Sparse Subspace Clustering: Algorithm, Theory, and Applications

Ehsan Elhamifar, *Student Member, IEEE,* and René Vidal, *Senior Member, IEEE*

Abstract—Many real-world problems deal with collections of high-dimensional data, such as images, videos, text and web documents, DNA microarray data, and more. Often, such high-dimensional data lie close to low-dimensional structures corresponding to several classes or categories to which the data belong. In this paper, we propose and study an algorithm, called Sparse Subspace Clustering (SSC), to cluster data points that lie in a union of low-dimensional subspaces. The key idea is that, among the infinitely many possible representations of a data point in terms of other points, a sparse representation corresponds to selecting a few points from the same subspace. This motivates solving a sparse optimization program whose solution is used in a spectral clustering framework to infer the clustering of the data into subspaces. Since solving the sparse optimization program is in general NP-hard, we consider a convex relaxation and show that, under appropriate conditions on the arrangement of the subspaces and the distribution of the data, the proposed minimization program succeeds in recovering the desired sparse representations. The proposed algorithm is efficient and can handle data points near the intersections of subspaces. Another key advantage of the proposed algorithm with respect to the state of the art is that it can deal directly with data nuisances, such as noise, sparse outlying entries, and missing entries, by incorporating the model of the data into the sparse optimization program. We demonstrate the effectiveness of the proposed algorithm through experiments on synthetic data as well as the two real-world problems of motion segmentation and face clustering.

Index Terms—High-dimensional data, intrinsic low-dimensionality, subspaces, clustering, sparse representation, ℓ_1-minimization, convex programming, spectral clustering, principal angles, motion segmentation, face clustering.

1 Introduction

High-dimensional data are ubiquitous in many areas of machine learning, signal and image processing, computer vision, pattern recognition, bioinformatics, etc. For instance, images consist of billions of pixels, videos can have millions of frames, text and web documents are associated with hundreds of thousands of features, etc. The high-dimensionality of the data not only increases the computational time and memory requirements of algorithms, but also adversely affects their performance due to the noise effect and insufficient number of samples with respect to the ambient space dimension, commonly referred to as the "curse of dimensionality" [1]. However, high-dimensional data often lie in low-dimensional structures instead of being uniformly distributed across the ambient space. Recovering low-dimensional structures in the data helps to not only reduce the computational cost and memory requirements of algorithms, but also reduce the effect of high-dimensional noise in the data and improve the performance of inference, learning, and recognition tasks.

In fact, in many problems, data in a class or category can be well represented by a low-dimensional subspace of the high-dimensional ambient space. For example, feature trajectories of a rigidly moving object in a video [2], face images of a subject under varying illumination [3], and multiple instances of a hand-written digit with different rotations, translations, and thicknesses [4] lie in a low-dimensional subspace of the ambient space. As a result, the collection of data from multiple classes or categories lie in a union of low-dimensional subspaces. Subspace clustering (see [5] and references therein) refers to the problem of separating data according to their underlying subspaces and finds numerous applications in image processing (e.g., image representation and compression [6]) and computer vision (e.g., image segmentation [7], motion segmentation [8], [9], and temporal video segmentation [10]), as illustrated in Figures 1 and 2. Since data in a subspace are often distributed arbitrarily and not around a centroid, standard clustering methods [11] that take advantage of the spatial proximity of the data in each cluster are not in general applicable to subspace clustering. Therefore, there is a need for having clustering algorithms that take into account the multi-subspace structure of the data.

- *E. Elhamifar is with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, USA. E-mail: ehsan@eecs.berkeley.edu.*
- *R. Vidal is with the Center for Imaging Science and the Department of Biomedical Engineering, The Johns Hopkins University, USA. E-mail: rvidal@cis.jhu.edu.*

1.1 Prior Work on Subspace Clustering

Existing algorithms can be divided into four main categories: iterative, algebraic, statistical, and spectral clustering-based methods.

Iterative methods. Iterative approaches, such as K-subspaces [12], [13] and median K-flats [14] alternate between assigning points to subspaces and fitting a subspace to each cluster. The main drawbacks of such approaches are that they generally require to know the number and dimensions of the subspaces, and that they are sensitive to initialization.

Algebraic approaches. Factorization-based algebraic approaches such as [8], [9], [15] find an initial segmentation by thresholding the entries of a similarity matrix built from the factorization of the data matrix. These methods are provably correct when the subspaces are independent, but fail when this assumption is violated. In addition, they are sensitive to noise and outliers in the data. Algebraic-geometric approaches such as Generalized Principal Component Analysis (GPCA) [10], [16], fit the data with a polynomial whose gradient at a point gives the normal vector to the subspace containing that point. While GPCA can

Fig. 1. Motion segmentation: given feature points on multiple rigidly moving objects tracked in multiple frames of a video (top), the goal is to separate the feature trajectories according to the moving objects (bottom).

Fig. 2. Face clustering: given face images of multiple subjects (top), the goal is to find images that belong to the same subject (bottom).

deal with subspaces of different dimensions, it is sensitive to noise and outliers, and its complexity increases exponentially in terms of the number and dimensions of subspaces.

Statistical methods. Iterative statistical approaches, such as Mixtures of Probabilistic PCA (MPPCA) [17], Multi-Stage Learning (MSL) [18], or [19], assume that the distribution of the data inside each subspace is Gaussian and alternate between data clustering and subspace estimation by applying Expectation Maximization (EM). The main drawbacks of these methods are that they generally need to know the number and dimensions of the subspaces, and that they are sensitive to initialization. Robust statistical approaches, such as Random Sample Consensus (RANSAC) [20], fit a subspace of dimension d to randomly chosen subsets of d points until the number of inliers is large enough. The inliers are then removed, and the process is repeated to find a second subspace, and so on. RANSAC can deal with noise and outliers, and does not need to know the number of subspaces. However, the dimensions of the subspaces must be known and equal. In addition, the complexity of the algorithm increases exponentially in the dimension of the subspaces. Information-theoretic statistical approaches, such as Agglomerative Lossy Compression (ALC) [21], look for the segmentation of the data that minimizes the coding length needed to fit the points with a mixture of degenerate Gaussians up to a given distortion. As this minimization problem is NP-hard, a suboptimal solution is found by first assuming that each point forms its own group, and then iteratively merging pairs of groups to reduce the coding length. ALC can handle noise and outliers in the data. While, in principle, it does not need to know the number and dimensions of the subspaces, the number of subspaces found by the algorithms is dependent on the choice of a distortion parameter. In addition, there is no theoretical proof for the optimality of the agglomerative algorithm.

Spectral clustering-based methods. Local spectral clustering-based approaches such as Local Subspace Affinity (LSA) [22], Locally Linear Manifold Clustering (LLMC) [23], Spectral Local Best-fit Flats (SLBF) [24], and [25] use local information around each point to build a similarity between pairs of points. The segmentation of the data is then obtained by applying spectral clustering [26], [27] to the similarity matrix. These methods have difficulties in dealing with points near the intersection of two subspaces, because the neighborhood of a point can contain points from different subspaces. In addition, they are sensitive to the right choice of the neighborhood size to compute the local information at each point.

Global spectral clustering-based approaches try to resolve these issues by building better similarities between data points using global information. Spectral Curvature Clustering (SCC) [28] uses multi-way similarities that capture the curvature of a collection of points within an affine subspace. SCC can deal with noisy data but requires to know the number and dimensions of subspaces and assumes that subspaces have the same dimension. In addition, the complexity of building the multi-way similarity grows exponentially with the dimensions of the subspaces, hence, in practice, a sampling strategy is employed to reduce the computational cost. Using advances in sparse [29], [30], [31] and low-rank [32], [33], [34] recovery algorithms, Sparse Subspace Clustering (SSC) [35], [36], [37], Low-Rank Recovery (LRR) [38], [39], [40], and Low-Rank Subspace Clustering (LRSC) [41] algorithms pose the clustering problem as one of finding a sparse or low-rank representation of the data in the dictionary of the data itself. The solution of the corresponding global optimization algorithm is then used to build a similarity graph from which the segmentation of the data is obtained. The advantages of these methods with respect to most state-of-the-art algorithms are that they can handle noise and outliers in data, and that they do not need to know the dimensions and, in principle, the number of subspaces a priori.

1.2 Paper Contributions

In this paper, we propose and study an algorithm based on sparse representation techniques, called Sparse Subspace Clustering (SSC), to cluster a collection of data points lying in a union of low-dimensional subspaces. The underlying idea behind the algorithm is what we call the *self-expressiveness* property of the data, which states that each data point in a union of subspaces can be efficiently represented as a linear or affine combination of other points. Such a representation is not unique in general because there are infinitely many ways in which a data point can be

expressed as a combination of other points. The key observation is that a *sparse representation* of a data point ideally corresponds to a combination of a few points from its own subspace. This motivates solving a global sparse optimization program whose solution is used in a spectral clustering framework to infer the clustering of data. As a result, we can overcome the problems of local spectral clustering-based algorithms, such as choosing the right neighborhood size and dealing with points near the intersection of subspaces, since, for a given data point, the sparse optimization program automatically picks a few other points that are not necessarily close to it but belong to the same subspace.

Since solving the sparse optimization program is in general NP-hard, we consider its ℓ_1 relaxation. We show that, under mild conditions on the arrangement of subspaces and data distribution, the proposed ℓ_1-minimization program recovers the desired solution, guaranteeing the success of the algorithm. Our theoretical analysis extends the sparse representation theory to the multi-subspace setting where the number of points in a subspace is arbitrary, possibly much larger than its dimension. Unlike block-sparse recovery problems [42], [43], [44], [45], [46], [47] where the bases for the subspaces are known and given, we do not have the bases for subspaces nor do we know which data points belong to which subspace, making our case more challenging. We only have the sparsifying dictionary for the union of subspaces given by the matrix of data points.

The proposed ℓ_1-minimization program can be solved efficiently using convex programming tools [48], [49], [50] and does not require initialization. Our algorithm can directly deal with noise, sparse outlying entries, and missing entries in the data as well as the more general class of affine subspaces by incorporating the data corruption or subspace model into the sparse optimization program. Finally, through experimental results, we show that our algorithm outperforms state-of-the-art subspace clustering methods on the two real-world problems of motion segmentation (Fig. 1) and face clustering (Fig. 2).

Paper Organization. In Section 2, we motivate and introduce the SSC algorithm for clustering data points in a union of linear subspaces. In Section 3, we generalize the algorithm to deal with noise, sparse outlying entries, and missing entries in the data as well as the more general class of affine subspaces. In Section 4, we investigate theoretical conditions under which the ℓ_1-minimization program recovers the desired sparse representations of data points. In Section 5, we discuss the connectivity of the similarity graph and propose a regularization term to increase the connectivity of points in each subspace. In Section 6, we verify our theoretical analysis through experiments on synthetic data. In Section 7, we compare the performance of SSC with the state of the art on the two real-world problems of motion segmentation and face clustering. Finally, Section 8 concludes the paper.

2 Sparse Subspace Clustering

In this section, we introduce the sparse subspace clustering (SSC) algorithm for clustering a collection of multi-subspace data using sparse representation techniques. We motivate and formulate the algorithm for data points that perfectly lie in a union of linear subspaces. In the next section, we will generalize the algorithm to deal with data nuisances such as noise, sparse outlying entries, and missing entries as well as the more general class of affine subspaces.

Let $\{\mathcal{S}_\ell\}_{\ell=1}^n$ be an arrangement of n linear subspaces of $\mathbb{R}^D$ of dimensions $\{d_\ell\}_{\ell=1}^n$. Consider a given collection of N noise-free data points $\{\boldsymbol{y}_i\}_{i=1}^N$ that lie in the union of the n subspaces. Denote the matrix containing all the data points as

$$\boldsymbol{Y} \triangleq \begin{bmatrix}\boldsymbol{y}_1 & \dots & \boldsymbol{y}_N\end{bmatrix} = \begin{bmatrix}\boldsymbol{Y}_1 & \dots & \boldsymbol{Y}_n\end{bmatrix}\boldsymbol{\Gamma}, \tag{1}$$

where $\boldsymbol{Y}_\ell \in \mathbb{R}^{D\times N_\ell}$ is a rank-d_ℓ matrix of the $N_\ell > d_\ell$ points that lie in $\mathcal{S}_\ell$ and $\boldsymbol{\Gamma} \in \mathbb{R}^{N\times N}$ is an unknown permutation matrix. We assume that we do not know a priori the bases of the subspaces nor do we know which data points belong to which subspace. The *subspace clustering* problem refers to the problem of finding the number of subspaces, their dimensions, a basis for each subspace, and the segmentation of the data from $\boldsymbol{Y}$.

To address the subspace clustering problem, we propose an algorithm that consists of two steps. In the first step, for each data point, we find a few other points that belong to the same subspace. To do so, we propose a global sparse optimization program whose solution encodes information about the memberships of data points to the underlying subspace of each point. In the second step, we use these information in a spectral clustering framework to infer the clustering of the data.

2.1 Sparse Optimization Program

Our proposed algorithm takes advantage of what we refer to as the *self-expressiveness property* of the data, i.e.,

> *each data point in a union of subspaces can be efficiently reconstructed by a combination of other points in the dataset.*

More precisely, each data point for data point $\boldsymbol{y}_i \in \cup_{\ell=1}^n \mathcal{S}_\ell$ can be written as

$$\boldsymbol{y}_i = \boldsymbol{Y}\boldsymbol{c}_i, \quad c_{ii} = 0, \tag{2}$$

where $\boldsymbol{c}_i \triangleq \begin{bmatrix}c_{i1} & c_{i2} & \dots & c_{iN}\end{bmatrix}^\top$ and the constraint $c_{ii} = 0$ eliminates the trivial solution of writing a point as a linear combination of itself. In other words, the matrix of data points $\boldsymbol{Y}$ is a self-expressive dictionary in which each point can be written as a linear combination of other points. However, the representation of $\boldsymbol{y}_i$ in the dictionary $\boldsymbol{Y}$ is *not unique* in general. This comes from the fact that the number of data points in a subspace is often greater than its dimension, i.e., $N_\ell > d_\ell$. As a result, each $\boldsymbol{Y}_\ell$, and consequently $\boldsymbol{Y}$, has a non-trivial nullspace giving rise to infinitely many representations of each data point.

The key observation in our proposed algorithm is that among all solutions of (2),

> *there exists a sparse solution, $\boldsymbol{c}_i$, whose nonzero entries correspond to data points from the same subspace as $\boldsymbol{y}_i$. We refer to such a solution as a subspace-sparse representation.*

More specifically, a data point $\boldsymbol{y}_i$ that lies in the d_ℓ-dimensional subspace $\mathcal{S}_\ell$ can be written as a linear combination of d_ℓ other points in general directions from $\mathcal{S}_\ell$. As a result, ideally, a sparse representation of a data point finds points from the same subspace where the number of the nonzero elements corresponds to the dimension of the underlying subspace.

