A Subspace Learning Approach for High Dimensional Matrix Decomposition with Efficient Column/Row Sampling

Mostafa Rahmani MOSTAFA@KNIGHTS.UCF.EDU
George Atia GEORGE.ATIA@UCF.EDU
University of Central Florida, 4000 Central Florida Blvd. Orlando, Florida, 32816

Abstract

This paper presents a new randomized approach to high-dimensional low rank (LR) plus sparse matrix decomposition. For a data matrix $\mathbf{D} \in \mathbb{R}^{N_1 \times N_2}$, the complexity of conventional decomposition methods is $\mathcal{O}(N_1 N_2 r)$, which limits their usefulness in big data settings (r is the rank of the LR component). In addition, the existing randomized approaches rely for the most part on uniform random sampling, which may be inefficient for many real world data matrices. The proposed subspace learning-based approach recovers the LR component using only a small subset of the columns/rows of data and reduces complexity to $\mathcal{O}(\max(N_1, N_2) r^2)$. Even when the columns/rows are sampled uniformly at random, the sufficient number of sampled columns/rows is shown to be roughly $\mathcal{O}(r\mu)$, where μ is the coherency parameter of the LR component. In addition, efficient sampling algorithms are proposed to address the problem of column/row sampling from structured data.

1. Introduction

Suppose we are given a data matrix $\mathbf{D} \in \mathbb{R}^{N_1 \times N_2}$, which can be expressed as

$$\mathbf{D} = \mathbf{L} + \mathbf{S}, \tag{1}$$

where $\mathbf{L}$ is a low rank (LR) matrix and $\mathbf{S}$ is a sparse matrix with arbitrary unknown support. In many machine learning and data analysis applications, the given data can be naturally modeled using (1). It was shown in (Chandrasekaran et al., 2011) that if the column space (CS) and row space (RS) of $\mathbf{L}$ are sufficiently incoherent with the standard basis and the non-zero elements of $\mathbf{S}$ are sufficiently diffused, the convex program

$$\begin{aligned} \min_{\dot{\mathbf{L}}, \dot{\mathbf{S}}} \quad & \lambda \|\dot{\mathbf{S}}\|_1 + \|\dot{\mathbf{L}}\|_* \\ \text{subject to} \quad & \dot{\mathbf{L}} + \dot{\mathbf{S}} = \mathbf{D} \end{aligned} \tag{2}$$

yields the exact decomposition, where $\|.\|_1$ is the ℓ_1-norm and $\|.\|_*$ is the nuclear norm.

For the LR matrix $\mathbf{L}$ with rank r and compact SVD $\mathbf{L} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ (where $\mathbf{U} \in \mathbb{R}^{N_1 \times r}$, $\mathbf{\Sigma} \in \mathbb{R}^{r \times r}$ and $\mathbf{V} \in \mathbb{R}^{N_2 \times r}$), the incoherence condition is typically defined through the requirements (Candès et al., 2011; Chandrasekaran et al., 2011)

$$\begin{aligned} & \max_i \|\mathbf{U}^T \mathbf{e}_i\|_2^2 \leq \frac{\mu r}{N_1} \ , \ \max_i \|\mathbf{V}^T \mathbf{e}_i\|_2^2 \leq \frac{\mu r}{N_2} \\ & \text{and } \ \|\mathbf{U}\mathbf{V}^T\|_\infty \leq \sqrt{\frac{\mu r}{N_2 N_1}} \end{aligned} \tag{3}$$

for some parameter μ that bounds the projection of the standard basis $\{\mathbf{e}_i\}$ onto the column and row subspaces. Other useful measures for the coherency of subspaces are given in (Candès & Romberg, 2007) as,

$$\gamma(\mathbf{U}) = \sqrt{N_1} \max_{i,j} |\mathbf{U}(i,j)|, \gamma(\mathbf{V}) = \sqrt{N_2} \max_{i,j} |\mathbf{V}(i,j)|, \tag{4}$$

where $\gamma(\mathbf{U})$ and $\gamma(\mathbf{V})$ bound the coherency of the column space and the row space, respectively. It is not hard to show that $\max(\gamma(\mathbf{V}), \gamma(\mathbf{U})) \leq \sqrt{\mu}$.

In (Candès et al., 2011), the sparsity pattern of the sparse matrix is selected uniformly at random to ensure that the sparse matrix is not a LR matrix (the non-zero elements are sufficiently diffused) with overwhelming probability. In this model, termed the Bernoulli model, each element of the sparse matrix can be non-zero independently with a constant probability. In this paper, we also use the Bernoulli model for the sparsity pattern of the sparse matrix.

The optimization problem (2) is convex. Similar to the iterative shrinking algorithms for ℓ_1-norm and nuclear norm minimization, a family of iterative algorithms for solving

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

(2) were proposed (Lin et al., 2010; Yuan & Yang, 2009), albeit they have complexity $\mathcal{O}(N_1 N_2 r)$ per iteration and require saving the entire (big) data in the working memory.

Since the LR and the sparse matrices are low-dimension geometrical structures, the robust principal component analysis (PCA) and matrix decomposition problems can be conceivably solved using small data sketches, i.e., a small set of random observations of the data (Halko et al., 2011; Li & Haupt, 2015; Mackey et al., 2011a; Rahmani & Atia, 2015a;b; Wright et al., 2013). In (Wright et al., 2013), it was shown that the LR and the sparse components can be precisely recovered using a small set of random linear measurements of $\mathbf{D}$. A convex program was proposed in (Wright et al., 2013) to recover these components using random matrix embedding with a polylogarithmic penalty factor in sample complexity, albeit the formulation requires solving a high-dimensional optimization problem.

The divide-and-conquer approach proposed in (Mackey et al., 2011b) can achieve super-linear speedups over full-scale matrix decomposition. This approach forms an estimate of $\mathbf{L}$ by combining two low-rank approximations obtained from submatrices formed from sampled rows and columns of $\mathbf{D}$ using the generalized Nyström method. Our approach also achieves super-linear speedups in decomposition, yet is fundamentally different from (Mackey et al., 2011b) and offers several advantages for the following reasons. First, our approach is a *subspace-pursuit approach* that focuses on subspace learning in a structure-preserving data sketch. Once the column space is learned, each column of the data is decomposed independently using a proposed randomized vector decomposition algorithm. Second, unlike (Mackey et al., 2011b), which is a batch approach that requires to store the entire data, the structure of the proposed approach naturally lends itself to online implementation. Third, while the analysis provided in (Mackey et al., 2011b) requires roughly $\mathcal{O}(r^2\mu^2 \max(N_1, N_2))$ random observations to ensure exact decomposition with high probability (whp), we show that the order of sufficient number of random observations depends linearly on the rank and the coherency parameter even if uniform random sampling is used. Fourth, the structure of the proposed approach allows us to leverage efficient sampling methods for challenging real world scenarios in which the columns and rows of $\mathbf{L}$ are not uniformly distributed in their respective subspaces, or when the data exhibits additional structures (e.g. clustering structures) (c.f. Sections 3.2, 3.3). In such settings, the uniform random sampling used in (Mackey et al., 2011b) requires significantly larger amounts of data to carry out the decomposition.

