using wavelet-domain compressed images, in: Proc. The 6th European Workshop on Visual Information Processing (EUVIP), 2016, pp. 1–6.
[18] Y. Liu, D.A. Pados, Compressed-sensed-domain L_1-pca video surveillance, IEEE Trans. Multimed. 18 (3) (2016) 351–363.
[19] Y. Liu, D.A. Pados, S.N. Batalama, M.J. Medley, Iterative re-weighted l_1-norm principal-component analysis, in: The 51st Asilomar Conference on Signals, Systems, and Computers, 2017, pp. 425–429.
[20] Y. Li, On incremental and robust subspace learning, J. Pattern Recognit. 37 (7) (2004) 1509–1518.
[21] S. S. Bucak, B. Gunsel, Incremental subspace learning via non-negative matrix factorization, J. Pattern Recognit. 42 (5) (2009) 788–797.
[22] J. Feng, H. Xu, S. Yan, Online robust PCA via stochastic optimization, in: Proc. Advances in Neural Info. Process. Syst. (NIPS), 2013, pp. 404–412.
[23] J. Feng, H. Xu, S. Mannor, S. Yan, Online PCA for contaminated data, in: Proc. Advances in Neural Info. Process. Syst. (NIPS), 2013, pp. 404–412.
[24] M. Mardani, G. Mateos, G. B. Giannakis, Dynamic anomalography: tracking network anomalies via sparsity and low rank, IEEE J. Sel. Top. Signal Process. 7 (1) (2013) 50–66.
[25] S. Chouvardas, Y. Kopsinis, S. Theodoridis, An adaptive projected subgradient based algorithm for robust subspace tracking, in: Proc. IEEE Int. Conf. Acoust. Speech, Signal Process. (ICASSP), 2014, pp. 5497–5501.
[26] S. Hauberg, A. Feragen, R. Enficiaud, M.J. Black, Scalable robust principal component analysis using Grassmann averages, IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) 38 (11) (2016) 2298–2311.
[27] J. He, L. Balzano, A. Szlam, Incremental gradient on the Grassmannian for online foreground and background separation in subsampled video, in: IEEE Conf. on Comp. Vis. Pat. Rec. (CVPR), 2012, pp. 1568–1575.
[28] H. Mansour, X. Jiang, A robust online subspace estimation and tracking algorithm, in: Proc. IEEE Int. Conf. Acoustics, Speech and Signal Process. (ICASSP), 2015, pp. 4065–4069.
[29] C. Qiu, N. Vaswani, Real-time robust principal components' pursuit, in: Proc. The 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2010, pp. 591–598.
[30] C. Qiu, N. Vaswani, Recursive sparse recovery in large but correlated noise, in: Proc. The 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2011, pp. 752–759.
[31] H. Guo, C. Qiu, N. Vaswani, An online algorithm for separating sparse and low-dimensional signal sequences from their sum, IEEE Trans. Signal Process. 62 (16) (2014) 4284–4297.
[32] H. Yong, D. Meng, W. Zuo, L. Zhang, Robust online matrix factorization for dynamic background subtraction, IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) 40 (7) (2018) 1726–1740.