For a system of equations such as (2) with infinitely many solutions, one can restrict the set of solutions by minimizing an objective function such as the ℓ_q-norm of the solution[1] as

$$\min \|\boldsymbol{c}_i\|_q \quad \text{s.t.} \quad \boldsymbol{y}_i = \boldsymbol{Y}\boldsymbol{c}_i,\ c_{ii} = 0. \tag{3}$$

1. The ℓ_q-norm of $\boldsymbol{c}_i \in \mathbb{R}^N$ is defined as $\|\boldsymbol{c}_i\|_q \triangleq (\sum_{j=1}^N |c_{ij}|^q)^{\frac{1}{q}}$.

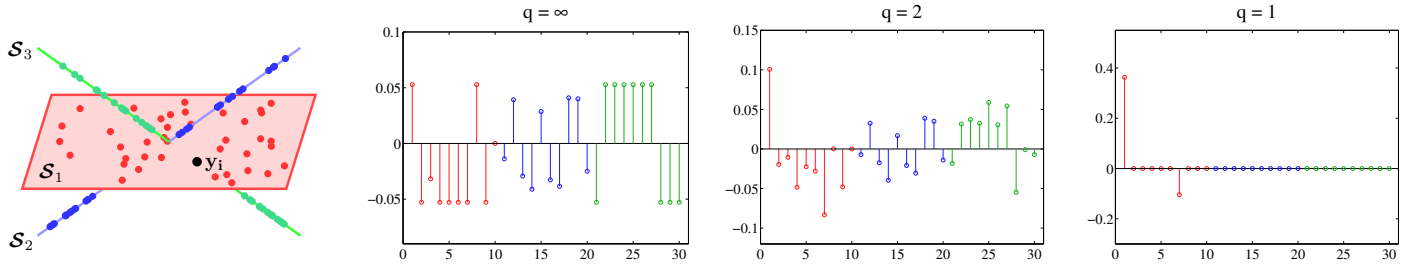


Fig. 3. Three subspaces in $\mathbb{R}^3$ with 10 data points in each subspace, ordered such that the fist and the last 10 points belong to $\mathcal{S}_1$ and $\mathcal{S}_3$, respectively. The solution of the ℓ_q-minimization program in (3) for $\boldsymbol{y}_i$ lying in $\mathcal{S}_1$ for $q = 1, 2, \infty$ is shown. Note that as the value of q decreases, the sparsity of the solution increases. For $q = 1$, the solution corresponds to choosing two other points lying in $\mathcal{S}_1$.

Different choices of q have different effects in the obtained solution. Typically, by decreasing the value of q from infinity toward zero, the sparsity of the solution increases, as shown in Figure 3. The extreme case of $q = 0$ corresponds to the general NP-hard problem [51] of finding the sparsest representation of the given point, as the ℓ_0-norm counts the number of nonzero elements of the solution. Since we are interested in efficiently finding a non-trivial sparse representation of $\boldsymbol{y}_i$ in the dictionary $\boldsymbol{Y}$, we consider minimizing the tightest convex relaxation of the ℓ_0-norm, i.e.,

$$\min \|\boldsymbol{c}_i\|_1 \quad \text{s.t.} \quad \boldsymbol{y}_i = \boldsymbol{Y}\boldsymbol{c}_i,\ c_{ii} = 0, \tag{4}$$

which can be solved efficiently using convex programming tools [48], [49], [50] and is known to prefer sparse solutions [29], [30], [31].

We can also rewrite the sparse optimization program (4) for all data points $i = 1, \ldots, N$ in matrix form as

$$\min \|\boldsymbol{C}\|_1 \quad \text{s.t.} \quad \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C}, \quad \operatorname{diag}(\boldsymbol{C}) = \boldsymbol{0}, \tag{5}$$

where $\boldsymbol{C} \triangleq \begin{bmatrix}\boldsymbol{c}_1 & \boldsymbol{c}_2 & \ldots & \boldsymbol{c}_N\end{bmatrix} \in \mathbb{R}^{N\times N}$ is the matrix whose i-th column corresponds to the sparse representation of $\boldsymbol{y}_i$, $\boldsymbol{c}_i$, and $\operatorname{diag}(\boldsymbol{C}) \in \mathbb{R}^N$ is the vector of the diagonal elements of $\boldsymbol{C}$.

Ideally, the solution of (5) corresponds to subspace-sparse representations of the data points, which we use next to infer the clustering of the data. In Section 4, we study conditions under which the convex optimization program in (5) is guaranteed to recover a subspace-sparse representation of each data point.

2.2 Clustering using Sparse Coefficients

After solving the proposed optimization program in (5), we obtain a sparse representation for each data point whose nonzero elements ideally correspond to points from the same subspace. The next step of the algorithm is to infer the segmentation of the data into different subspaces using the sparse coefficients.

To address this problem, we build a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$, where $\mathcal{V}$ denotes the set of N nodes of the graph corresponding to N data points and $\mathcal{E} \subseteq \mathcal{V}\times\mathcal{V}$ denotes the set of edges between nodes. $\boldsymbol{W} \in \mathbb{R}^{N\times N}$ is a symmetric non-negative similarity matrix representing the weights of the edges, i.e., node i is connected to node j by an edge whose weight is equal to w_{ij}. An ideal similarity matrix $\boldsymbol{W}$, hence an ideal similarity graph $\mathcal{G}$, is one in which nodes that correspond to points from the same subspace are connected to each other and there are no edges between nodes that correspond to points in different subspaces.

Note that the sparse optimization program ideally recovers to a subspace-sparse representation of each point, i.e., a representation whose nonzero elements correspond to points from the same subspace of the given data point. This provides an immediate choice of the similarity matrix as $\boldsymbol{W} = |\boldsymbol{C}| + |\boldsymbol{C}|^\top$. In other words, each node i connects itself to a node j by an edge whose weight is equal to $|c_{ij}| + |c_{ji}|$. The reason for the symmetrization is that, in general, a data point $\boldsymbol{y}_i \in \mathcal{S}_\ell$ can write itself as a linear combination of some points including $\boldsymbol{y}_j \in \mathcal{S}_\ell$. However, $\boldsymbol{y}_j$ may not necessarily choose $\boldsymbol{y}_i$ in its sparse representation. By this particular choice of the weight, we make sure that nodes i and j get connected to each other if either $\boldsymbol{y}_i$ or $\boldsymbol{y}_j$ is in the sparse representation of the other.[2]

The similarity graph built this way has ideally n connected components corresponding to the n subspaces, i.e.,

$$\boldsymbol{W} = \begin{bmatrix}\boldsymbol{W}_1 & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{W}_n\end{bmatrix}\boldsymbol{\Gamma}, \tag{6}$$

where $\boldsymbol{W}_\ell$ is the similarity matrix of data points in $\mathcal{S}_\ell$. Clustering of data into subspaces follows then by applying spectral clustering [26] to the graph $\mathcal{G}$. More specifically, we obtain the clustering of data by applying the Kmeans algorithm [11] to the normalized rows of a matrix whose columns are the n bottom eigenvectors of the symmetric normalized Laplacian matrix of the graph.

Remark 1: An optional step prior to building the similarity graph is to normalize the sparse coefficients as $\boldsymbol{c}_i \leftarrow \boldsymbol{c}_i / \|\boldsymbol{c}_i\|_\infty$. This helps to better deal with different norms of data points. More specifically, if a data point with a large Euclidean norm selects a few points with small Euclidean norms, then the values of the nonzero coefficients will generally be large. On the other hand, if a data point with a small Euclidean norm selects a few points with large Euclidean norms, then the values of the nonzero coefficients will generally be small. Since spectral clustering puts more emphasis on keeping the stronger connections in the graph, by the normalization step we make sure that the largest edge weights for all the nodes are of the same scale.

Algorithm 1 summarizes the SSC algorithm. Note that an advantage of spectral clustering, which will be shown in the experimental results, is that it provides robustness with respect to a few errors in the sparse representations of the data points. In other words, as long as edges between points in different subspaces are weak, spectral clustering can find the correct segmentation.

2. To obtain a symmetric similarity matrix, one can directly impose the constraint of $\boldsymbol{C} = \boldsymbol{C}^\top$ in the optimization program. However, this results in increasing the complexity of the optimization program and, in practice, does not perform better than the post-symmetrization of $\boldsymbol{C}$, as described above. See also [52] for other processing approaches of the similarity matrix.

Algorithm 1 : Sparse Subspace Clustering (SSC)

Input: A set of points $\{\boldsymbol{y}_i\}_{i=1}^N$ lying in a union of n linear subspaces $\{\mathcal{S}_i\}_{i=1}^n$.

1: Solve the sparse optimization program (5) in the case of uncorrupted data or (13) in the case of corrupted data.
2: Normalize the columns of $\boldsymbol{C}$ as $\boldsymbol{c}_i \leftarrow \frac{\boldsymbol{c}_i}{\|\boldsymbol{c}_i\|_\infty}$.
3: Form a similarity graph with N nodes representing the data points. Set the weights on the edges between the nodes by $\boldsymbol{W} = |\boldsymbol{C}| + |\boldsymbol{C}|^\top$.
4: Apply spectral clustering [26] to the similarity graph.

Output: Segmentation of the data: $\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_n$.

Remark 2: In principle, SSC does not need to know the number of subspaces. More specifically, under the conditions of the theoretical results in Section 4, in the similarity graph there will be no connections between points in different subspaces. Thus, one can determine the number of subspaces by finding the number of graph components, which can be obtained by analyzing the eigenspectrum of the Laplacian matrix of $\mathcal{G}$ [27]. However, when there are connections between points in different subspaces, other model selection techniques should be employed [53].

3 Practical Extensions

In real-world problems, data are often corrupted by noise and sparse outlying entries due to measurement/process noise and ad-hoc data collection techniques. In such cases, the data do not lie perfectly in a union of subspaces. For instance, in the motion segmentation problem, because of the malfunctioning of the tracker, feature trajectories can be corrupted by noise or can have entries with large errors [21]. Similarly, in clustering of human faces, images can be corrupted by errors due to specularities, cast shadows, and occlusions [54]. On the other hand, data points may have missing entries, e.g., when the tracker loses track of some feature points in a video due to occlusions [55]. Finally, data may lie in a union of affine subspaces, a more general model which includes linear subspaces as a particular case.

In this section, we generalize the SSC algorithm for clustering data lying perfectly in a union of linear subspaces, to deal with the aforementioned challenges. Unlike state-of-the-art methods, which require to run a separate algorithm first to correct the errors in the data [21], [55], we deal with these problems in a unified framework by incorporating a model for the corruption into the sparse optimization program. Thus, the sparse coefficients again encode information about memberships of data to subspaces, which are used in a spectral clustering framework, as before.

3.1 Noise and Sparse Outlying Entries

In this section, we consider clustering of data points that are contaminated with sparse outlying entries and noise. Let

$$\boldsymbol{y}_i = \boldsymbol{y}_i^0 + \boldsymbol{e}_i^0 + \boldsymbol{z}_i^0 \tag{7}$$

be the i-th data point that is obtained by corrupting an error-free point $\boldsymbol{y}_i^0$, which perfectly lies in a subspace, with a vector of sparse outlying entries $\boldsymbol{e}_i^0 \in \mathbb{R}^D$ that has only a few large nonzero elements, i.e., $\|\boldsymbol{e}_i^0\|_0 \leq k$ for some integer k, and with a noise $\boldsymbol{z}_i^0 \in \mathbb{R}^D$ whose norm is bounded as $\|\boldsymbol{z}_i^0\|_2 \leq \zeta$ for some $\zeta > 0$. Since error-free data points perfectly lie in a union of subspaces, using the self-expressiveness property, we can reconstruct $\boldsymbol{y}_i^0 \in \mathcal{S}_\ell$ in terms of other error-free points as

$$\boldsymbol{y}_i^0 = \sum_{j \neq i} c_{ij} \boldsymbol{y}_j^0. \tag{8}$$

Note that the above equation has a sparse solution since $\boldsymbol{y}_i^0$ can be expressed as a linear combination of at most d_ℓ other points from $\mathcal{S}_\ell$. Rewriting $\boldsymbol{y}_i^0$ using (7) in terms of the corrupted point $\boldsymbol{y}_i$, the sparse outlying entries vector $\boldsymbol{e}_i^0$, and the noise vector $\boldsymbol{z}_i^0$ and substituting it into (8), we obtain

$$\boldsymbol{y}_i = \sum_{j \neq i} c_{ij} \boldsymbol{y}_j + \boldsymbol{e}_i + \boldsymbol{z}_i, \tag{9}$$

where the vectors $\boldsymbol{e}_i \in \mathbb{R}^D$ and $\boldsymbol{z}_i \in \mathbb{R}^D$ are defined as

$$\boldsymbol{e}_i \triangleq \boldsymbol{e}_i^0 - \textstyle\sum_{j \neq i} c_{ij} \boldsymbol{e}_j^0, \tag{10}$$

$$\boldsymbol{z}_i \triangleq \boldsymbol{z}_i^0 - \textstyle\sum_{j \neq i} c_{ij} \boldsymbol{z}_j^0. \tag{11}$$

Since (8) has a sparse solution $\boldsymbol{c}_i$, $\boldsymbol{e}_i$ and $\boldsymbol{z}_i$ also correspond to vectors of sparse outlying entries and noise, respectively. More precisely, when a few c_{ij} are nonzero, $\boldsymbol{e}_i$ is a vector of sparse outlying entries since it is a linear combination of a few vectors of outlying entries in (10). Similarly, when a few c_{ij} are nonzero and do not have significantly large magnitudes[3], $\boldsymbol{z}_i$ is a vector of noise since it is linear combination of a few noise vectors in (11).

Collecting $\boldsymbol{e}_i$ and $\boldsymbol{z}_i$ as columns of the matrices $\boldsymbol{E}$ and $\boldsymbol{Z}$, respectively, we can rewrite (9) in matrix form as

$$\boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C} + \boldsymbol{E} + \boldsymbol{Z}, \quad \operatorname{diag}(\boldsymbol{C}) = \boldsymbol{0}. \tag{12}$$

Our objective is then to find a solution $(\boldsymbol{C}, \boldsymbol{E}, \boldsymbol{Z})$ for (12), where $\boldsymbol{C}$ corresponds to a sparse coefficient matrix, $\boldsymbol{E}$ corresponds to a matrix of sparse outlying entries, and $\boldsymbol{Z}$ is a noise matrix. To do so, we propose to solve the following optimization program

$$\begin{aligned} \min \quad & \|\boldsymbol{C}\|_1 + \lambda_e \|\boldsymbol{E}\|_1 + \frac{\lambda_z}{2} \|\boldsymbol{Z}\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C} + \boldsymbol{E} + \boldsymbol{Z}, \ \operatorname{diag}(\boldsymbol{C}) = \boldsymbol{0}, \end{aligned} \tag{13}$$

where the ℓ_1-norm promotes sparsity of the columns of $\boldsymbol{C}$ and $\boldsymbol{E}$, while the Frobenius norm promotes having small entries in the columns of $\boldsymbol{Z}$. The two parameters $\lambda_e > 0$ and $\lambda_z > 0$ balance the three terms in the objective function. Note that the optimization program in (13) is convex with respect to the optimization variables $(\boldsymbol{C}, \boldsymbol{E}, \boldsymbol{Z})$, hence, can be solved efficiently using convex programming tools.