2. Structure of the Proposed Approach

Let us rewrite (1) as $\mathbf{D} = \mathbf{U}\mathbf{Q} + \mathbf{S}$, where $\mathbf{Q} = \mathbf{\Sigma}\mathbf{V}$. We call $\mathbf{Q} \in \mathbb{R}^{r \times N_2}$ the representation matrix which contains the expansion of the columns of $\mathbf{L}$ in the basis $\mathbf{U}$. Let $\mathbf{D}_{s1} \in \mathbb{R}^{N_1 \times m_1}$ denote a matrix consisting of m_1 columns sampled from $\mathbf{D}$, and $\mathbf{D}_{s2} \in \mathbb{R}^{m_2 \times N_2}$ a matrix of m_2 sampled rows. The table of Algorithm 1 presents the structure of the proposed method. The main idea is to exploit the low dimensionality of $\mathrm{span}(\mathbf{U})$ to learn the CS of $\mathbf{L}$ using a small subset of the columns of $\mathbf{D}$ (matrix $\mathbf{D}_{s1}$). Once the CS of $\mathbf{L}$ is obtained, each column can be viewed as a summation of an unknown vector from a known subspace and an unknown sparse vector. The representation matrix is then obtained using a small subset of the rows of $\mathbf{D}$ (matrix $\mathbf{D}_{s2}$).

Algorithm 1 Structure of Proposed Approach

Input: Data matrix $\mathbf{D} \in \mathbb{R}^{N_1 \times N_2}$
1. CS Learning
1.1 Column sampling: Matrix $\mathbf{D}_{s1} \in \mathbb{R}^{N_1 \times m_1}$ contains m_1 sampled columns of $\mathbf{D}$.
1.2 Decompose $\mathbf{D}_{s1}$ via

$$\begin{aligned} \min_{\dot{\mathbf{L}}_{s1}, \dot{\mathbf{S}}_{s1}} \quad & \frac{1}{\sqrt{N_1}} \|\dot{\mathbf{S}}_{s1}\|_1 + \|\dot{\mathbf{L}}_{s1}\|_* \\ \text{subject to} \quad & \dot{\mathbf{L}}_{s1} + \dot{\mathbf{S}}_{s1} = \mathbf{D}_{s1}, \end{aligned} \tag{5}$$

and define $\hat{\mathbf{L}}_{s1}$ and $\hat{\mathbf{S}}_{s1}$ as the optimal point of (5).
1.3 CS calculation: Matrix $\hat{\mathbf{U}}$ is obtained as an orthonormal basis for the CS of $\hat{\mathbf{L}}_{s1}$.
2. Representation Matrix Learning
2.1 Row sampling: Matrix $\mathbf{D}_{s2} \in \mathbb{R}^{m_2 \times N_2}$ contains m_2 sampled rows of $\mathbf{D}$. Define $\mathbf{S}_2 \in \mathbb{R}^{N_1 \times m_2}$ as the row sampling matrix, $\mathbf{D}_{s2} = \mathbf{S}_2^T \mathbf{D}$.
2.2 Representation learning: The representation matrix with respect to $\hat{\mathbf{U}}$ is obtained as the optimal point of

$$\min_{\dot{\mathbf{Q}}} \|\mathbf{D}_{s2} - \mathbf{S}_2^T \hat{\mathbf{U}} \dot{\mathbf{Q}}\|_1 \,. \tag{6}$$

Output: If $\hat{\mathbf{Q}}$ is the optimal point of (9), $\hat{\mathbf{L}} = \hat{\mathbf{U}}\hat{\mathbf{Q}}$ is the obtained LR matrix.

The following theorem establishes that the sufficient values for m_1 and m_2 scale linearly with the rank. To simplify the analysis, it is assumed that the CS of the LR matrix is sampled from the random orthogonal model (Candès & Recht, 2009), i.e., the columns of $\mathbf{U}$ are selected uniformly at random among all families of r-orthonormal vectors. Due to space limitations, the proof of Theorem 1 is deferred to an extended version of this paper (Rahmani & Atia, 2015b).

Theorem 1. *Suppose the CS of the LR matrix is sampled from the random orthogonal model and the support set of* $\mathbf{S}$ *follows the Bernoulli model with parameter* ρ*. In addition, it is assumed that Algorithm 1 samples the columns and*

rows uniformly at random. If for any small $\delta > 0$,

$$
\begin{aligned}
m_1 &\geq \max\left(r\gamma^2(\mathbf{V})\max\left(c_2\log r, c_3\log\frac{3}{\delta}\right),\right.\\
&\qquad\left.\frac{r}{\rho_r}\,\mu^{'}(\log N_1)^2\right),\\
m_2 &\geq \max\left(r\log N_1\max\left(c_2^{'}\log r, c_3^{'}\log\frac{3}{\delta}\right),\right.\\
&\qquad\frac{2r\beta(\beta-2)\log\left(\frac{N_2}{\delta}\right)}{3(\beta-1)^2}\left(c_6\kappa\log\frac{N_1N_2}{\delta}+1\right),\\
&\qquad\left.c_5(\log\frac{N_1N_2}{\delta})^2, \sqrt[7]{\frac{3}{\delta}}\right)\\
\rho &\leq \min\left(\rho_s, \frac{0.5}{r\beta\left(c_6\kappa\log\frac{N_1N_2}{\delta}+1\right)}\right)
\end{aligned}
\tag{7}
$$

where

$$
\begin{aligned}
\mu^{'} &= \max\left(\frac{c_7\max(r,\log N_1)}{r}, 6\gamma^2(\mathbf{V}),\right.\\
&\left.(c_9\gamma(\mathbf{V})\log N_1)^2\right), \quad \kappa = \frac{\log(N_1)}{r},
\end{aligned}
\tag{8}
$$

$\{c_i\}_{i=1}^{9}$, ρ_r, ρ_s, $c_2^{'}$ and $c_3^{'}$ are constant numbers and β any real number greater that one, then the proposed approach (Algorithm 1) yields the exact decomposition with probability at least $(1-5\delta-c_8N_1^{-3})$.