[33] X.J. Hunt, R. Willett, Online data thinning via multi-subspace tracking, IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) 41 (5) (2019) 1173–1187.
[34] B. Fan, X. Li, Y. Cong, Y. Tang, Structured and weighted multi-task low rank tracker, J. Pattern Recognit. 81 (2018) 528–544.
[35] I.F. Akyildiz, W. Lee, M.C. Vuran, S. Mohanty, A survey on spectrum management in cognitive radio networks, IEEE Commun. Mag. 46 (4) (2008) 40–48.
[36] A.W. Min, K. Kim, K.G. Shin, Robust cooperative sensing via state estimation in cognitive radio networks, in: Proc. IEEE Int. Symp. Dynamic Spectrum Access Networks (DySPAN), 2011, pp. 185–196.
[37] F. Lin, Z. Hu, S. Hou, J. Yu, C. Zhang, N. Guo, M. Wicks, R.C. Qiu, K. Currie, Cognitive radio network as wireless sensor network (ii): security consideration, in: Proc. IEEE National Aerospace and Electronics Conf. (NAECON), 2011, pp. 324–328.
[38] H. Krim, M. Viberg, Two decades of array signal processing research: the parametric approach, IEEE Signal Process. Mag. 13 (4) (1996) 67–94.
[39] L.C. Godara, Application of antenna arrays to mobile communications. ii. beam-forming and direction-of-arrival considerations, Proc. IEEE 85 (8) (1997) 1195–1245.
[40] B. Yang, Projection approximation subspace tracking, IEEE Trans. Signal Process. 43 (1) (1995) 95–107.

Ying Liu received the B.S. degree in communications engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2006, the M.S. and Ph.D. degrees in Electrical Engineering from The State University of New York at Buffalo, Buffalo, NY, USA, in 2008 and 2012, respectively. She currently is an Assistant Professor in the Department of Computer and Science Engineering at Santa Clara University, Santa Clara, CA, USA. Her general areas of expertise are computer vision, machine learning, and signal processing.

Konstantinos Tountas received the Diploma and M.Sc. degrees in electronic and computer engineering from the Technical University of Crete, Chania, Greece, in 2014 and 2016, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL, USA. He was with the Telecom Lab, Technical University of Crete. His research interests span the areas of signal processing and localization, software defined wireless communications, and underwater acoustic communications.

Dimitris A. Pados received the Diploma degree in computer science and engineering (five-year program) from the University of Patras, Greece, in 1989, and the Ph.D. degree in electrical engineering from the University of Virginia, Charlottesville, VA, in 1994. Dr. Pados is a Professor, the I-SENSE Fellow, and the Charles E. Schmidt Eminent Scholar in Engineering in the Department of Computer and Electrical Engineering and Computer Science at Florida Atlantic University, Boca Raton, FL. He currently leads the University Initiative on Autonomous Systems and is the Director of the ExtremeComms Laboratory. His basic research interests are in the general areas of data and signal processing and communications theory.

Stella N. Batalama serves as the Dean of the College of Engineering and Computer Science at Florida Atlantic University since August 2017. She served as the Chair of the Electrical Engineering Department, University at Buffalo, The State University of New York, from 2010 to 2017 and as the Associate Dean for Research of the School of Engineering and Applied Sciences from 2009 to 2011. From 2003 to 2004, she was the Acting Director of the AFRL Center for Integrated Transmission and Exploitation, Rome NY, USA. Her research interests include cognitive and cooperative communications and networks, multimedia security and data hiding, underwater signal processing, communications and networks. She has published over 180 papers in scientific journals and conference proceedings in her research field. She was a recipient of the 2015 SUNY Chancellor's Award for Excellence in Research. She was an Associate Editor for the IEEE Communications Letters (2000–2005) and the IEEE Transactions on Communications (2002–2008). Dr. Batalama is a senior member of the Institute of Electrical and Electronics Engineering (IEEE), a member of the Society of Women Engineers, and a member of the American Society for Engineering Education. Dr. Batalama received her Ph.D. in electrical engineering from the University of Virginia and her undergraduate and graduate degrees in computer science and engineering from the University of Patras in Greece. She also completed the Program for Leadership Development at Harvard Business School.

Michael J. Medley received the B.S., M.S. and Ph.D. degrees in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1990, 1991 and 1995, respectively. He is a principal research engineer in airborne network communications with the United States Air Force Research Laboratory in Rome, NY, with research activities related to adaptive interference suppression, spread spectrum waveform design, spectrum management, covert messaging, and terahertz network communications. He also serves as Associate Professor of Electrical and Computer Engineering at the State University of New York Polytechnic Institute in Utica, NY.