When data are corrupted only by noise, we can eliminate $\boldsymbol{E}$ from the optimization program in (13). On the other hand, when the data are corrupted only by sparse outlying entries, we can eliminate $\boldsymbol{Z}$ in (13). In practice, however, $\boldsymbol{E}$ can also deal with small errors due to noise. The following proposition suggests setting $\lambda_z = \alpha_z/\mu_z$ and $\lambda_e = \alpha_e/\mu_e$, where $\alpha_z, \alpha_e > 1$ and

$$\mu_z \triangleq \min_i \max_{j \neq i} |\boldsymbol{y}_i^\top \boldsymbol{y}_j|, \quad \mu_e \triangleq \min_i \max_{j \neq i} \|\boldsymbol{y}_j\|_1. \tag{14}$$

The proofs of all theoretical results in the paper are provided in the supplementary material.

Proposition 1: Consider the optimization program (13). Without the term $\boldsymbol{Z}$, if $\lambda_e \leq 1/\mu_e$, then there exists at least

3. One can show that, under broad conditions, sum of $|c_{ij}|$ is bounded above by the square root of the dimension of the underlying subspace of $\boldsymbol{y}_i$. Theoretical guarantees of the proposed optimization program in the case of corrupted data is the subject of the current research.

one data point $\boldsymbol{y}_\ell$ for which in the optimal solution we have $(\boldsymbol{c}_\ell, \boldsymbol{e}_\ell) = (\boldsymbol{0}, \boldsymbol{y}_\ell)$. Also, without the term $\boldsymbol{E}$, if $\lambda_z \leq 1/\mu_z$, then there exists at least one data point $\boldsymbol{y}_\ell$ for which $(\boldsymbol{c}_\ell, \boldsymbol{z}_\ell) = (\boldsymbol{0}, \boldsymbol{y}_\ell)$.

After solving the proposed optimization programs, we use $\boldsymbol{C}$ to build a similarity graph and infer the clustering of data using spectral clustering. Thus, by incorporating the corruption model of data into the sparse optimization program, we can deal with clustering of corrupted data, as before, without explicitly running a separate algorithm to correct the errors in the data [21], [55].

3.2 Missing Entries

We consider now the clustering of incomplete data, where some of the entries of a subset of the data points are missing. Note that when only a small fraction of the entries of each data point is missing, clustering of incomplete data can be cast as clustering of data with sparse outlying entries. More precisely, one can fill in the missing entries of each data point with random values, hence obtain data points with sparse outlying entries. Then clustering of the data follows by solving (13) and applying spectral clustering to the graph built using the obtained sparse coefficients. However, the drawback of this approach is that it disregards the fact that we know the locations of the missing entries in the data matrix.

It is possible, in some cases, to cast the clustering of data with missing entries as clustering of complete data. To see this, consider a collection of data points $\{\boldsymbol{y}_i\}_{i=1}^N$ in $\mathbb{R}^D$. Let $J_i \subset \{1, \ldots, D\}$ denote indices of the known entries of $\boldsymbol{y}_i$ and define $J \triangleq \bigcap_{i=1}^N J_i$. Thus, for every index in J, all data points have known entries. When the size of J, denoted by $|J|$, is not small relative to the ambient space dimension, D, we can project the data, hence, the original subspaces, into a subspace spanned by the columns of the identity matrix indexed by J and apply the SSC algorithm to the obtained complete data. In other words, we can only keep the rows of $\boldsymbol{Y}$ indexed by J, obtain a new data matrix of complete data $\bar{\boldsymbol{Y}} \in \mathbb{R}^{|J| \times N}$, and solve the sparse optimization program (13). We can then infer the clustering of the data by applying spectral clustering to the graph built using the sparse coefficient matrix. Note that the approach described above is based on the assumption that J is nonempty. Addressing the problem of subspace clustering with missing entries when J is empty or has a small size is the subject of the future research.

3.3 Affine Subspaces

In some real-world problems, the data lie in a union of affine rather than linear subspaces. For instance, the motion segmentation problem involves clustering of data that lie in a union of 3-dimensional affine subspaces [2], [55]. A naive way to deal with this case is to ignore the affine structure of the data and perform clustering as in the case of linear subspaces. This comes from the fact that a d_ℓ-dimensional affine subspace $\mathcal{S}_\ell$ can be considered as a subset of a $(d_\ell + 1)$-dimensional linear subspace that includes $\mathcal{S}_\ell$ and the origin. However, this has the drawback of possibly increasing the dimension of the intersection of two subspaces, which in some cases can result in indistinguishability of subspaces from each other. For example, two different lines $x = -1$ and $x = +1$ in the x-y plane form the same 2-dimensional linear subspace after including the origin, hence become indistinguishable.

To directly deal with affine subspaces, we use the fact that any data point $\boldsymbol{y}_i$ in an affine subspace $\mathcal{S}_\ell$ of dimension d_ℓ can be written as an affine combination of $d_\ell + 1$ other points from $\mathcal{S}_\ell$. In other words, a sparse solution of

$$\boldsymbol{y}_i = \boldsymbol{Y}\boldsymbol{c}_i, \quad \boldsymbol{1}^\top \boldsymbol{c}_i = 1, \;\; c_{ii} = 0, \tag{15}$$

corresponds to $d_\ell + 1$ other points that belong to $\mathcal{S}_\ell$ containing $\boldsymbol{y}_i$. Thus, to cluster data points lying close to a union of affine subspaces, we propose to solve the sparse optimization program

$$\begin{aligned} \min \;\; & \|\boldsymbol{C}\|_1 + \lambda_e \|\boldsymbol{E}\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{Z}\|_F^2 \\ \text{s.t.} \;\; & \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C} + \boldsymbol{E} + \boldsymbol{Z}, \;\; \boldsymbol{1}^\top \boldsymbol{C} = \boldsymbol{1}^\top, \;\; \operatorname{diag}(\boldsymbol{C}) = \boldsymbol{0}, \end{aligned} \tag{16}$$

which, in comparison to (13) for the case of linear subspaces, includes additional linear equality constraints. Note that (16) can deal with linear subspaces as well since a linear subspace is also an affine subspace.

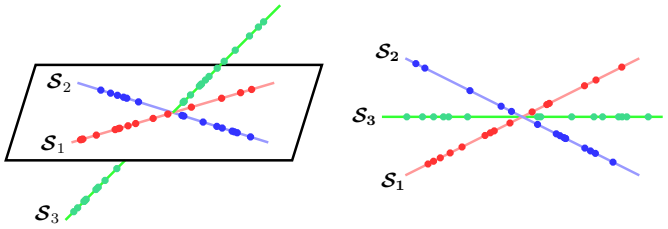


Fig. 4. Left: the three 1-dimensional subspaces are independent as they span the 3-dimensional space and the sum of their dimensions is also 3. Right: the three 1-dimensional are disjoint as any two subspaces intersect at the origin.

4 Subspace-Sparse Recovery Theory

The underlying assumption for the success of the SSC algorithm is that the proposed optimization program recovers a subspace-sparse representation of each data point, i.e., a representation whose nonzero elements correspond to the subspace of the given point. In this section, we investigate conditions under which, for data points that lie in a union of linear subspaces, the sparse optimization program in (4) recovers subspace-sparse representations of data points. We investigate recovery conditions for two classes of subspace arrangements: *independent* and *disjoint* subspace models [36].

Definition 1: *A collection of subspaces $\{\mathcal{S}_i\}_{i=1}^n$ is said to be* independent *if $\dim(\oplus_{i=1}^n \mathcal{S}_i) = \sum_{i=1}^n \dim(\mathcal{S}_i)$, where $\oplus$ denotes the direct sum operator.*

As an example, the three 1-dimensional subspaces shown in Figure 4 (left) are independent since they span a 3-dimensional space and the sum of their dimensions is also 3. On the other hand, the subspaces shown in Figure 4 (right) are not independent since they span a 2-dimensional space while the sum of their dimensions is 3.

Definition 2: *A collection of subspaces $\{\mathcal{S}_i\}_{i=1}^n$ is said to be* disjoint *if every pair of subspaces intersect only at the origin. In other words, for every pair of subspaces we have $\dim(\mathcal{S}_i \oplus \mathcal{S}_j) = \dim(\mathcal{S}_i) + \dim(\mathcal{S}_j)$.*

As an example, both subspace arrangements shown in Figure 4 are disjoint since each pair of subspaces intersect at the origin. Note that, based on the above definitions, the notion of disjointness is weaker than independence as an independent subspace model is always disjoint while the converse is not necessarily true. An

important notion that can be used to characterize two disjoint subspaces is the smallest principal angle, defined as follows.

Definition 3: *The* smallest principal angle *between two subspaces $\mathcal{S}_i$ and $\mathcal{S}_j$, denoted by θ_{ij}, is defined as*

$$\cos(\theta_{ij}) \triangleq \max_{\boldsymbol{v}_i \in \mathcal{S}_i, \boldsymbol{v}_j \in \mathcal{S}_j} \frac{\boldsymbol{v}_i^\top \boldsymbol{v}_j}{\|\boldsymbol{v}_i\|_2 \|\boldsymbol{v}_j\|_2}. \tag{17}$$

Note that two disjoint subspaces intersect at the origin, hence their smallest principal angle is greater than zero and $\cos(\theta_{ij}) \in [0, 1)$.

4.1 Independent Subspace Model

In this section, we consider data points that lie in a union of independent subspaces, which is the underlying model of many subspace clustering algorithms. We show that the ℓ_1-minimization program in (4) and more generally the ℓ_q-minimization in (3) for $q < \infty$ always recover subspace-sparse representations of the data points. More specifically, we show the following result.

Theorem 1: *Consider a collection of data points drawn from n independent subspaces $\{\mathcal{S}_i\}_{i=1}^n$ of dimensions $\{d_i\}_{i=1}^n$. Let $\boldsymbol{Y}_i$ denote N_i data points in $\mathcal{S}_i$, where $\operatorname{rank}(\boldsymbol{Y}_i) = d_i$, and let $\boldsymbol{Y}_{-i}$ denote data points in all subspaces except $\mathcal{S}_i$. Then, for every $\mathcal{S}_i$ and every nonzero $\boldsymbol{y}$ in $\mathcal{S}_i$, the ℓ_q-minimization program*

$$\begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{c}_-^* \end{bmatrix} = \operatorname{argmin} \left\| \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix} \right\|_q \quad \text{s.t.} \quad \boldsymbol{y} = [\boldsymbol{Y}_i \;\; \boldsymbol{Y}_{-i}] \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix}, \tag{18}$$

for $q < \infty$, recovers a subspace-sparse representation, i.e., $\boldsymbol{c}^* \neq \boldsymbol{0}$ and $\boldsymbol{c}_-^* = \boldsymbol{0}$.

Note that the subspace-sparse recovery holds without any assumption on the distribution of the data points in each subspace, other than $\operatorname{rank}(\boldsymbol{Y}_i) = d_i$. This comes at the price of having a more restrictive model for the subspace arrangements. Next, we will show that for the more general class of disjoint subspaces, under appropriate conditions on the relative configuration of the subspaces as well as the distribution of the data in each subspace, the ℓ_1-minimization in (4) recovers subspace-sparse representations of the data points.

4.2 Disjoint Subspace Model

We consider now the more general class of disjoint subspaces and investigate conditions under which the optimization program in (4) recovers a subspace-sparse representation of each data point. To that end, we consider a vector $\boldsymbol{x}$ in the intersection of $\mathcal{S}_i$ with $\oplus_{j \neq i} \mathcal{S}_j$ and let the optimal solution of the ℓ_1-minimization when we restrict the dictionary to data points from $\mathcal{S}_i$ be

$$\boldsymbol{a}_i = \operatorname{argmin} \|\boldsymbol{a}\|_1 \quad \text{s.t.} \quad \boldsymbol{x} = \boldsymbol{Y}_i \, \boldsymbol{a}. \tag{19}$$

We also let the optimal solution of the ℓ_1-minimization when we restrict the dictionary to points from all subspaces except $\mathcal{S}_i$ be

$$\boldsymbol{a}_{-i} = \operatorname{argmin} \|\boldsymbol{a}\|_1 \quad \text{s.t.} \quad \boldsymbol{x} = \boldsymbol{Y}_{-i} \, \boldsymbol{a}.^4 \tag{20}$$

We show in the supplementary material that the SSC algorithm succeeds in recovering subspace-sparse representations of data points in each $\mathcal{S}_i$, if for every nonzero $\boldsymbol{x}$ in the intersection of $\mathcal{S}_i$ with $\oplus_{j \neq i} \mathcal{S}_j$, the ℓ_1-norm of the solution of (19) is strictly smaller than the ℓ_1-norm of the solution of (20), i.e.,

$$\forall \, \boldsymbol{x} \in \mathcal{S}_i \cap (\oplus_{j \neq i} \mathcal{S}_j), \boldsymbol{x} \neq \boldsymbol{0} \implies \|\boldsymbol{a}_i\|_1 < \|\boldsymbol{a}_{-i}\|_1. \tag{21}$$

More precisely, we show the following result.

Theorem 2: *Consider a collection of data points drawn from n disjoint subspaces $\{\mathcal{S}_i\}_{i=1}^n$ of dimensions $\{d_i\}_{i=1}^n$. Let $\boldsymbol{Y}_i$ denote N_i data points in $\mathcal{S}_i$, where $\operatorname{rank}(\boldsymbol{Y}_i) = d_i$, and let $\boldsymbol{Y}_{-i}$ denote data points in all subspaces except $\mathcal{S}_i$. The ℓ_1-minimization*

$$\begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{c}_-^* \end{bmatrix} = \operatorname{argmin} \left\| \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix} \right\|_1 \quad \text{s.t.} \quad \boldsymbol{y} = [\boldsymbol{Y}_i \;\; \boldsymbol{Y}_{-i}] \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix}, \tag{22}$$

recovers a subspace-sparse representation of every nonzero $\boldsymbol{y}$ in $\mathcal{S}_i$, i.e., $\boldsymbol{c}^* \neq \boldsymbol{0}$ and $\boldsymbol{c}_-^* = \boldsymbol{0}$, if and only if (21) *holds.*

While the necessary and sufficient condition in (21) guarantees a successful subspace-sparse recovery via the ℓ_1-minimization program, it does not explicitly show the relationship between the subspace arrangements and the data distribution for the success of the ℓ_1-minimization program. To establish such a relationship, we show that $\|\boldsymbol{a}_i\|_1 \leq \beta_i$, where β_i depends on the singular values of data points in $\mathcal{S}_i$, and $\beta_{-i} \leq \|\boldsymbol{a}_{-i}\|_1$, where β_{-i} depends on the subspace angles between $\mathcal{S}_i$ and other subspaces. Then, the sufficient condition $\beta_i < \beta_{-i}$ establishes the relationship between the subspace angles and the data distribution under which the ℓ_1-minimization is successful in subspace-sparse recovery, since it implies that

$$\|\boldsymbol{a}_i\|_1 \leq \beta_i < \beta_{-i} \leq \|\boldsymbol{a}_{-i}\|_1, \tag{23}$$

i.e., the condition of Theorem 2 holds.