Theorem 1 confirms the intuition that the sufficient number of observations is linear with the rank and coherency parameter. We note that we have used a random model for the CS of $\mathbf{L}$. The analysis in (Mackey et al., 2011b) – which requires $\mathcal{O}(r^2\mu^2)$ randomly samples columns/rows – does not assume a random model for the LR component. The proposed approach has $\mathcal{O}(\max(N_1,N_2)\times\max(m_1,m_2)\times r)$ per-iteration running time complexity, which is significantly lower than $\mathcal{O}(N_1N_2r)$ per iteration for full scale decomposition (2) (Lin et al., 2010) implying remarkable speedups for big data because the sufficient values for m_1 and m_2 scale linearly with r. For instance, consider $\mathbf{U}$ and $\mathbf{Q}$ sampled from $\mathcal{N}(0,1)$, $r = 5$, and $\mathbf{S}$ following the Bernoulli model with $\rho = 0.02$. For values of $N_1 = N_2$ equal to 500, 1000, 5000, 10^4 and 2×10^4, if $m_1 = m_2 = 10r$, the proposed approach yields the correct decomposition with 90, 300, 680, 1520 and 4800 - fold speedup, respectively, over directly solving (2). In addition, the randomized method merely needs to save a small set of the columns and rows of the data in the working memory.

Once the CS of $\mathbf{L}$ is learned (i.e., $\hat{\mathbf{U}}$ is obtained), *each column* can be decomposed independently. Specifically, for $\mathbf{d}_i$ (the i^{th} column of $\mathbf{D}$) the i^{th} column of $\mathbf{Q}$ is obtained as a solution to

$$
\min_{\dot{\mathbf{q}}_i}\|\mathbf{S}_2^T\mathbf{d}_i - \mathbf{S}_2^T\hat{\mathbf{U}}\dot{\mathbf{q}}_i\|_1\,. \tag{9}
$$

As a result, the presented approach is amenable to online implementation. The CS can be obtained from a small batch of the data and each new data column can be decomposed based on the learned CS. Also, if the underlying subspace is changing, an alternating minimization approach can be used to update the learned CS. For brevity, we defer the discussion of online implementation to an extended version of this work (Rahmani & Atia, 2015b).

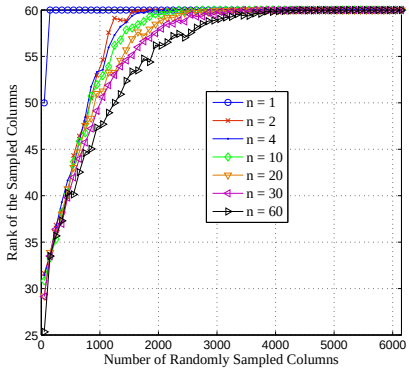

Figure 1. The rank of a set of uniformly random sampled columns for different number of clusters.

3. Efficient Column/Row Sampling

In sharp contrast to randomized algorithms for matrix approximations (Halko et al., 2011), in matrix decomposition and robust PCA we do not have direct access to the LR matrix to measure how informative particular columns/rows are. As such, the existing randomized algorithms for matrix decomposition and robust PCA (Li & Haupt, 2015; Mackey et al., 2011a) have predominantly relied upon uniform random sampling of columns/rows. In Section 3.1, we briefly describe the implications of non-uniform data distribution and show that uniform random sampling may not be favorable for data matrices exhibiting some structures that prevail much of the real datasets. In Section 3.2, we demonstrate an efficient column sampling strategy to be used in the proposed decomposition method. The proposed decomposition method with efficient column/row sampling is presented in Sections 3.3 and 3.4.

3.1. Non-uniform data distribution

Uniform random sampling is only effective when the columns/rows of $\mathbf{L}$ are distributed uniformly in the CS/RS of $\mathbf{L}$. In practice, the data points in a low-dimensional subspace may not be uniformly distributed and could exhibit some additional structures. A prevailing structure in many modern applications is clustered data (e.g. computer vision

(Rahmani & Atia, 2015c; Vidal, 2011), recommendation systems (Liu & Li, 2014)). For instance, consider a matrix $\mathbf{G} \in \mathbb{R}^{2000\times 6150}$ generated as $\mathbf{G} = [\mathbf{G}_1\, \mathbf{G}_2 \ldots \mathbf{G}_n]$, i.e., by concatenating the columns of matrices $\mathbf{G}_i, i = 1, \ldots, n$. For $1 \leq i \leq \frac{n}{2}$, $\mathbf{G}_i = \mathbf{U}_i\mathbf{Q}_i$, where $\mathbf{U}_i \in \mathbb{R}^{2000\times \frac{r}{n}}$, $\mathbf{Q}_i \in \mathbb{R}^{\frac{r}{n}\times \frac{200r}{n}}$. For $n/2 + 1 \leq i \leq n$, $\mathbf{G}_i = \mathbf{U}_i\mathbf{Q}_i$, where $\mathbf{U}_i \in \mathbb{R}^{2000\times \frac{r}{n}}$, $\mathbf{Q}_i \in \mathbb{R}^{\frac{r}{n}\times \frac{5r}{n}}$. The elements of $\mathbf{U}_i$ and $\mathbf{Q}_i$ are sampled independently from a normal $\mathcal{N}(0, 1)$ distribution. The parameter r is set equal to 60, thus, the rank of $\mathbf{G}$ is equal to 60 whp. Fig. 1 illustrates the rank of the randomly sampled columns versus the number of sampled columns for different number of clusters n. As n increases, so does the required number of uniformly sampled columns. When $n = 60$, it turns out that we need to sample more than half of the columns to span the CS. As such, we cannot evade high-dimensionality with uniform random column/row sampling.