Theorem 3: *Consider a collection of data points drawn from n disjoint subspaces $\{\mathcal{S}_i\}_{i=1}^n$ of dimensions $\{d_i\}_{i=1}^n$. Let $\mathbb{W}_i$ be the set of all full-rank submatrices $\tilde{\boldsymbol{Y}}_i \in \mathbb{R}^{D \times d_i}$ of $\boldsymbol{Y}_i$, where $\operatorname{rank}(\boldsymbol{Y}_i) = d_i$. If the condition*

$$\max_{\tilde{\boldsymbol{Y}}_i \in \mathbb{W}_i} \sigma_{d_i}(\tilde{\boldsymbol{Y}}_i) > \sqrt{d_i} \, \|\boldsymbol{Y}_{-i}\|_{1,2} \max_{j \neq i} \cos(\theta_{ij}) \tag{24}$$

holds, then for every nonzero $\boldsymbol{y}$ in $\mathcal{S}_i$, the ℓ_1-minimization in (22) *recovers a subspace-sparse solution, i.e., $\boldsymbol{c}^* \neq \boldsymbol{0}$ and $\boldsymbol{c}_-^* = \boldsymbol{0}$.*[5]

Loosely speaking, the sufficient condition in Theorem 3 states that if the smallest principal angle between each $\mathcal{S}_i$ and any other subspace is larger than a certain value that depends on the data distribution in $\mathcal{S}_i$, then the subspace-sparse recovery holds. This bound can be rather high when the norms of the data points are oddly distributed, e.g., when the maximum norm of data points in $\mathcal{S}_i$ is much smaller than the maximum norm of data points in all other subspaces. Since the segmentation of the data does not change when data points are scaled, we can apply SSC to linear subspaces after normalizing the data points to have unit Euclidean norms. In this case, the sufficient condition in (24) reduces to

$$\max_{\tilde{\boldsymbol{Y}}_i \in \mathbb{W}_i} \sigma_{d_i}(\tilde{\boldsymbol{Y}}_i) > \sqrt{d_i} \max_{j \neq i} \cos(\theta_{ij}). \tag{25}$$

Remark 3: For independent subspaces, the intersection of a subspace with the direct sum of other subspaces is the origin, hence, the condition in (21) always holds. As a result, from Theorem 2, the ℓ_1-minimization always recovers subspace-sparse representations of data points in independent subspaces.

4. In fact, $\boldsymbol{a}_i$ and $\boldsymbol{a}_{-i}$ depend on $\boldsymbol{x}$, $\boldsymbol{Y}_i$, and $\boldsymbol{Y}_{-i}$. Since this dependence is clear from the context, we drop the arguments in $\boldsymbol{a}_i(\boldsymbol{x}, \boldsymbol{Y}_i)$ and $\boldsymbol{a}_{-i}(\boldsymbol{x}, \boldsymbol{Y}_{-i})$.

5. The induced norm $\|\boldsymbol{Y}_{-i}\|_{1,2}$ denotes the maximum ℓ_2-norm of the columns of $\boldsymbol{Y}_{-i}$.

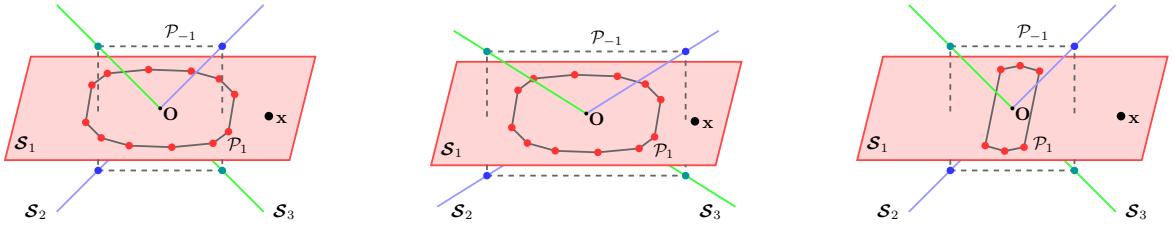


Fig. 5. Left: for any nonzero $\boldsymbol{x}$ in the intersection of $\mathcal{S}_1$ and $\mathcal{S}_2 \oplus \mathcal{S}_3$, the polytope $\alpha\mathcal{P}_1$ reaches $\boldsymbol{x}$ for a smaller α than $\alpha\mathcal{P}_{-1}$, hence, subspace-sparse recovery holds. Middle: when the subspace angle decreases, the polytope $\alpha\mathcal{P}_{-1}$ reaches $\boldsymbol{x}$ for a smaller α than $\alpha\mathcal{P}_1$. Right: when the distribution of the data in $\mathcal{S}_1$ becomes nearly degenerate, in this case close to a line, the polytope $\alpha\mathcal{P}_{-1}$ reaches $\boldsymbol{x}$ for a smaller α than $\alpha\mathcal{P}_1$. In both cases, in the middle and right, the subspace-sparse recovery does not hold for points at the intersecion.

Remark 4: The condition in (21) is closely related to the nullspace property in the sparse recovery literature [56], [57], [45], [58]. The key difference, however, is that we only require the inequality in (21) to hold for the optimal solutions of (19) and (20) instead of any feasible solution. Thus, while the inequality can be violated for many feasible solutions, it can still hold for the optimal solutions, guaranteeing successful subspace-sparse recovery from Theorem 2. Thus, our result can be thought of as a generalization of the nullspace property to the multi-subspace setting where the number of points in each subspace is arbitrary.

4.3 Geometric interpretation

In this section, we provide a geometric interpretation of the subspace-sparse recovery conditions in (21) and (24). To do so, it is necessary to recall the relationship between the ℓ_1-norm of the optimal solution of

$$\min \|\boldsymbol{a}\|_1 \quad \text{s.t.} \quad \boldsymbol{x} = \boldsymbol{B}\boldsymbol{a}, \tag{26}$$

and the symmetrized convex polytope of the columns of $\boldsymbol{B}$ [59]. More precisely, if we denote the columns of $\boldsymbol{B}$ by $\boldsymbol{b}_i$ and define the symmetrized convex hull of the columns of $\boldsymbol{B}$ by

$$\mathcal{P} \triangleq \text{conv}(\pm\boldsymbol{b}_1, \pm\boldsymbol{b}_2, \cdots), \tag{27}$$

then the ℓ_1-norm of the optimal solution of (26) corresponds to the smallest $\alpha > 0$ such that the scaled polytope $\alpha\mathcal{P}$ reaches $\boldsymbol{x}$ [59]. Let us denote the symmetrized convex polytopes of $\boldsymbol{Y}_i$ and $\boldsymbol{Y}_{-i}$ by $\mathcal{P}_i$ and $\mathcal{P}_{-i}$, respectively. Then the condition in (21) has the following geometric interpretation:

> *the subspace-sparse recovery in $\mathcal{S}_i$ holds if and only if for any nonzero $\boldsymbol{x}$ in the intersection of $\mathcal{S}_i$ and $\oplus_{j\neq i}\mathcal{S}_j$, $\alpha\mathcal{P}_i$ reaches $\boldsymbol{x}$ before $\alpha\mathcal{P}_{-i}$, i.e., for a smaller α.*

As shown in the left plot of Figure 5, for $\boldsymbol{x}$ in the intersection of $\mathcal{S}_1$ and $\mathcal{S}_2 \oplus \mathcal{S}_3$, the polytope $\alpha\mathcal{P}_1$ reaches $\boldsymbol{x}$ before $\alpha\mathcal{P}_{-1}$, hence the subspace-sparse recovery condition holds. On the other hand, when the principal angles between $\mathcal{S}_1$ and other subspaces decrease, as shown in the middle plot of Figure 5, the subspace-sparse recovery condition does not hold since the polytope $\alpha\mathcal{P}_{-1}$ reaches $\boldsymbol{x}$ before $\alpha\mathcal{P}_1$. Also, as shown in the right plot of Figure 5, when the distribution of the data in $\mathcal{S}_1$ becomes nearly degenerate, in this case close to a 1-dimensional subspace orthogonal to the direction of $\boldsymbol{x}$, then the subspace-sparse recovery condition does not hold since $\alpha\mathcal{P}_{-1}$ reaches $\boldsymbol{x}$ before $\alpha\mathcal{P}_1$. Note that the sufficient condition in (24) translates the relationship between the polytopes, mentioned above, explicitly in terms of a relationship between the subspace angles and the singular values of the data.

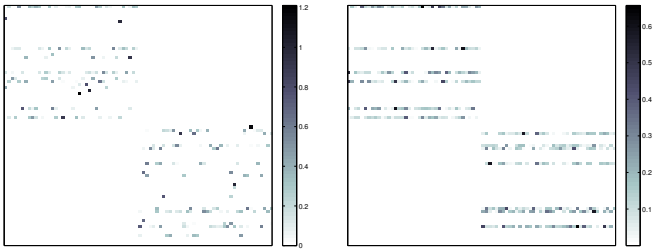


Fig. 6. Coefficient matrix obtained from the solution of (30) for data points in two subspaces. Left: $\lambda_r = 0$. Right: $\lambda_r = 10$. Increasing λ_r results in concentration of the nonzero elements in a few rows of the coefficient matrix, hence choosing a few common data points.

5 Graph Connectivity

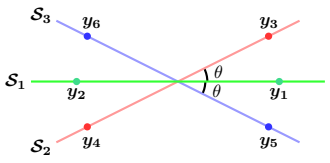
In the previous section, we studied conditions under which the proposed ℓ_1-minimization program recovers subspace-sparse representations of data points. As a result, in the similarity graph, the points that lie in different subspaces do not get connected to each other. On the other hand, our extensive experimental results on synthetic and real data show that data points in the same subspace always form a connected component of the graph, hence, for n subspaces the similarity graph has n connected components. [60] has theoretically verified the connectivity of points in the same subspace for 2 and 3 dimensional subspaces. However, it has shown that, for subspaces of dimensions greater than or equal to 4, under odd distribution of the data, it is possible that points in the same subspace form multiple components of the graph.

In this section, we consider a regularization term in the sparse optimization program that promotes connectivity of the points within each subspace.[6] We use the idea that if data points in each subspace choose a few *common* points from the same subspace in their sparse representations, then they form a single component of the similarity graph. Thus, we add to the sparse optimization program the regularization term

$$\|\boldsymbol{C}\|_{r,0} \triangleq \sum_{i=1}^{N} \text{I}(\|\boldsymbol{c}^i\|_2 > 0), \tag{28}$$

where $\text{I}(\cdot)$ denotes the indicator function and $\boldsymbol{c}^i$ denotes the i-th row of $\boldsymbol{C}$. Hence, minimizing (28) corresponds to minimizing the number of nonzero rows of $\boldsymbol{C}$ [61], [62], [63], i.e., choosing a few common data points in the sparse representation of each point.

6. Another approach to deal with the connectivity issue is to analyze the subspaces corresponding to the components of the graph and merge the components whose associated subspaces have a small distance from each other, i.e., have a small principal angle. However, the result can be sensitive to the choice of the dimension of the subspaces to fit to each component as well as the threshold value on the principal angles to merge the subspaces.



$$\boldsymbol{C}_1 = \left(\begin{array}{cc|cc|cc} 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \end{array}\right) \qquad \boldsymbol{C}_2 = \left(\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline +\frac{1}{4\cos\theta} & -\frac{1}{4\cos\theta} & 0 & -1 & 0 & 0 \\ -\frac{1}{4\cos\theta} & +\frac{1}{4\cos\theta} & -1 & 0 & 0 & 0 \\ \hline +\frac{1}{4\cos\theta} & -\frac{1}{4\cos\theta} & 0 & 0 & 0 & -1 \\ -\frac{1}{4\cos\theta} & +\frac{1}{4\cos\theta} & 0 & 0 & -1 & 0 \end{array}\right)$$

Fig. 7. Left: three 1-dimensional subspaces in $\mathbb{R}^2$ with normalized data points. Middle: $\boldsymbol{C}_1$ corresponds to the solution of (30) for $\lambda_r = 0$. The similarity graph of $\boldsymbol{C}_1$ has three components corresponding to the three subspaces. Right: $\boldsymbol{C}_2$ corresponds to the solution of (30) for $\lambda_r \to +\infty$ and $\theta \in (0, \frac{4\pi}{10})$. The similarity graph of $\boldsymbol{C}_2$ has only one connected component.

Since a minimization problem that involves (28) is in general NP-hard, we consider its convex relaxation as

$$\|\boldsymbol{C}\|_{r,1} \triangleq \sum_{i=1}^{N} \|\boldsymbol{c}^i\|_2. \tag{29}$$

Thus, to increase the connectivity of data points from the same subspace in the similarity graph, we propose to solve

$$\min \|\boldsymbol{C}\|_1 + \lambda_r \|\boldsymbol{C}\|_{r,1} \quad \text{s.t.} \quad \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C}, \ \operatorname{diag}(\boldsymbol{C}) = \boldsymbol{0}, \tag{30}$$

where $\lambda_r > 0$ sets the trade-off between the sparsity of the solution and the connectivity of the graph. Figure 6 shows how adding this regularization term promotes selecting common points in sparse representations. The following example demonstrates the reason for using the row-sparsity term as a regularizer but not as an objective function instead of the ℓ_1-norm.

Example 1: Consider the three 1-dimensional subspaces in $\mathbb{R}^2$, shown in Figure 7, where the data points have unit Euclidean norms and the angle between $\mathcal{S}_1$ and $\mathcal{S}_2$ as well as between $\mathcal{S}_1$ and $\mathcal{S}_3$ is equal to θ. Note that in this example, the sufficient condition in (24) holds for all values of $\theta \in (0, \frac{\pi}{2})$. As a result, the solution of (30) with $\lambda_r = 0$ recovers a subspace-sparse representation for each data point, which in this example is uniquely given by $\boldsymbol{C}_1$ shown in Figure 7. Hence, the similarity graph has exactly 3 connected components corresponding to the data points in each subspace. Another feasible solution of (30) is given by $\boldsymbol{C}_2$, shown in Figure 7, where the points in $\mathcal{S}_1$ choose points from $\mathcal{S}_2$ and $\mathcal{S}_3$ in their representations. Hence, the similarity graph has only one connected component. Note that for a large range of subspace angles $\theta \in (0, \frac{4\pi}{10})$ we have

$$\|\boldsymbol{C}_2\|_{r,1} = \sqrt{16 + 2/\cos^2(\theta)} < \|\boldsymbol{C}_1\|_{r,1} = 6. \tag{31}$$

As a result, for large values of λ_r, i.e., when we only minimize the second term of the objective function in (30), we cannot recover subspace-sparse representations of the data points. This suggests using the row-sparsity regularizer with a small value of λ_r.