3.2. Efficient column sampling

Column sampling is a well-known dimensionality reduction and feature selection technique (Halko et al., 2011). In the column sampling problem, the LR matrix (or the matrix whose span is to be approximated with a small set of its columns) is available. Thus, the columns are sampled based on their importance, measured by the so-called leverage scores, as opposed to blind uniform sampling. We refer the reader to (Deshpande & Rademacher, 2010; Halko et al., 2011) and references therein for more information about efficient column sampling methods. In this subsection, we present a sampling approach which will be used in the next subsection where the proposed decomposition algorithm with efficient sampling is presented. The proposed sampling strategy is inspired by the approach in (Deshpande & Rademacher, 2010) in the context of volume sampling. The table of Algorithm 2 details the proposed column sampling procedure. Given a matrix $\mathbf{A}$ with rank r_A, the algorithm aims to sample a small subset of the columns of $\mathbf{A}$ that span its CS. The first column is sampled uniformly at random or based on a judiciously chosen probability distribution (Boutsidis et al., 2009). The next columns are selected sequentially so as to maximize the novelty to the span of the selected columns. As shown in step 2.2 of Algorithm 2, a design threshold τ is used to decide whether a given column brings sufficient novelty to the sampled columns by thresholding the ℓ_2-norm of its projection on the complement of the span of the sampled columns. The threshold τ is naturally set to zero in a noise-free setting. Once the selected columns are believed to span the CS of $\mathbf{A}$, they are removed from $\mathbf{A}$. This procedure is repeated C times (using the remaining columns). In each time, the algorithm finds r columns which span the CS of $\mathbf{A}$. After every iteration, the rank of the matrix of remaining columns is bounded above by r_A. As such, the algorithm samples approximately $m_1 \approx Cr_A$ columns in total. In the proposed decomposition method with efficient column/row sampling (presented in Section 3.3), we set C large enough to ensure that the selected columns form a low rank matrix.

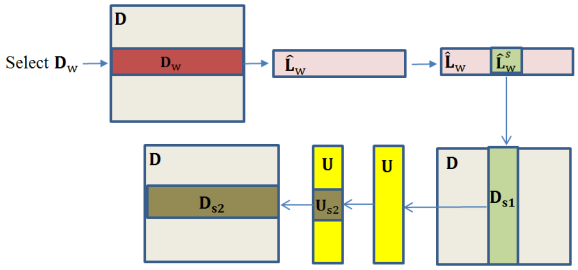

Figure 2. Visualization of the matrices defined in Section 3.3. Matrix $\mathbf{D}_w$ is selected randomly or via Algorithm 3.

Algorithm 2 Efficient Sampling from LR Matrices

Input: Matrix $\mathbf{A}$.

1. Initialize

1.1 Set C as an integer greater than or equal to 1. The algorithm finds C sets of linearly dependent columns.

1.2 Set $\mathcal{I} = \emptyset$ as the index set of the sampled columns and set $v = \tau$, $\mathbf{B} = \mathbf{A}$ and $\mathbf{C} = [\,]$.

2. Repeat C times

2.1 Let $\mathbf{b}$ be a non-zero randomly sampled column from $\mathbf{B}$ with index i_b. Update $\mathbf{C}$ and $\mathcal{I}$ as $\mathbf{C} = [\mathbf{C}\ \mathbf{b}], \mathcal{I} = \{\mathcal{I}\ ,\ i_b\}$.

2.2 While $v \geq \tau$

2.2.1 Set $\mathbf{E} = \mathbf{P}_c\mathbf{B}$, where $\mathbf{P}_c$ is the projection matrix onto the complement space of $\mathrm{span}(\mathbf{C})$.

2.2.2 Define $\mathbf{f}$ as the column of $\mathbf{E}$ with the maximum ℓ_2-norm with index i_f. Update $\mathbf{C}$, $\mathcal{I}$ and v as $\mathbf{C} = [\mathbf{C}\quad \mathbf{f}]$, $\mathcal{I} = \{\mathcal{I}\ ,\ i_f\}$ and $v = \|\mathbf{f}\|_2$.

2.2 End While

2.3 Set $\mathbf{C} = [\,]$ and set $\mathbf{B}$ equal to $\mathbf{A}$ with the columns indexed by $\mathcal{I}$ set to zero.

2. End Repeat

Output: The set $\mathcal{I}$ contains the indices of the selected columns.

3.3. Proposed decomposition algorithm with efficient sampling

In this section, we develop a modified decomposition algorithm that utilizes efficient information sampling as opposed to uniform random sampling. In Section 4, it is shown that the proposed technique can remarkably reduce the sampling requirement. We consider a setting wherein the columns of $\mathbf{L}$ are not uniformly distributed, rather they admit an additional structure (such as clustering), wherefore a small subset of uniformly sampled columns is not likely to span the CS. However, we assume that the *rows* of $\mathbf{L}$ are distributed well enough such that $C_r r$ rows of $\mathbf{L}$ sampled uniformly at random span its RS whp, for some

constant C_r. In Section 3.4, we dispense with this assumption. The proposed decomposition algorithm rests on three key ideas detailed next.

3.3.1. INFORMATIVE COLUMN SAMPLING

The first key idea underlying the proposed sampling approach is to start sampling along the dimension that has the better distribution. For instance, in the example of Section 3.1, the columns of $\mathbf{G}$ admit a clustering structure. However, since the CS of $\mathbf{G}$ is a random r-dimensional subspace, the *rows* of $\mathbf{G}$ are distributed uniformly at random in the RS of $\mathbf{G}$. Thus, in this case we start with row sampling. The main intuition is that while almost 60 randomly sampled rows of $\mathbf{G}$ span the RS, a considerable portion of the columns (almost 4000 columns) should be sampled to capture the CS as shown in Fig. 1. As another example, consider an extreme scenario where only two columns of $\mathbf{G} \in \mathbb{R}^{1000\times 1000}$ are non-zero. In this case, random sampling would sample almost all the columns to ensure that the sampled columns span the CS of $\mathbf{G}$. But, if the non-zero columns are non-sparse, a small subset of randomly chosen rows of $\mathbf{G}$ will span its row space.

Let $\hat{r}$ denote a known upper bound on r. We sample $C_r\hat{r}$ rows of $\mathbf{D}$ uniformly at random. For visualization, Fig. 2 provides a simplified illustration of the matrices defined in this section. Let $\mathbf{D}_w \in \mathbb{R}^{(C_r\hat{r})\times N_2}$ denote the matrix of sampled rows. We choose C_r sufficiently large to ensure that the non-sparse component of $\mathbf{D}_w$ is a LR matrix. Define $\mathbf{L}_w$, assumably with rank r, as the LR component of $\mathbf{D}_w$. If we locate a subset of the columns of $\mathbf{L}_w$ that span its CS, the corresponding columns of $\mathbf{L}$ would span the CS of $\mathbf{L}$. To this end, the convex program (2) is applied to $\mathbf{D}_w$ to extract a low rank component denoted $\hat{\mathbf{L}}_w$. Then, Algorithm 2 is applied to the recovered LR component $\hat{\mathbf{L}}_w$ to find a set of informative columns by sampling $m_1 \approx C\hat{r}$ columns. In the following remark, we discuss how to choose C here in the algorithm. Define $\hat{\mathbf{L}}_w^s$ as the matrix of columns selected from $\hat{\mathbf{L}}_w$. The matrix $\mathbf{D}_{s1}$ is formed using the columns of $\mathbf{D}$ corresponding to the sampled columns of $\hat{\mathbf{L}}_w$.

3.3.2. CS LEARNING

Similar to the CS learning step of Algorithm 1, we can obtain the CS of $\mathbf{L}$ by decomposing $\mathbf{D}_{s1}$. However, we propose to leverage valuable information in the matrix $\hat{\mathbf{L}}_w^s$ in decomposing $\mathbf{D}_{s1}$. In particular, if $\mathbf{D}_w$ is decomposed correctly, the RS of $\hat{\mathbf{L}}_w^s$ would be the same as that of $\mathbf{L}_{s1}$ given that $\mathbf{L}_w$ has rank r. Let $\mathbf{V}_{s1}$ be an orthonormal basis for the RS of $\hat{\mathbf{L}}_w^s$. Thus, in order to learn the CS of $\mathbf{D}_{s1}$, we only need to solve

$$\min_{\hat{\mathbf{U}}} \|\mathbf{D}_{s1} - \hat{\mathbf{U}}\mathbf{V}_{s1}^T\|_1 \,. \tag{10}$$

Remark 1. *Define $\mathbf{d}_{s1}^i$ as the i^{th} row of $\mathbf{D}_{s1}$. According to (10), $\mathbf{U}^i$ (the i^{th} row of $\mathbf{U}$) is obtained as the optimal point of*

$$\min_{\hat{\mathbf{U}}^i} \|(\mathbf{d}_{s1}^i)^T - \mathbf{V}_{s1}(\hat{\mathbf{U}}^i)^T\|_1 \,. \tag{11}$$

Define $\mathbf{S}_{s1}^i$ as the i^{th} row of $\mathbf{S}_{s1}$ and $\|\mathbf{S}_{s1}^i\|_0$ as the number of non-zero elements of $\mathbf{S}_{s1}^i$. Based on the analysis provided in the compressive sensing literature (Candès & Romberg, 2007), in order to ensure that (11) yields correct decomposition (the optimal point of (11) is equal to $\mathbf{U}^i$), $(m_1 - r)$ has to be sufficiently greater than $\|\mathbf{S}_{s1}^i\|_0$, i.e., $m_1 \geq r + \eta\|\mathbf{S}_{s1}^i\|_0$ for some sufficiently large constant number η where the required value for η depends on the coherency of the subspace spanned by $\mathbf{V}_{s1}$ (Candès & Romberg, 2007; Candès & Tao, 2005). Thus, here C is determined based on the rank of $\mathbf{L}$ and the sparsity of $\mathbf{S}$, i.e., $Cr - r$ has to be sufficiently greater than the expected value of the number of non-zero elements of the rows of $\mathbf{S}_{s1}$.

Remark 2. *Let $\mathbf{D}_w^s$ be the matrix of the columns of $\mathbf{D}_w$ corresponding to the columns of $\hat{\mathbf{L}}_w^s$ (which were selected from $\hat{\mathbf{L}}_w$). According to our investigations, an improved $\mathbf{V}_{s1}$ can be obtained by applying the alternating decomposition algorithm presented in (Ke & Kanade, 2005) to $\mathbf{D}_w^s$ using the RS of $\hat{\mathbf{L}}_w^s$ as an initial guess for the RS of the non-sparse component of $\mathbf{D}_w^s$. Since $\mathbf{D}_w^s$ is low-dimensional, this extra step is a low-complexity operation.*

3.3.3. REPRESENTATION MATRIX LEARNING

Suppose that the CS of $\mathbf{L}$ was learned correctly, i.e., the span of the optimal point of (10) is equal to the span of $\mathbf{U}$. Thus, we use $\mathbf{U}$ as a basis for the learned CS. Now we leverage the information embedded in $\mathbf{U}$ to select the informative rows. Algorithm 2 is applied to $\mathbf{U}^T$ to locate $m_2 \approx Cr$ rows of $\mathbf{U}$. We form the matrix $\mathbf{D}_{s2}$ from the rows of $\mathbf{D}$ corresponding to the selected rows of $\mathbf{U}$ (Fig. 2). Thus, the representation matrix is learned as

$$\min_{\hat{\mathbf{Q}}} \|\mathbf{D}_{s2} - \mathbf{U}_{s2}\hat{\mathbf{Q}}\|_1 \,, \tag{12}$$

where $\mathbf{U}_{s2} \in \mathbb{R}^{m_2\times r}$ is the matrix of the selected rows of $\mathbf{U}$. Subsequently, the LR matrix can be obtained from the learned CS and the representation matrix.

3.4. Column/Row sampling from corrupted LR matrices

In Section 3.3, we assumed that the LR component of $\mathbf{D}_w$ has rank r. However, if the rows are not well-distributed, a reasonably sized random subset of the rows may not span the RS of $\mathbf{L}$. Here, we present a sampling approach which can locate informative columns/rows even when both the columns and the rows exhibit clustering structures such that a small random subset of the columns/rows of $\mathbf{L}$ cannot

Algorithm 3 Efficient Column/Row Sampling from Sparsely Corrupted LR Matrices

1. Initialization

Form $\mathbf{D}_w \in \mathbb{R}^{C_r \hat{r} \times N_2}$ by randomly choosing $C_r \hat{r}$ rows of $\mathbf{D}$. Initialize $k = 1$ and set T equal to an integer greater than 1.

2. While $k > 0$

2.1 Sample the most informative columns

2.1.1 Obtain $\hat{\mathbf{L}}_w$ via (2) as the LR component of $\mathbf{D}_w$.

2.1.2 Apply Algorithm 2 to $\hat{\mathbf{L}}_w$ with $C = C_r$.

2.1.3 Form the matrix $\mathbf{D}_c$ from the columns of $\mathbf{D}$ corresponding to the sampled columns of $\hat{\mathbf{L}}_w$.

2.2 Sample the most informative rows

2.2.1 Obtain $\hat{\mathbf{L}}_c$ via (2) as the LR component of $\mathbf{D}_c$.

2.2.2 Apply Algorithm 2 to $\hat{\mathbf{L}}_c^T$ with $C = C_r$.

2.2.3 Form the matrix $\mathbf{D}_w$ from the rows of $\mathbf{D}$ corresponding to the sampled rows of $\hat{\mathbf{L}}_c$.

2.3 If the dimension of the RS of $\hat{\mathbf{L}}_w$ is not increased in T consecutive iterations, set $k = 0$ to stop the algorithm.