6 Experiments with Synthetic Data

In Section 4, we showed that the success of the ℓ_1-minimization for subspace-sparse recovery depends on the principal angles between subspaces and the distribution of the data in each subspace. In this section, we verify this relationship through experiments on synthetic data.

We consider three disjoint subspaces $\{\mathcal{S}_i\}_{i=1}^3$ of the same dimension d embedded in the D-dimensional ambient space. To make the problem hard enough so that every data point in a subspace can also be reconstructed as a linear combination of points in other subspaces, we generate subspace bases $\{\boldsymbol{U}_i \in \mathbb{R}^{D\times d}\}_{i=1}^3$ such that each subspace lies in the direct sum of the other two subspaces, i.e., $\operatorname{rank}([\boldsymbol{U}_1 \ \boldsymbol{U}_2 \ \boldsymbol{U}_3]) = 2d$. In addition, we generate the subspaces such that the smallest principal angles θ_{12} and θ_{23} are equal to θ. Thus, we can verify the effect of the smallest principal angle in the subspace-sparse recovery by changing the value of θ. To investigate the effect of the data distribution in the subspace-sparse recovery, we generate the same number of data points, N_g, in each subspace at random and change the value of N_g. Typically, as the number of data points in a subspace increases, the probability of the data being close to a degenerate subspace decreases.[7]

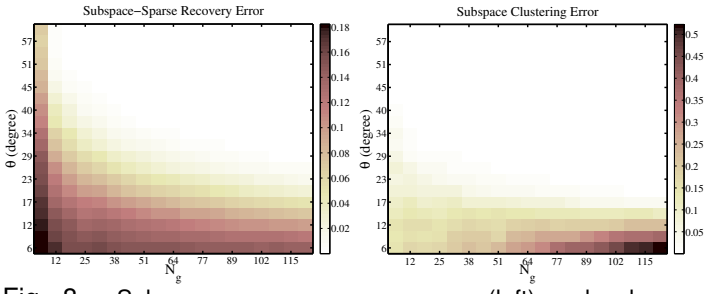


Fig. 8. Subspace-sparse recovery error (left) and subspace clustering error (right) for three disjoint subspaces. Increasing the number of points or smallest principal angle decreases the errors.

After generating three d-dimensional subspaces associated to (θ, N_g), we solve the ℓ_1-minimization program in (4) for each data point and measure two different errors. First, denoting the sparse representation of $\boldsymbol{y}_i \in \mathcal{S}_{k_i}$ by $\boldsymbol{c}_i^\top \triangleq [\boldsymbol{c}_{i1}^\top \ \boldsymbol{c}_{i2}^\top \ \boldsymbol{c}_{i3}^\top]$, with $\boldsymbol{c}_{ij}$ corresponding to points in $\mathcal{S}_j$, we measure the *subspace-sparse recovery error* by

$$\text{ssr error} = \frac{1}{3N_g} \sum_{i=1}^{3N_g} (1 - \frac{\|\boldsymbol{c}_{ik_i}\|_1}{\|\boldsymbol{c}_i\|_1}) \in [0, 1], \tag{32}$$

where each term inside the summation indicates the fraction of the ℓ_1-norm of $\boldsymbol{c}_i$ that comes from points in other subspaces. The error being zero corresponds to $\boldsymbol{y}_i$ choosing points only in its own subspace, while the error being equal to one corresponds to $\boldsymbol{y}_i$ choosing points from other subspaces. Second, after building the similarity graph using the sparse coefficients and applying spectral clustering, we measure the *subspace clustering error* by

$$\text{subspace clustering error} = \frac{\#\text{ of misclassified points}}{\text{total }\#\text{ of points}}. \tag{33}$$

In our experiments, we set the dimension of the ambient space to $D = 50$. We change the smallest principal angle between subspaces as $\theta \in [6, 60]$ degrees and change the number of points in each subspace as $N_g \in [d+1, 32d]$. For each pair (θ, N_g) we

7. To remove the effect of different scalings of data points, i.e., to consider only the effect of the principal angle and number of points, we normalize the data points.

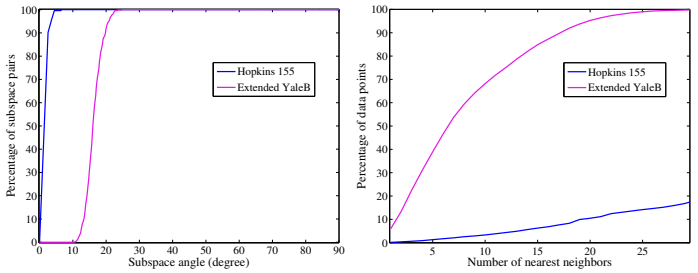


Fig. 9. Left: percentage of pairs of subspaces whose smallest principal angle is smaller than a given value. Right: average percentage of data points in pairs of subspaces that have one or more of their K-nearest neighbors in the other subspace.

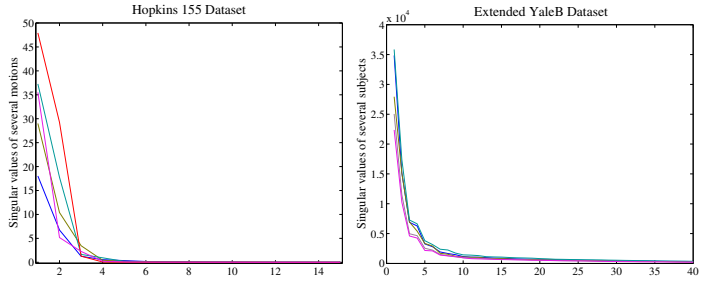


Fig. 10. Left: singular values of several motions in the Hopkins 155 dataset. Each motion corresponds to a subspace of dimension at most 4. Right: singular values of several faces in the Extended Yale B dataset. Each subject corresponds to a subspace of dimension around 9.

compute the average of the errors in (32) and (33) over 100 trials (randomly generated subspaces and data points). The results for $d = 4$ are shown in Figure 8. Note that when either θ or N_g is small, both the subspace-sparse recovery error and the clustering error are large, as predicted by our theoretical analysis. On the other hand, when θ or N_g increases, the errors decrease, and for (θ, N_g) sufficiently large we obtain zero errors. The results also verify that the success of the clustering relies on the success of the ℓ_1-minimization in recovering subspace-sparse representations of data points. Note that for small θ as we increase N_g, the subspace-sparse recovery error is large and slightly decreases, while the clustering error increases. This is due to the fact that increasing the number of points, the number of undesirable edges between different subspaces in the similarity graph increases, making the spectral clustering more difficult. Note also that, for the values of (θ, N_g) where the subspace-sparse recovery error is zero, i.e., points in different subspaces are not connected to each other in the similarity graph, the clustering error is also zero. This implies that, in such cases, the similarity graph has exactly three connected components, i.e., data points in the same subspace form a single component of the graph.

7 Experiments with Real Data

In this section, we evaluate the performance of the SSC algorithm in dealing with two real-world problems: segmenting multiple motions in videos (Fig. 1) and clustering images of human faces (Fig. 2). We compare the performance of SSC with the best state-of-the-art subspace clustering algorithms: LSA [22], SCC [28], LRR [38], and LRSC [41].

Implementation details. We implement the SSC optimization algorithm in (13) using an Alternating Direction Method of Multipliers (ADMM) framework [50], [64] whose derivation is provided in the supplementary material. For the motion segmentation experiments, we use the noisy variation of the optimization program (13), i.e., without the term $\boldsymbol{E}$, with the affine constraint, and choose $\lambda_z = 800/\mu_z$ in all the experiments (μ_z is defined in (14)). For the face clustering experiments, we use the sparse outlying entries variation of the optimization program (13), i.e., without the term $\boldsymbol{Z}$, and choose $\lambda_e = 20/\mu_e$ in all the experiments (μ_e is defined in (14)). It is also worth mentioning that SSC performs better with the ADMM approach than with general interior point solvers [49], which typically return many small nonzero coefficients, degrading the spectral clustering result.

For the state-of-the-art algorithms, we use the codes provided by their authors. For LSA, we use $K = 8$ nearest neighbors and dimension $d = 4$, to fit local subspaces, for motion segmentation and use $K = 7$ nearest neighbors and dimension $d = 5$ for face clustering. For SCC, we use dimension $d = 3$, for the subspaces, for motion segmentation and $d = 9$ for face clustering. For LRR, we use $\lambda = 4$ for motion segmentation and $\lambda = 0.18$ for face clustering. Note that the LRR algorithm according to [38], similar to SSC, applies spectral clustering to a similarity graph built directly from the solution of its proposed optimization program. However, the code of the algorithm applies a heuristic post-processing step, similar to [65], to the low-rank solution prior to building the similarity graph [40]. Thus, to compare the effectiveness of sparse versus low-rank objective function and to investigate the effect of the post-processing step of LRR, we report the results for both cases of without (LRR) and with (LRR-H) the heuristic post-processing step.[8] For LRSC, we use the method in [41, Lemma 1] with parameter $\tau = 420$ for motion segmentation, and an ALM variant of the method in [41, Section 4.2] with parameters $\alpha = 3\tau = 0.5 * (1.25/\sigma_1(\boldsymbol{Y}))^2$, $\gamma = 0.008$ and $\rho = 1.5$ for face clustering. Finally, as LSA and SCC need to know the number of subspaces a priori and the estimation of the number of subspaces from the eigenspectrum of the graph Laplacian in the noisy setting is often unreliable, in order to have a fair comparison, we provide the number of subspaces as an input to all the algorithms.

Datasets and some statistics. For the motion segmentation problem, we consider the Hopkins 155 dataset [66], which consists of 155 video sequences of 2 or 3 motions corresponding to 2 or 3 low-dimensional subspaces in each video [2], [67]. For the face clustering problem, we consider the Extended Yale B dataset [68], which consists of face images of 38 human subjects, where images of each subject lie in a low-dimensional subspace [3].

Before describing each problem in detail and presenting the experimental results, we present some statistics on the two datasets that help to better understand the challenges of subspace clustering and the performance of different algorithms. First, we compute the smallest principal angle for each pair of subspaces, which in the motion segmentation problem corresponds to a pair of motions in a video and in the face clustering problem corresponds to a pair of subjects. Then, we compute the percentage of the subspace pairs whose smallest principal angle is below a certain value, which ranges from 0 to 90 degrees. Figure 9 (left) shows the corresponding graphs for the two datasets. As

8. The original published code of LRR contains the function "compacc.m" for computing the misclassification rate, which is erroneous. We have used the correct code for computing the misclassification rate and as a result, the reported performance for LRR-H is different from the published results in [38] and [40].

TABLE 1

Clustering error (%) of different algorithms on the Hopkins 155 dataset with the $2F$-dimensional data points.

Algorithms	LSA	SCC	LRR	LRR-H	LRSC	SSC
2 Motions						
Mean	4.23	2.89	4.10	2.13	3.69	**1.52** (2.07)
Median	0.56	**0.00**	0.22	**0.00**	0.29	**0.00** (0.00)
3 Motions						
Mean	7.02	8.25	9.89	**4.03**	7.69	4.40 (5.27)
Median	1.45	**0.24**	6.22	1.43	3.80	0.56 (0.40)
All						
Mean	4.86	4.10	5.41	2.56	4.59	**2.18** (2.79)
Median	0.89	**0.00**	0.53	**0.00**	0.60	**0.00** (0.00)

TABLE 2

Clustering error (%) of different algorithms on the Hopkins 155 dataset with the $4n$-dimensional data points obtained by applying PCA.

Algorithms	LSA	SCC	LRR	LRR-H	LRSC	SSC
2 Motions						
Mean	3.61	3.04	4.83	3.41	3.87	**1.83** (2.14)
Median	0.51	**0.00**	0.26	**0.00**	0.26	**0.00** (0.00)
3 Motions						
Mean	7.65	7.91	9.89	4.86	7.72	**4.40** (5.29)
Median	1.27	1.14	6.22	1.47	3.80	**0.56** (0.40)
All						
Mean	4.52	4.14	5.98	3.74	4.74	**2.41** (2.85)
Median	0.57	**0.00**	0.59	**0.00**	0.58	**0.00** (0.00)

shown, subspaces in both datasets have relatively small principal angles. In the Hopkins-155 dataset, principal angles between subspaces are always smaller than 10 degrees, while in the Extended Yale B dataset, principal angles between subspaces are between 10 and 20 degrees. Second, for each pair of subspaces, we compute the percentage of data points that have one or more of their K-nearest neighbors in the other subspace. Figure 9 (right) shows the average percentages over all possible pairs of subspaces in each dataset. As shown, in the Hopkins-155 dataset for almost all data points, their few nearest neighbors belong to the same subspace. On the other hand, for the Extended Yale B dataset, there is a relatively large number of data points whose nearest neighbors come from the other subspace. This percentage rapidly increases as the number of nearest neighbors increases. As a result, from the two plots in Figure 9, we can conclude that in the Hopkins 155 dataset the challenge is that subspaces have small principal angles, while in the Extended Yale B dataset, beside the principal angles between subspaces being small, the challenge is that data points in a subspace are very close to other subspaces.

7.1 Motion Segmentation

Motion segmentation refers to the problem of segmenting a video sequence of multiple rigidly moving objects into multiple spatiotemporal regions that correspond to different motions in the scene (Fig. 1). This problem is often solved by extracting and tracking a set of N feature points $\{\boldsymbol{x}_{fi} \in \mathbb{R}^2\}_{i=1}^N$ through the frames $f = 1, \ldots, F$ of the video. Each data point $\boldsymbol{y}_i$, which is also called a feature trajectory, corresponds to a $2F$-dimensional vector obtained by stacking the feature points $\boldsymbol{x}_{fi}$ in the video as

$$\boldsymbol{y}_i \triangleq \begin{bmatrix} \boldsymbol{x}_{1i}^\top & \boldsymbol{x}_{2i}^\top & \cdots & \boldsymbol{x}_{Fi}^\top \end{bmatrix}^\top \in \mathbb{R}^{2F}. \tag{34}$$

Motion segmentation refers to the problem of separating these feature trajectories according to their underlying motions. Under the affine projection model, all feature trajectories associated with a single rigid motion lie in an affine subspace of $\mathbb{R}^{2F}$ of dimension at most 3, or equivalently lie in a linear subspace of $\mathbb{R}^{2F}$ of dimension at most 4 [2], [67]. Therefore, feature trajectories of n rigid motions lie in a union of n low-dimensional subspaces of $\mathbb{R}^{2F}$. Hence, motion segmentation reduces to clustering of data points in a union of subspaces.