2. End While

Output: The matrix $\mathbf{D}_w$ and $\hat{\mathbf{L}}_w$ can be used for column sampling in the first step of the Algorithm presented in subsection 3.3.

span its CS/RS. Algorithm 3 presented in this section can be independently used as an efficient sampling approach from big data. In this paper, we use Algorithm 3 to form $\mathbf{D}_w$ if both the columns and rows exhibit a clustering structure.

Algorithm 3, Fig. 3 and its caption provide the details of the proposed sampling approach and the definitions of the used matrices. We start the cycle from the position marked "I" in Fig. 3 with $\mathbf{D}_w$ formed according to the initialization step of Algorithm 3. For ease of exposition, assume that $\hat{\mathbf{L}}_w = \mathbf{L}_w$ and $\hat{\mathbf{L}}_c = \mathbf{L}_c$, i.e., $\mathbf{D}_w$ and $\mathbf{D}_c$ are decomposed correctly. The matrix $\hat{\mathbf{L}}_w^s$ is the informative columns of $\hat{\mathbf{L}}_w$. Thus, the rank of $\hat{\mathbf{L}}_w^s$ is equal to the rank of $\hat{\mathbf{L}}_w$. Since $\hat{\mathbf{L}}_w = \mathbf{L}_w$, $\hat{\mathbf{L}}_w^s$ is a subset of the rows of $\mathbf{L}_c$. If the rows of $\mathbf{L}$ exhibit a clustering structure, it is likely that $\text{rank}(\hat{\mathbf{L}}_w^s) < \text{rank}(\mathbf{L}_c)$. Thus, $\text{rank}(\mathbf{L}_w) < \text{rank}(\mathbf{L}_c)$. We continue one cycle of the algorithm by going through steps 1, 2 and 3 of Fig. 3 to update $\mathbf{D}_w$. Using a similar argument, we see that the rank of an updated $\mathbf{L}_w$ will be greater than the rank of $\mathbf{L}_c$. Thus, if we run more cycles of the algorithm – each time updating $\mathbf{D}_w$ and $\mathbf{D}_c$ – the rank of $\mathbf{L}_w$ and $\mathbf{L}_c$ will increase. As detailed in the table of Algorithm 3, we stop if the rank of the obtained LR component does not change in T consecutive iterations. While there is no guarantee that the rank of $\mathbf{L}_w$ will converge to r (it can converge to a value smaller than r), our investigations have shown that Algorithm 3 performs quite well and the RS of $\mathbf{L}_w$ converges to the RS of $\mathbf{L}$ in few steps. We have also found that adding some randomly sampled columns (rows) to $\mathbf{D}_c(\mathbf{D}_w)$ can effectively avert converging to a lower dimensional subspace. For instance, some randomly sampled columns can be added to $\mathbf{D}_c$, which was obtained by Applying Algorithm 2 to $\hat{\mathbf{L}}_w$.

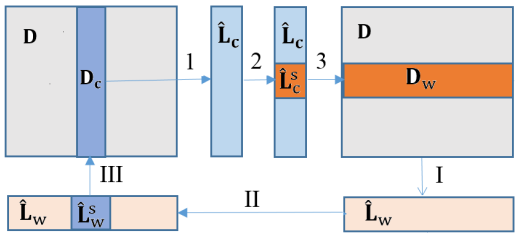

Figure 3. Visualization of Algorithm 3. One cycle of the algorithm starts from the point marked "I" and proceeds as follows. **I**: Matrix $\mathbf{D}_w$ is decomposed and $\hat{\mathbf{L}}_w$ is the obtained LR component of $\mathbf{D}_w$. **II**: Algorithm 2 is applied to $\hat{\mathbf{L}}_w$ to select the informative columns of $\hat{\mathbf{L}}_w$. $\hat{\mathbf{L}}_w^s$ is the matrix of columns selected from $\hat{\mathbf{L}}_w$. **III**: Matrix $\mathbf{D}_c$ is formed as the columns of $\mathbf{D}$ corresponding to the columns used to form $\hat{\mathbf{L}}_w^s$. **1**: Matrix $\mathbf{D}_c$ is decomposed and $\hat{\mathbf{L}}_c$ is the obtained LR component of $\mathbf{D}_c$. **2**: Algorithm 2 is applied to $\hat{\mathbf{L}}_c^T$ to select the informative rows of $\hat{\mathbf{L}}_c$. $\hat{\mathbf{L}}_c^s$ is the matrix of rows selected from $\hat{\mathbf{L}}_c$. **3**: Matrix $\mathbf{D}_w$ is formed as the rows of $\mathbf{D}$ corresponding to the rows used to form $\hat{\mathbf{L}}_c^s$.

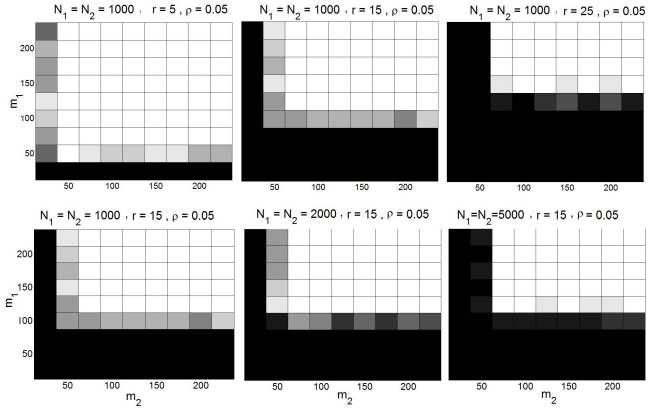

Figure 4. Phase transition plots for various rank values and data matrix dimensions. White designates successful decomposition and black designates incorrect decomposition.

We have found that Algorithm 3 converges in a very small number of iterations (usually less than 4 iterations). Thus, even when we use Algorithm 3 to form the matrix $\mathbf{D}_w$, the order of complexity of the proposed decomposition method with efficient column/row sampling (presented in Section 3.3) is roughly $\mathcal{O}(\max(N_1, N_2)\, r^2)$.

4. Numerical Simulations

In this section, we present some numerical simulations confirming the provided analysis to study the performance of the proposed scalable approach. In all simulations, the Augmented Lagrange multiplier (ALM) algorithm (Boyd

et al., 2011; Candès et al., 2011; Lin et al., 2010; Minaee & Wang, 2015) is used to solve the optimization problems.

4.1. Phase transition plots

In this section, we investigate the required number of randomly sampled columns/rows. The LR matrix is generated as a product $\mathbf{L} = \mathbf{U}_r\mathbf{Q}_r$, where $\mathbf{U}_r \in \mathbb{R}^{N_1 \times r}$ and $\mathbf{Q}_r \in \mathbb{R}^{r \times N_2}$. The elements of $\mathbf{U}_r$ and $\mathbf{Q}_r$ are sampled independently from a standard normal $\mathcal{N}(0,1)$ distribution. The sparse matrix $\mathbf{S}$ follows the Bernoulli model with $\rho = 0.02$. In this experiment, Algorithm 1 is used and the column/rows are sampled uniformly at random. Fig. 4 shows the phase transition plots for different numbers of randomly sampled rows/columns. For each (m_1, m_2), we generate 10 random realizations. A trial is considered successful if the recovered LR matrix $\hat{\mathbf{L}}$ satisfies $\frac{\|\mathbf{L}-\hat{\mathbf{L}}\|_F}{\|\mathbf{L}\|_F} \leq 5 \times 10^{-3}$.

In the first row of Fig. 4, the data is a 1000×1000 matrix. The required number of sampled columns/rows increases as the rank increases. In this experiment, the CS and RS of $\mathbf{L}$ are sampled from the random orthogonal model. Thus, the CS and RS have small coherency whp (Candès & Recht, 2009). Therefore, the important factor governing the sample complexity is the rank of $\mathbf{L}$. The second row of Fig. 4 shows the phase transition for different sizes of the data matrix when the rank of $\mathbf{L}$ is fixed. In agreement with our analysis, the required values for m_1 and m_2 are almost independent of the size of the data.

4.2. Efficient column/row sampling

In this experiment, the algorithm presented in Section 3.3 is compared to the randomized decomposition algorithm in (Mackey et al., 2011b). It is shown that the proposed sampling strategy can effectively reduce the required number of sampled columns/rows, and makes the proposed method remarkably robust to structured data. In this experiment, $\mathbf{D}$ is a 2000×4200 matrix. The LR component is generated as $\mathbf{L} = [\mathbf{G}_1\ \mathbf{G}_2\ ...\ \mathbf{G}_n]$. For $1 \leq i \leq \frac{n}{2}$, $\mathbf{G}_i = \mathbf{U}_i\mathbf{Q}_i$, where $\mathbf{U}_i \in \mathbb{R}^{2000 \times \frac{r}{n}}$, $\mathbf{Q}_i \in \mathbb{R}^{\frac{r}{n} \times \frac{130r}{n}}$ and the elements of $\mathbf{U}_i$ and $\mathbf{Q}_i$ are sampled independently from a normal distribution $\mathcal{N}(0,1)$. For $n/2 + 1 \leq i \leq n$, $\mathbf{G}_i = 13\mathbf{U}_i\mathbf{Q}_i$, where $\mathbf{U}_i \in \mathbb{R}^{2000 \times \frac{r}{n}}$, $\mathbf{Q}_i \in \mathbb{R}^{\frac{r}{n} \times \frac{10r}{n}}$, and the elements of $\mathbf{U}_i$ and $\mathbf{Q}_i$ are sampled independently from an $\mathcal{N}(0,1)$ distribution. We set r equal to 60; thus, the rank of $\mathbf{L}$ is equal to 60 whp. The sparse matrix $\mathbf{S}$ follows the Bernoulli model and each element of $\mathbf{S}$ is non-zero with probability 0.02. In this simulation, we do not use the alternating method presented in Section 3.4 to form $\mathbf{D}_w$. The matrix $\mathbf{D}_w$ is formed from 300 uniformly sampled *rows* of $\mathbf{D}$.

We evaluate the performance of the algorithm for different values of n, i.e., different number of clusters. Fig. 5 shows the performance of the proposed approach and the approach in (Mackey et al., 2011b) for different values of m_1 and m_2. For each value of $m_1 = m_2$, we compute the error in LR matrix recovery $\frac{\|\mathbf{L}-\hat{\mathbf{L}}\|_F}{\|\mathbf{L}\|_F}$ averaged over 10 independent runs, and conclude that the algorithm can yield correct decomposition if the average error is less than 0.01. In Fig. 5, the values 0, 1 designate incorrect and correct decomposition, respectively. It can be seen that the presented approach requires a significantly smaller number of samples to yield the correct decomposition. This is due to the fact that the randomized algorithm (Mackey et al., 2011b) samples both the columns and rows uniformly at random and independently. In sharp contrast, we use $\hat{\mathbf{L}}_w$ to find the most informative columns to form $\mathbf{D}_{s1}$, and also leverage the information embedded in the CS to find the informative rows to form $\mathbf{D}_{s2}$.

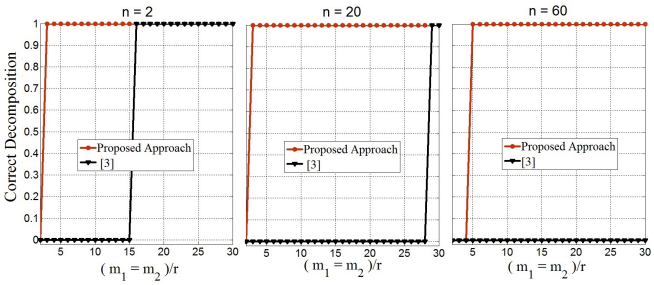

Figure 5. Performance of the proposed approach and the randomized algorithm in (Mackey et al., 2011b). A value 1 indicates correct decomposition and a value 0 indicates incorrect decomposition.

4.3. Vector decomposition for background subtraction

The LR plus sparse matrix decomposition can be effectively used to detect a moving object in a stationary background (Candès et al., 2011). The background is modeled as a LR matrix and the moving object as a sparse matrix. Since videos are typically high dimensional objects, standard algorithms can be quite slow for such applications. Our algorithm is a good candidate for such a problem as it reduces the dimensionality effectively. The decomposition problem can be further simplified by leveraging prior information about the stationary background. In particular, we know that the background does not change or we can construct it with some pre-known dictionary. For example, consider the video from (Li et al., 2004), which was also used in (Candès et al., 2011). Few frames of the stationary background are illustrated in Fig. 6. Thus, we can simply form the column subspace of the LR matrix using these frames which can describe the stationary background in different states. Accordingly, we just need to learn the representation matrix and the background subtraction problem is simplified to a vector decomposition problem. Fig. 7

Figure 6. Stationary background.

Figure 7. Two frames of a video taken in a lobby. The first column displays the original frames. The second and third columns display the LR and sparse components recovered using the proposed approach.