In this section, we evaluate the performance of the SSC algorithm as well as that of state-of-the-art subspace clustering methods for the problem of motion segmentation. To do so, we consider the Hopkins 155 dataset [66] that consists of 155 video sequences, where 120 of the videos have two motions and 35 of the videos have three motions. On average, in the dataset, each sequence of 2 motions has $N = 266$ feature trajectories and $F = 30$ frames, while each sequence of 3 motions has $N = 398$ feature trajectories and $F = 29$ frames. The left plot of Figure 10 shows the singular values of several motions in the dataset. Note that the first four singular values are nonzero and the rest of the singular values are very close to zero, corroborating the 4-dimensionality of the underlying linear subspace of each motion.[9] In addition, it shows that the feature trajectories of each video can be well modeled as data points that almost perfectly lie in a union of linear subspaces of dimension at most 4.

The results of applying subspace clustering algorithms to the dataset when we use the original $2F$-dimensional feature trajectories and when we project the data into a $4n$-dimensional subspace (n is the number of subspaces) using PCA are shown in Table 1 and Table 2, respectively. From the results, we make the following conclusions:

– In both cases, SSC obtains a small clustering error outperforming the other algorithms. This suggests that the separation of different motion subspaces in terms of their principal angles and the distribution of the feature trajectories in each motion subspace are sufficient for the success of the sparse optimization program, hence clustering. The numbers inside parentheses show the clustering errors of SSC without normalizing the similarity matrix, i.e., without step 2 in Algorithm 1. Notice that, as explained in Remark 1, the normalization step helps to improve the clustering results. However, this improvement is small (about 0.5%), i.e., SSC performs well with or without the post-processing of $\boldsymbol{C}$.

– Without post-processing of its coefficient matrix, LRR has higher errors than other algorithms. On the other hand, post-processing of the low-rank coefficient matrix significantly improves the clustering performance (LRR-H).

– LRSC tries to find a noise-free dictionary for data while finding their low-rank representation. This helps to improve over LRR. Also, note that the errors of LRSC are higher than the reported ones in [41]. This comes from the fact that [41] has used the erroneous compacc.m function from [32] to compute the errors.

– The clustering performances of different algorithms when using the $2F$-dimensional feature trajectories or the $4n$-dimensional PCA projections are close. This comes from the fact that the feature trajectories of n motions in a video almost perfectly lie in a $4n$-dimensional linear subspace of the $2F$-dimensional ambient space. Thus, projection using PCA onto a $4n$-dimensional subspace preserves the structure of the subspaces and the data,

9. If we subtract the mean of the data points in each motion from them, the singular values drop at 3, showing the 3-dimensionality of the affine subspaces.

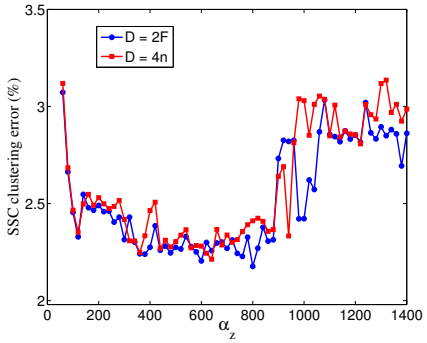


Fig. 11. Clustering error (%) of SSC as a function of α_z in the regularization parameter $\lambda_z = \alpha_z/\mu_z$ for the two cases of clustering of $2F$-dimensional data and $4n$-dimensional data obtained by PCA.

hence, for each algorithm, the clustering error in Table 1 is close to the error in Table 2.

In Figure 11 we show the effect of the regularization parameter $\lambda_z = \alpha_z/\mu_z$ in the clustering performance of SSC over the entire Hopkins 155 dataset. Note that the clustering errors of SSC as a function of α_z follow a similar pattern using both the $2F$-dimensional data and the $4n$-dimensional data. Moreover, in both cases the clustering error is less than 2.5% in both cases for a large range of values of α_z.

Finally, notice that the results of SSC in Tables 1-2 do not coincide with those reported in [35]. This is mainly due to the fact in [35] we used random projections for dimensionality reduction, while here we use PCA or the original $2F$-dimensional data. In addition, in [35] we used a CVX solver to compute a subspace-sparse representation, while here we use an ADMM solver. Also, notice that we have improved the overall clustering error of LSA, for the the case of $4n$-dimensional data, from 4.94%, reported in [66], [35], to 4.52%. This is due to using $K = 8$ nearest neighbors here instead of $K = 5$ in [66].

7.2 Face Clustering

Given face images of multiple subjects, acquired with a fixed pose and varying illumination, we consider the problem of clustering images according to their subjects (Fig. 2). It has been shown that, under the Lambertian assumption, images of a subject with a fixed pose and varying illumination lie close to a linear subspace of dimension 9 [3]. Thus, the collection of face images of multiple subjects lie close to a union of 9-dimensional subspaces.

In this section, we evaluate the clustering performance of SSC as well as the state-of-the-art methods on the Extended Yale B dataset [68]. The dataset consists of 192×168 pixel cropped face images of $n = 38$ individuals, where there are $N_i = 64$ frontal face images for each subject acquired under various lighting conditions. To reduce the computational cost and the memory requirements of all algorithms, we downsample the images to 48×42 pixels and treat each $2,016$-dimensional vectorized image as a data point, hence, $D = 2,016$. The right plot in Figure 10 shows the singular values of data points of several subjects in the dataset. Note that the singular value curve has a knee around 9, corroborating the approximate 9-dimensionality of the face data in each subject. In addition, the singular values gradually decay to zero, showing that the data are corrupted by errors. Thus, the face images of n subjects can be modeled as corrupted data points lying close to a union of 9-dimensional subspaces.

To study the effect of the number of subjects in the clustering performance of different algorithms, we devise the following experimental setting: we divide the 38 subjects into 4 groups, where the first three groups correspond to subjects 1 to 10, 11 to 20, 21 to 30, and the fourth group corresponds to subjects 31 to 38. For each of the first three groups we consider all choices of $n \in \{2, 3, 5, 8, 10\}$ subjects and for the last group we consider all choices of $n \in \{2, 3, 5, 8\}$.[10] Finally, we apply clustering algorithms for each trial, i.e., each set of n subjects.

7.2.1 Applying RPCA separately on each subject

As shown by the SVD plot of the face data in Figure 10 (right), the face images do not perfectly lie in a linear subspace as they are corrupted by errors. In fact, the errors correspond to the cast shadows and specularities in the face images and can be modeled as sparse outlying entries. As a result, it is important for a subspace clustering algorithm to effectively deal with data with sparse corruptions.

In order to validate the fact that corruption of faces is due to sparse outlying errors and show the importance of dealing with corruptions while clustering, we start by the following experiment. We apply the Robust Principal Component Analysis (RPCA) algorithm [32] to remove the sparse outlying entries of the face data in each subject. Note that *in practice*, we do not know the clustering of the data beforehand, hence cannot apply the RPCA to the faces of each subject. However, as we will show, this experiment illustrates some of the challenges of the face clustering and validates several conclusions about the performances of different algorithms.

Table 3 shows the clustering error of different algorithms after applying RPCA to the data points in each subject and removing the sparse outlying entries, i.e., after bringing the data points back to their low-dimensional subspaces. From the results, we make the following conclusions:

– The clustering error of SSC is very close to zero for different number of subjects suggesting that SSC can deal well with face clustering if the face images are corruption free. In other words, while the data in different subspaces are very close to each other, as shown in Figure 9 (right), the performance of the SSC is more dependent on the principal angles between subspaces which, while small, are large enough for the success of SSC.

– The LRR and LRSC algorithms have also low clustering errors (LRSC obtains zero errors) showing the effectiveness of removing sparse outliers in the clustering performance. On the other hand, while LRR-H has a low clustering error for 2, 3, and 5 subjects, it has a relatively large error for 8 and 10 subjects, showing that the post processing step on the obtained low-rank coefficient matrix not always improves the result of LRR.

– For LSA and SCC, the clustering error is relatively large and the error increases as the number of subjects increases. This comes from the fact that, as shown in Figure 9 (right), for face images, the neighborhood of each data point contains points that belong to other subjects and, in addition, the number of neighbors from other subjects increases as we increase the number of subjects.

10. Note that choosing n out of 38 leads to extremely large number of trials. Thus, we have devised the above setting in order to have a repeatable experiment with a reasonably large number of trials for each n.

TABLE 3
Clustering error (%) of different algorithms on the Extended Yale B dataset after applying RPCA separately to the data points in each subject.

Algorithm	LSA	SCC	LRR	LRR-H	LRSC	SSC
2 Subjects						
Mean	6.15	1.29	0.09	0.05	**0.00**	0.06
Median	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
3 Subjects						
Mean	11.67	19.33	0.12	0.10	**0.00**	0.08
Median	2.60	8.59	**0.00**	**0.00**	**0.00**	**0.00**
5 Subjects						
Mean	21.08	47.53	0.16	0.15	**0.00**	0.07
Median	19.21	47.19	**0.00**	**0.00**	**0.00**	**0.00**
8 Subjects						
Mean	30.04	64.20	4.50	11.57	**0.00**	0.06
Median	29.00	63.77	0.20	15.43	**0.00**	**0.00**
10 Subjects						
Mean	35.31	63.80	0.15	13.02	**0.00**	0.89
Median	30.16	64.84	**0.00**	13.13	**0.00**	0.31

TABLE 4
Clustering error (%) of different algorithms on the Extended Yale B dataset after applying RPCA simultaneously to all the data in each trial.

Algorithm	LSA	SCC	LRR	LRR-H	LRSC	SSC
2 Subjects						
Mean	32.53	9.29	7.27	5.72	5.67	**2.09**
Median	47.66	7.03	6.25	3.91	4.69	**0.78**
3 Subjects						
Mean	53.02	32.00	12.29	10.01	8.72	**3.77**
Median	51.04	37.50	11.98	9.38	8.33	**2.60**
5 Subjects						
Mean	58.76	53.05	19.92	15.33	10.99	**6.79**
Median	56.87	51.25	19.38	15.94	10.94	**5.31**
8 Subjects						
Mean	62.32	66.27	31.39	28.67	16.14	**10.28**
Median	62.50	64.84	33.30	31.05	14.65	**9.57**
10 Subjects						
Mean	62.40	63.07	35.89	32.55	21.82	**11.46**
Median	62.50	60.31	34.06	30.00	25.00	**11.09**

7.2.2 Applying RPCA simultaneously on all subjects

In practice, we cannot apply RPCA separately to the data in each subject because the clustering is unknown. In this section, we deal with sparse outlying entries in the data by applying the RPCA algorithm to the collection of all data points for each trial prior to clustering. The results are shown in Table 4 from which we make the following conclusions:

– The clustering error for SSC is low for all different number of subjects. Specifically, SSC obtains 2.09% and 11.46% for clustering of data points in 2 and 10 subjects, respectively.

– Applying RPCA to all data points simultaneously may not be as effective as applying RPCA to data points in each subject separately. This comes from the fact that RPCA tends to bring the data points into a common low-rank subspace which can result in decreasing the principal angles between subspaces and decreasing the distances between data points in different subjects. This can explain the increase in the clustering error of all clustering algorithms with respect to the results in Table 3.

7.2.3 Using original data points

Finally, we apply the clustering algorithms to the original data points without pre-processing the data. The results are shown in Table 5 from which we make the following conclusions:

TABLE 5
Clustering error (%) of different algorithms on the Extended Yale B dataset without pre-processing the data.

Algorithm	LSA	SCC	LRR	LRR-H	LRSC	SSC
2 Subjects						
Mean	32.80	16.62	9.52	2.54	5.32	**1.86**
Median	47.66	7.82	5.47	0.78	4.69	**0.00**
3 Subjects						
Mean	52.29	38.16	19.52	4.21	8.47	**3.10**
Median	50.00	39.06	14.58	2.60	7.81	**1.04**
5 Subjects						
Mean	58.02	58.90	34.16	6.90	12.24	**4.31**
Median	56.87	59.38	35.00	5.63	11.25	**2.50**
8 Subjects						
Mean	59.19	66.11	41.19	14.34	23.72	**5.85**
Median	58.59	64.65	43.75	10.06	28.03	**4.49**
10 Subjects						
Mean	60.42	73.02	38.85	22.92	30.36	**10.94**
Median	57.50	75.78	41.09	23.59	28.75	**5.63**

– The SSC algorithm obtains a low clustering error for all numbers of subjects, obtaining 1.86% and 10.94% clustering error for 2 and 10 subjects, respectively. In fact, the error is smaller than when applying RPCA to all data points. This is due to the fact that SSC directly incorporates the corruption model of the data by sparse outlying entries into the sparse optimization program, giving it the ability to perform clustering on the corrupted data.

– While LRR also has a regularization term to deal with the corrupted data, the clustering error is relatively large especially as the number of subjects increases. This can be due to the fact that there is not a clear relationship between corruption of each data point and the LRR regularization term in general [38]. On the other hand, the post processing step of LRR-H on the low-rank coefficient matrix helps to significantly reduce the clustering error, although it is larger than the SSC error.

– As LRSC tries to recover error-free data points while finding their low-rank representation, it obtains smaller errors than LRR.

– LSA and SCC do not have an explicit way to deal with corrupted data. This together with the fact that the face images of each subject have relatively a large number of neighbors in other subjects, as shown in Figure 9 (right), result in low performances of these algorithms.

7.2.4 Computational time comparison

The average computational time of each algorithm as a function of the number of subjects (or equivalently the number of data points) is shown in Figure 12. Note that the computational time of SCC is drastically higher than other algorithms. This comes from the fact that the complexity of SCC increases exponentially in the dimension of the subspaces, which in this case is $d = 9$. On the other hand, SSC, LRR and LRSC use fast and efficient convex optimization techniques which keeps their computational time lower than other algorithms. The exact computational times are provided in the supplementary materials.

8 Conclusions and Future Work

We studied the problem of clustering a collection of data points that lie in or close to a union of low-dimensional subspaces. We proposed a subspace clustering algorithm based on sparse representation techniques, called SSC, that finds a sparse representation of each point in the dictionary of the other points,

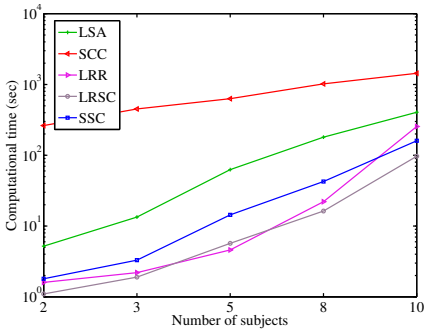


Fig. 12. Average computational time (sec.) of the algorithms on the Extended Yale B dataset as a function of the number of subjects.

builds a similarity graph using the sparse coefficients, and obtains the segmentation of the data using spectral clustering. We showed that, under appropriate conditions on the arrangement of subspaces and the distribution of data, the algorithm succeeds in recovering the desired sparse representations of data points. A key advantage of the algorithm is its ability to directly deal with data nuisances, such as noise, sparse outlying entries, and missing entries as well as the more general class of affine subspaces by incorporating the corresponding models into the sparse optimization program. Experiments on real data such as face images and motions in videos showed the effectiveness of our algorithm and its superiority over the state of the art.