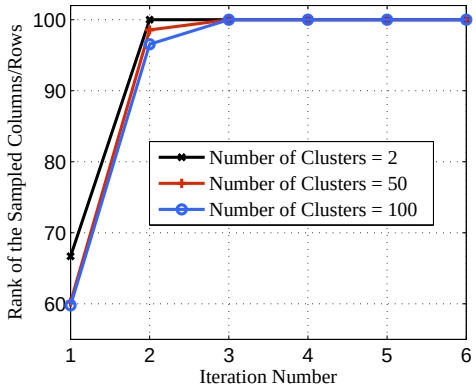

Figure 8. The rank of the matrix of sampled columns.

shows that the proposed method successfully separates the background and the moving objects. In this experiment, 500 randomly sampled rows are used (i.e., 500 randomly sampled pixels) for the representation matrix learning (9). While the running time of our approach is just few milliseconds, it takes about half an hour if we use (2) to decompose the video file (Candès et al., 2011).

4.4. Alternating algorithm for column sampling

Finally, we investigate the performance of Algorithm 3 for column sampling. It is shown that the CS (RS) of the selected columns (rows) converges to the CS of $\mathbf{L}$ (RS of $\mathbf{L}$) even when both the rows and columns of $\mathbf{L}$ exhibit a highly structured distribution. To generate the LR matrix $\mathbf{L}$, we first generate a matrix $\mathbf{G}$ as in Section 3.1 but setting $r = 100$. Then, we construct the matrix $\mathbf{U}_g$ from the first r right singular vectors of $\mathbf{G}$. Again, we generate $\mathbf{G}$ in a similar way and set $\mathbf{V}_g$ equal to the first r right singular vectors of $\mathbf{G}$. Let the matrix $\mathbf{L} = \mathbf{U}_g\mathbf{V}_g^T$. For example, for $n = 100$, $\mathbf{L} \in \mathbb{R}^{10250\times 10250}$. Note that the resulting LR matrix is nearly sparse since in this simulation we consider a very challenging scenario in which both the columns and rows of $\mathbf{L}$ are highly structured and coherent. In this experiment we set the sparse matrix equal to zero and use Algorithm 3 as follows. The matrix $\mathbf{D}_c$ is formed using 300 columns sampled uniformly at random and the following steps are performed iteratively:
1. Apply Algorithm 2 to $\mathbf{D}_c^T$ with $C = 3$ to sample approximately $3r$ columns of $\mathbf{D}_c^T$ and form $\mathbf{D}_w$ from the rows of $\mathbf{D}$ corresponding to the selected rows of $\mathbf{D}_c$.
2. Apply Algorithm 2 to $\mathbf{D}_w$ with $C = 3$ to sample approximately $3r$ columns of $\mathbf{D}_w$ and form $\mathbf{D}_c$ from the columns of $\mathbf{D}$ corresponding to the selected columns of $\mathbf{D}_c$.
Fig. 8 shows the rank of $\mathbf{D}_c$ after each iteration. It is evident that the algorithm converges to the rank of $\mathbf{L}$ in less than 3 iterations even for $n = 100$ clusters. For all values of n, i.e., $n \in \{2, 50, 100\}$, the data is a 10250×10250 matrix.

Acknowledgments

This work was supported by NSF Grant CCF-1320547 and NSF CAREER Award CCF-1552497.

References

Boutsidis, Christos, Mahoney, Michael W, and Drineas, Petros. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 968–977. Society for Industrial and Applied Mathematics, 2009.

Boyd, Stephen, Parikh, Neal, Chu, Eric, Peleato, Borja, and Eckstein, Jonathan. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

Candès, Emmanuel and Romberg, Justin. Sparsity and incoherence in compressive sampling. *Inverse problems*, 23(3):969, 2007.

Candès, Emmanuel J and Recht, Benjamin. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

Candès, Emmanuel J and Tao, Terence. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

Candès, Emmanuel J, Li, Xiaodong, Ma, Yi, and Wright,

John. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.

Chandrasekaran, Venkat, Sanghavi, Sujay, Parrilo, Pablo A, and Willsky, Alan S. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21(2):572–596, 2011.

Deshpande, Amit and Rademacher, Luis. Efficient volume sampling for row/column subset selection. In *51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 329–338, 2010.

Halko, Nathan, Martinsson, Per-Gunnar, and Tropp, Joel A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

Ke, Qifa and Kanade, Takeo. Robust L_1 norm factorization in the presence of outliers and missing data by alternative convex programming. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pp. 739–746, 2005.

Li, Liyuan, Huang, Weimin, Gu, Irene Yu-Hua, and Tian, Qi. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Transactions on Image Processing*, 13(11):1459–1472, 2004.

Li, Xingguo and Haupt, Jarvis. Identifying outliers in large matrices via randomized adaptive compressive sampling. *IEEE Transactions on Signal Processing*, 63 (7):1792–1807, 2015.

Lin, Zhouchen, Chen, Minming, and Ma, Yi. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.

Liu, Guangcan and Li, Ping. Recovery of coherent data via low-rank dictionary pursuit. In *Advances in Neural Information Processing Systems*, pp. 1206–1214, 2014.

Mackey, Lester, Talwalkar, Ameet, and Jordan, Michael I. Distributed matrix completion and robust factorization. *arXiv preprint arXiv:1107.0789*, 2011a.

Mackey, Lester W, Jordan, Michael I, and Talwalkar, Ameet. Divide-and-conquer matrix factorization. In *Advances in Neural Information Processing Systems*, pp. 1134–1142, 2011b.

Minaee, Shervin and Wang, Yao. Screen content image segmentation using least absolute deviation fitting. In *IEEE International Conference on Image Processing (ICIP)*, pp. 3295–3299, 2015.

Rahmani, Mostafa and Atia, George. Randomized robust subspace recovery for high dimensional data matrices. *arXiv preprint arXiv:1505.05901*, 2015a.

Rahmani, Mostafa and Atia, George. High dimensional low rank plus sparse matrix decomposition. *arXiv preprint arXiv:1502.00182*, 2015b.

Rahmani, Mostafa and Atia, George. Innovation pursuit: A new approach to subspace clustering. *arXiv preprint arXiv:1512.00907*, 2015c.

Vidal, Rene. Subspace clustering. *IEEE Signal Processing Magazine*, 2(28):52–68, 2011.

Wright, John, Ganesh, Arvind, Min, Kerui, and Ma, Yi. Compressive principal component pursuit. *Information and Inference*, 2(1):32–68, 2013.

Yuan, Xiaoming and Yang, Junfeng. Sparse and low-rank matrix decomposition via alternating direction methods. *preprint*, 12, 2009.