Interesting avenues of research, which we are currently investigating, include theoretical analysis of the subspace-sparse recovery in the presence of noise, sparse outlying entries, and missing entries in the data. As our extensive experiments on synthetic and real data show, the points in each subspace, in general, form a single component of the similarity graph. Theoretical analysis of the connectivity of the similarity graph for points in the same subspace in a probabilistic framework would provide a better understanding for this observation. Finally, making the two steps of solving a sparse optimization program and spectral clustering applicable to very large datasets is an interesting and a practical subject for the future work.

ACKNOWLEDGMENT

The authors would like to thank the financial support of grants NSF-ISS 0447739 and NSF-CSN 0931805.

REFERENCES

[1] R. E. Bellman, *Dynamic programming*. Princeton University Press, 1957.
[2] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography," *International Journal of Computer Vision*, vol. 9, no. 2, pp. 137–154, 1992.
[3] R. Basri and D. Jacobs, "Lambertian reflection and linear subspaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 3, pp. 218–233, 2003.
[4] T. Hastie and P. Simard, "Metrics and models for handwritten character recognition," *Statistical Science*, vol. 13, no. 1, pp. 54–65, 1998.
[5] R. Vidal, "Subspace clustering," *Signal Processing Magazine*, vol. 28, no. 2, pp. 52–68, 2011.
[6] W. Hong, J. Wright, K. Huang, and Y. Ma, "Multi-scale hybrid linear models for lossy image representation," *IEEE Trans. on Image Processing*, vol. 15, no. 12, pp. 3655–3671, 2006.
[7] A. Yang, J. Wright, Y. Ma, and S. Sastry, "Unsupervised segmentation of natural images via lossy data compression," *Computer Vision and Image Understanding*, vol. 110, no. 2, pp. 212–225, 2008.
[8] J. Costeira and T. Kanade, "A multibody factorization method for independently moving objects." *Int. Journal of Computer Vision*, vol. 29, no. 3, 1998.
[9] K. Kanatani, "Motion segmentation by subspace separation and model selection," in *IEEE Int. Conf. on Computer Vision*, vol. 2, 2001, pp. 586–591.
[10] R. Vidal, Y. Ma, and S. Sastry, "Generalized Principal Component Analysis (GPCA)," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 12, pp. 1–15, 2005.
[11] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley-Interscience, October 2004.
[12] P. Tseng, "Nearest q-flat to m points," *Journal of Optimization Theory and Applications*, vol. 105, no. 1, pp. 249–252, 2000.
[13] J. Ho, M. H. Yang, J. Lim, K. Lee, and D. Kriegman, "Clustering appearances of objects under varying illumination conditions." in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2003.
[14] T. Zhang, A. Szlam, and G. Lerman, "Median k-flats for hybrid linear modeling with many outliers," in *Workshop on Subspace Methods*, 2009.
[15] C. W. Gear, "Multibody grouping from motion images," *Int. Journal of Computer Vision*, vol. 29, no. 2, pp. 133–150, 1998.
[16] Y. Ma, A. Yang, H. Derksen, and R. Fossum, "Estimation of subspace arrangements with applications in modeling and segmenting mixed data," *SIAM Review*, 2008.
[17] M. Tipping and C. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural Computation*, vol. 11, no. 2, pp. 443–482, 1999.
[18] Y. Sugaya and K. Kanatani, "Geometric structure of degeneracy for multi-body motion segmentation," in *Workshop on Statistical Methods in Video Processing*, 2004.
[19] A. Gruber and Y. Weiss, "Multibody factorization with uncertainty and missing data using the EM algorithm," in *IEEE Conf. on Computer Vision and Pattern Recognition*, vol. I, 2004, pp. 707–714.
[20] M. A. Fischler and R. C. Bolles, "RANSAC random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 26, pp. 381–395, 1981.
[21] S. Rao, R. Tron, R. Vidal, and Y. Ma, "Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 10, 2010.
[22] J. Yan and M. Pollefeys, "A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate," in *European Conf. on Computer Vision*, 2006, pp. 94–106.
[23] A. Goh and R. Vidal, "Segmenting motions of different types by unsupervised manifold clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
[24] T. Zhang, A. Szlam, Y. Wang, and G. Lerman, "Hybrid linear modeling via local best-fit flats," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1927–1934.
[25] L. Zelnik-Manor and M. Irani, "Degeneracies, dependencies and their implications in multi-body and multi-sequence factorization," in *IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2003, pp. 287–293.
[26] A. Ng, Y. Weiss, and M. Jordan, "On spectral clustering: analysis and an algorithm," in *Neural Information Processing Systems*, 2001, pp. 849–856.
[27] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, 2007.
[28] G. Chen and G. Lerman, "Spectral curvature clustering (SCC)," *International Journal of Computer Vision*, vol. 81, no. 3, pp. 317–330, 2009.
[29] D. L. Donoho, "For most large underdetermined systems of linear equations the minimal ℓ^1-norm solution is also the sparsest solution," *Communications on Pure and Applied Mathematics*, vol. 59, no. 6, pp. 797–829, 2006.
[30] E. Candès and T. Tao, "Decoding by linear programming," *IEEE Trans. on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.
[31] R. Tibshirani, "Regression shrinkage and selection via the LASSO," *Journal of the Royal Statistical Society B*, vol. 58, no. 1, pp. 267–288, 1996.
[32] E. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis," *Journal of the ACM*, vol. 58, 2011.
[33] E. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, pp. 717–772, 2008.
[34] B. Recht, M. Fazel, and P. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Review*, vol. 52, no. 3, pp. 471–501, 2010.
[35] E. Elhamifar and R. Vidal, "Sparse subspace clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
[36] ——, "Clustering disjoint subspaces via sparse representation," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2010.
[37] M. Soltanolkotabi and E. J. Candes, "A geometric analysis of subspace clustering with outliers," *Annals of Statistics*, 2012.
[38] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation," in *International Conference on Machine Learning*, 2010.
[39] G. Liu and S. Yan, "Latent low-rank representation for subspace segmentation and feature extraction," *International Conference on Computer Vision*, 2011.
[40] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[41] P. Favaro, R. Vidal, and A. Ravichandran, "A closed form solution to robust subspace estimation and clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[42] E. Elhamifar and R. Vidal, "Robust classification using structured sparse representation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[43] ——, "Block-sparse recovery via convex optimization," *IEEE Transactions on Signal Processing*, 2012.

[44] F. Parvaresh, H. Vikalo, S. Misra, and B. Hassibi, "Recovering sparse signals using sparse measurement matrices in compressed dna microarrays," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 3, pp. 275–285, Jun. 2008.

[45] M. Stojnic, F. Parvaresh, and B. Hassibi, "On the reconstruction of block-sparse signals with and optimal number of measurements," *IEEE Trans. Signal Processing*, vol. 57, no. 8, pp. 3075–3085, Aug. 2009.

[46] Y. C. Eldar and M. Mishali, "Robust recovery of signals from a structured union of subspaces," *IEEE Trans. Inform. Theory*, vol. 55, no. 11, pp. 5302–5316, 2009.

[47] Y. C. Eldar, P. Kuppinger, and H. Bolcskei, "Compressed sensing of block-sparse signals: Uncertainty relations and efficient recovery," *IEEE Trans. Signal Processing*, vol. 58, no. 6, pp. 3042–3054, June 2010.

[48] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[49] S. J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale l1-regularized least squares," *IEEE Journal on Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 606–617, 2007.

[50] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[51] E. Amaldi and V. Kann, "On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems," *Theoretical Computer Science*, vol. 209, pp. 237–260, 1998.

[52] R. Zass and A. Shashua, "Doubly stochastic normalization for spectral clustering," *Neural Information Processing Systems*, 2006.

[53] T. Brox and J. Malik, "Object segmentation by long term analysis of point trajectories," *European Conference on Computer Vision*, 2010.

[54] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, Feb. 2009.

[55] R. Vidal and R. Hartley, "Motion segmentation with missing data by PowerFactorization and Generalized PCA," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. II, 2004, pp. 310–316.

[56] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization," *PNAS*, vol. 100, no. 5, pp. 2197–2202, 2003.

[57] R. Gribonval and M. Nielsen, "Sparse representations in unions of bases," *IEEE Trans. Information Theory*, vol. 49, no. 12, pp. 3320–3325, Dec. 2003.

[58] E. van den Berg and M. Friedlander, "Theoretical and empirical results for recovery from multiple measurements," *IEEE Trans. Information Theory*, vol. 56, no. 5, pp. 2516–2527, 2010.

[59] D. L. Donoho, "Neighborly polytopes and sparse solution of underdetermined linear equations," *Technical Report, Stanford University*, 2005.

[60] B. Nasihatkon and R. Hartley, "Graph connectivity in sparse subspace clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[61] E. Elhamifar, G. Sapiro, and R. Vidal, "See all by looking at a few: Sparse modeling for finding representative objects," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[62] R. Jenatton, J. Y. Audibert, and F. Bach, "Structured variable selection with sparsity-inducing norms," *Journal of Machine Learning Research*, vol. 12, pp. 2777–2824, 2011.

[63] J. A. Tropp., "Algorithms for simultaneous sparse approximation. part ii: Convex relaxation," *Signal Processing, special issue "Sparse approximations in signal and image processing"*, vol. 86, pp. 589–602, 2006.

[64] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite-element approximations," *Comp. Math. Appl.*, vol. 2, pp. 17–40, 1976.

[65] F. Lauer and C. Schnörr, "Spectral clustering of linear subspaces for motion segmentation," in *IEEE International Conference on Computer Vision*, 2009.

[66] R. Tron and R. Vidal, "A benchmark for the comparison of 3-D motion segmentation algorithms," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[67] T. Boult and L. Brown, "Factorization-based segmentation of motions," in *IEEE Workshop on Motion Understanding*, 1991, pp. 179–186.

[68] K.-C. Lee, J. Ho, and D. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 684–698, 2005.

APPENDIX

PROOF OF PROPOSITION 1

In this section, we prove the result of Proposition 1 in the paper regarding the optimization program

$$\begin{aligned} \min \;\; & \|\boldsymbol{C}\|_1 + \lambda_e \|\boldsymbol{E}\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{Z}\|_F^2 \\ \text{s.t.} \;\; & \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C} + \boldsymbol{E} + \boldsymbol{Z}, \;\; \text{diag}(\boldsymbol{C}) = \boldsymbol{0}. \end{aligned} \tag{1}$$

The result of the proposition suggests to set the regularization parameters as

$$\lambda_e = \alpha_e/\mu_e, \quad \lambda_z = \alpha_z/\mu_z, \tag{2}$$

where $\alpha_e, \alpha_z > 1$ and μ_e and μ_z are defined as

$$\mu_e \triangleq \min_i \max_{j\neq i} \|\boldsymbol{y}_j\|_1, \quad \mu_z \triangleq \min_i \max_{j\neq i} |\boldsymbol{y}_i^\top \boldsymbol{y}_j|. \tag{3}$$

We use the following Lemma in the theoretical proof of the proposition. Proof of this Lemma can be found in [48].

Lemma 1: Consider the optimization program

$$\min \;\; \|\boldsymbol{c}\|_1 + \frac{\lambda}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{c}\|_2^2. \tag{4}$$

For $\lambda < \|\boldsymbol{A}^\top \boldsymbol{y}\|_\infty$, we have $\boldsymbol{c} = \boldsymbol{0}$.

Proposition 1: Consider the optimization program (1). Without the term $\boldsymbol{Z}$, if $\lambda_e \leq 1/\mu_e$, then there exists at least one data point $\boldsymbol{y}_\ell$ for which in the optimal solution we have $(\boldsymbol{c}_\ell, \boldsymbol{e}_\ell) = (\boldsymbol{0}, \boldsymbol{y}_\ell)$. Also, without the term $\boldsymbol{E}$, if $\lambda_z \leq 1/\mu_z$, then there exists at least one data point $\boldsymbol{y}_\ell$ for which $(\boldsymbol{c}_\ell, \boldsymbol{z}_\ell) = (\boldsymbol{0}, \boldsymbol{y}_\ell)$.

Proof: Note that solving the optimization program (1) is equivalent to solving N optimization programs as

$$\begin{aligned} \min \;\; & \|\boldsymbol{c}_i\|_1 + \lambda_e \|\boldsymbol{e}_i\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{z}_i\|_2^2 \\ \text{s.t.} \;\; & \boldsymbol{y}_i = \boldsymbol{Y}\boldsymbol{c}_i + \boldsymbol{e}_i + \boldsymbol{z}_i, \;\; c_{ii} = 0, \end{aligned} \tag{5}$$

where $\boldsymbol{c}_i$, $\boldsymbol{e}_i$, and $\boldsymbol{z}_i$ are the i-th columns of $\boldsymbol{C}$, $\boldsymbol{E}$, and $\boldsymbol{Z}$, repsectively.

(a) Consider the optimization program (5) without the term $\boldsymbol{z}_i$ and denote the objective function value by

$$\text{cost}(\boldsymbol{c}_i, \boldsymbol{e}_i) \triangleq \|\boldsymbol{c}_i\|_1 + \lambda_e \|\boldsymbol{e}_i\|_1. \tag{6}$$

Note that a feasible solution of (5) is given by $(\boldsymbol{0}, \boldsymbol{e}_i)$ for which the value of the objective function is equal to

$$\text{cost}(\boldsymbol{0}, \boldsymbol{e}_i) = \lambda_e \|\boldsymbol{y}_i\|_1. \tag{7}$$

On the other hand, using matrix norm properties, for any feasible solution $(\boldsymbol{c}_i, \boldsymbol{e}_i)$ of (5) we have

$$\|\boldsymbol{y}_i\|_1 = \|\boldsymbol{Y}\boldsymbol{c}_i + \boldsymbol{e}_i\|_1 \leq (\max_{j\neq i} \|\boldsymbol{y}_j\|_1) \, \|\boldsymbol{c}_i\|_1 + \|\boldsymbol{e}_i\|_1, \tag{8}$$

where we used the fact that $c_{ii} = 0$. Multiplying both sides of the above inequality by λ_e we obtain

$$\text{cost}(\boldsymbol{0}, \boldsymbol{y}_i) = \lambda_e \|\boldsymbol{Y}\|_1 \leq (\lambda_e \max_{j\neq i} \|\boldsymbol{y}_j\|_1) \, \|\boldsymbol{c}_i\|_1 + \lambda_e \|\boldsymbol{e}_i\|_1, \tag{9}$$

Note that if $\lambda_e < \frac{1}{\max_{j\neq i} \|\boldsymbol{y}_j\|_1}$, then from the above equation we have

$$\text{cost}(\boldsymbol{0}, \boldsymbol{y}_i) \leq \text{cost}(\boldsymbol{c}_i, \boldsymbol{e}_i). \tag{10}$$

In other words, $(\boldsymbol{c}_i = \boldsymbol{0}, \boldsymbol{e}_i = \boldsymbol{y}_i)$ achieve the minimum cost among all feasible solutions of (5). Hence, if $\lambda_e < \max_i \frac{1}{\max_{j\neq i} \|\boldsymbol{y}_j\|_1}$, then there exists $\ell \in \{1, \cdots, N\}$ such that in the solution of the optimization program (1) we have $(\boldsymbol{c}_\ell, \boldsymbol{e}_\ell) = (\boldsymbol{0}, \boldsymbol{y}_\ell)$.

(b) Consider the optimization program (5) without the term $\boldsymbol{e}_i$, which, using $\boldsymbol{z}_i = \boldsymbol{y}_i - \boldsymbol{Y}\boldsymbol{c}_i$, can be rewritten as

$$\min \;\; \|\boldsymbol{c}_i\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{y}_i - \boldsymbol{Y}\boldsymbol{c}_i\|_2^2 \;\; \text{s.t.} \;\; c_{ii} = 0. \tag{11}$$

From Lemma 1 we have that, for $\lambda_z < \frac{1}{\max_{j\neq i} |\boldsymbol{y}_j{}^\top \boldsymbol{y}_i|}$, the solution of (11) is equal to $\boldsymbol{c}_i = 0$, or equivalently, the solution of (5) is given by $(\boldsymbol{c}_i, \boldsymbol{z}_i) = (\boldsymbol{0}, \boldsymbol{y}_i)$. As a result, if $\lambda_z < \max_i \frac{1}{\max_{j\neq i} |\boldsymbol{y}_j^\top \boldsymbol{y}_i|}$, then there exists $\ell \in \{1, \cdots, N\}$ such that in the solution of the optimization program (1) we have $(\boldsymbol{c}_\ell, \boldsymbol{z}_\ell) = (\boldsymbol{0}, \boldsymbol{y}_\ell)$.

□

PROOF OF THEOREM 1

In this section, we prove Theorem 1 in the paper, where we showed that for data points in a union of independent subspaces, the solution of the ℓ_q-minimization recovers subspace-sparse representations of data points.

Theorem 1: *Consider a collection of data points drawn from n independent subspaces $\{\mathcal{S}_i\}_{i=1}^n$. Let $\boldsymbol{Y}_i$ denote the data points that lie in $\mathcal{S}_i$ and $\boldsymbol{Y}_{-i}$ denote the data points that lie in all subspaces except $\mathcal{S}_i$. Then for every i and every nonzero data point $\boldsymbol{y}$ in $\mathcal{S}_i$, the ℓ_q-minimization program*

$$\begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{c}_-^* \end{bmatrix} = \text{argmin} \left\| \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix} \right\|_q \;\; \text{s.t.} \;\; \boldsymbol{y} = [\boldsymbol{Y}_i \;\; \boldsymbol{Y}_{-i}] \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix}, \tag{12}$$

for $q < \infty$, recovers a subspace-sparse representation, i.e., $\boldsymbol{c}^* \neq \boldsymbol{0}$ and $\boldsymbol{c}_-^* = \boldsymbol{0}$.

Proof: We prove the result using contradiction. Assume $\boldsymbol{c}_-^* \neq 0$. Then we can write

$$\boldsymbol{y} = \boldsymbol{Y}_i \boldsymbol{c}^* + \boldsymbol{Y}_{-i} \boldsymbol{c}_-^*. \tag{13}$$

Since $\boldsymbol{y}$ is a data point in subspace $\mathcal{S}_i$, there exists a $\boldsymbol{c}$ such that $\boldsymbol{y} = \boldsymbol{Y}_i \boldsymbol{c}$. Substituting this into (13) we get

$$\boldsymbol{Y}_i(\boldsymbol{c} - \boldsymbol{c}^*) = \boldsymbol{Y}_{-i} \boldsymbol{c}_-^*. \tag{14}$$

Note that the left hand side of equation (14) corresponds to a point in the subspace $\mathcal{S}_i$ while the right hand side of (14) corresponds to a point in the subspace $\oplus_{j\neq i} \mathcal{S}_j$. By the independence assumption, the two subspaces $\mathcal{S}_i$ and $\oplus_{j\neq i} \mathcal{S}_j$ are also independent hence disjoint and intersect only at the origin. Thus, from (14) we must have $\boldsymbol{Y}_{-i} \boldsymbol{c}_-^* = \boldsymbol{0}$ and from (13) we obtain $\boldsymbol{y} = \boldsymbol{Y}_i \boldsymbol{c}^*$. In other words, $\begin{bmatrix} \boldsymbol{c}^{*\top} & \boldsymbol{0}^\top \end{bmatrix}^\top$ is a feasible solution of the optimization problem (12). Finally, from the assumption of $\boldsymbol{c}_-^* \neq 0$, we have

$$\left\| \begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{0} \end{bmatrix} \right\|_q < \left\| \begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{c}_-^* \end{bmatrix} \right\|_q \tag{15}$$

that contradicts the optimality of $\begin{bmatrix} \boldsymbol{c}^{*\top} & \boldsymbol{c}_-^* \end{bmatrix}^\top$. Thus, we must have $\boldsymbol{c}^* \neq \boldsymbol{0}$ and $\boldsymbol{c}_-^* = \boldsymbol{0}$, obtaining the desired result. □

PROOF OF THEOREM 2

In this section, we prove Theorem 2 in the paper, where we provide a necessary and sufficient condition for subspace-sparse recovery in a union of disjoint subspaces. To do so, we consider a vector $\mathbf{x}$ in the intersection of $\mathcal{S}_i$ with $\oplus_{j\neq i}\mathcal{S}_j$ and let the optimal solution of the ℓ_1-minimization when we restrict the dictionary to the points from $\mathcal{S}_i$ be

$$\boldsymbol{a}_i = \operatorname{argmin} \|\boldsymbol{a}\|_1 \quad \text{s.t.} \quad \mathbf{x} = \boldsymbol{Y}_i\, \boldsymbol{a}. \tag{16}$$

We also let the optimal solution of the ℓ_1 minimization when we restrict the dictionary to the points from all subspaces except $\mathcal{S}_i$ be

$$\boldsymbol{a}_{-i} = \operatorname{argmin} \|\boldsymbol{a}\|_1 \quad \text{s.t.} \quad \mathbf{x} = \boldsymbol{Y}_{-i}\, \boldsymbol{a}. \tag{17}$$

We show that if for every nonzero $\mathbf{x}$ in the intersection of $\mathcal{S}_i$ with $\oplus_{j\neq i}\mathcal{S}_j$, the ℓ_1-norm of the solution of (16) is strictly smaller than the ℓ_1-norm of the solution of (17), i.e.,

$$\forall\, \mathbf{x} \in \mathcal{S}_i \cap (\oplus_{j\neq i}\mathcal{S}_j), \mathbf{x} \neq \mathbf{0} \implies \|\boldsymbol{a}_i\|_1 < \|\boldsymbol{a}_{-i}\|_1, \tag{18}$$

then the SSC algorithm succeeds in recovering subspace-sparse representations of all the data points in $\mathcal{S}_i$.

Theorem 2: *Consider a collection of data points drawn from n disjoint subspaces $\{\mathcal{S}_i\}_{i=1}^n$. Let $\boldsymbol{Y}_i$ denote the data points that lie in $\mathcal{S}_i$ and $\boldsymbol{Y}_{-i}$ denote the data points that lie in all subspaces except $\mathcal{S}_i$. The ℓ_1-minimization program*

$$\begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{c}^*_- \end{bmatrix} = \operatorname{argmin} \left\| \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix} \right\|_1 \quad \text{s.t.} \quad \boldsymbol{y} = [\boldsymbol{Y}_i \;\; \boldsymbol{Y}_{-i}] \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix}, \tag{19}$$

recovers a subspace-sparse representation of every nonzero data point $\boldsymbol{y}$ in $\mathcal{S}_i$, i.e., $\boldsymbol{c}^* \neq \mathbf{0}$ and $\boldsymbol{c}^*_- = \mathbf{0}$, if and only if the condition in (18) *holds.*

Proof: ($\Longleftarrow$) We prove the result using contradiction. Assume $\boldsymbol{c}^*_- \neq \mathbf{0}$ and define

$$\mathbf{x} \triangleq \boldsymbol{y} - \boldsymbol{Y}_i \boldsymbol{c}^* = \boldsymbol{Y}_{-i} \boldsymbol{c}^*_-. \tag{20}$$

Since $\boldsymbol{y}$ lies in $\mathcal{S}_i$ and $\boldsymbol{Y}_i\boldsymbol{c}^*$ is a linear combination of points in $\mathcal{S}_i$, from the first equality in (20) we have that $\mathbf{x}$ is a vector in $\mathcal{S}_i$. Let $\boldsymbol{a}_i$ be the solution of (16) for $\mathbf{x}$. We have

$$\mathbf{x} = \boldsymbol{y} - \boldsymbol{Y}_i \boldsymbol{c}^* = \boldsymbol{Y}_i \boldsymbol{a}_i \;\Rightarrow\; \boldsymbol{y} = \boldsymbol{Y}_i(\boldsymbol{c}^* + \boldsymbol{a}_i). \tag{21}$$

On the other hand, since $\boldsymbol{Y}_{-i}\boldsymbol{c}^*_-$ is a linear combination of points in all subspaces except $\mathcal{S}_i$, from the second equality in (20) we have that $\mathbf{x}$ is a vector in $\oplus_{j\neq i}\mathcal{S}_j$. Let $\boldsymbol{a}_{-i}$ be the solution of (17) for $\mathbf{x}$. We have

$$\mathbf{x} = \boldsymbol{Y}_{-i} \boldsymbol{c}^*_- = \boldsymbol{Y}_{-i} \boldsymbol{a}_{-i} \;\Rightarrow\; \boldsymbol{y} = \boldsymbol{Y}_i \boldsymbol{c}^* + \boldsymbol{Y}_{-i} \boldsymbol{a}_{-i}. \tag{22}$$

Note that the left hand side of (22) together with the fact that $\boldsymbol{a}_{-i}$ is the optimal solution of (17) imply that

$$\|\boldsymbol{a}_{-i}\|_1 \leq \|\boldsymbol{c}^*_-\|_1. \tag{23}$$

From (21) and (22) we have that $\begin{bmatrix} \boldsymbol{c}^* + \boldsymbol{a}_i \\ \mathbf{0} \end{bmatrix}$ and $\begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{a}_{-i} \end{bmatrix}$ are feasible solutions of the original optimization program in (19). Thus, we have

$$\left\| \begin{bmatrix} \boldsymbol{c}^* + \boldsymbol{a}_i \\ \mathbf{0} \end{bmatrix} \right\|_1 \leq \|\boldsymbol{c}^*\|_1 + \|\boldsymbol{a}_i\|_1 < \|\boldsymbol{c}^*\|_1 + \|\boldsymbol{a}_{-i}\|_1 \leq \left\| \begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{c}^*_- \end{bmatrix} \right\|_1, \tag{24}$$

where the first inequality follows from triangle inequality, the second strict inequality follows from the sufficient condition in (18), and the last inequality follows from (23). This contradicts the optimality of $\begin{bmatrix} \boldsymbol{c}^{*\top} & \boldsymbol{c}^{*\top}_- \end{bmatrix}^\top$ for the original optimization program in (19), hence proving the desired result.

($\Longrightarrow$) We prove the result using contradiction. Assume the condition in (18) does not hold, i.e., there exists a nonzero $\mathbf{x}$ in the intersection of $\mathcal{S}_i$ and $\oplus_{j\neq i}\mathcal{S}_j$ for which we have $\|\boldsymbol{a}_{-i}\|_1 \leq \|\boldsymbol{a}_i\|_1$. As a result, for $\boldsymbol{y} = \mathbf{x}$, a solution of the ℓ_1-minimization program (19) corresponds to selecting points from all subspaces except $\mathcal{S}_i$, which contradicts the subspace-sparse recovery assumption. □

PROOF OF THEOREM 3

Theorem 3: *Consider a collection of data points drawn from n disjoint subspaces $\{\mathcal{S}_i\}_{i=1}^n$ of dimensions $\{d_i\}_{i=1}^n$. Let $\mathbb{W}_i$ be the set of all full rank submatrices $\tilde{\boldsymbol{Y}}_i \in \mathbb{R}^{D\times d_i}$ of $\boldsymbol{Y}_i$. If the sufficient condition*

$$\max_{\tilde{\boldsymbol{Y}}_i \in \mathbb{W}_i} \sigma_{d_i}(\tilde{\boldsymbol{Y}}_i) > \sqrt{d_i}\, \|\boldsymbol{Y}_{-i}\|_{1,2} \max_{j\neq i} \cos(\theta_{ij}) \tag{25}$$

is satisfied, then for every nonzero data point $\boldsymbol{y}$ in $\mathcal{S}_i$, the ℓ_1-minimization program in (19) *recovers a subspace-sparse solution, i.e., $\boldsymbol{c}^* \neq \mathbf{0}$ and $\boldsymbol{c}^*_- = \mathbf{0}$.*[1]

Proof: We prove the result in two steps. In step 1, we show that $\|\boldsymbol{a}_i\|_1 \leq \beta_i$. In step 2, we show that $\beta_{-i} \leq \|\boldsymbol{a}_{-i}\|_1$. Then, the sufficient condition $\beta_i < \beta_{-i}$ establishes the result of the theorem, since it implies

$$\|\boldsymbol{a}_i\|_1 \leq \beta_i < \beta_{-i} \leq \|\boldsymbol{a}_{-i}\|_1, \tag{26}$$

i.e., the condition of Theorem 2 holds.

Step 1: Upper bound on the ℓ_1-norm of (16) Let $\mathbb{W}_i$ be the set of all submatrices $\tilde{\boldsymbol{Y}}_i \in \mathbb{R}^{D\times d_i}$ of $\boldsymbol{Y}_i$ that are full column rank. We can write the vector $\mathbf{x} \in \mathcal{S}_i \cap (\oplus_{j\neq i}\mathcal{S}_j)$

$$\mathbf{x} = \tilde{\boldsymbol{Y}}_i \tilde{\boldsymbol{a}} \implies \tilde{\boldsymbol{a}} = (\tilde{\boldsymbol{Y}}_i^\top \tilde{\boldsymbol{Y}}_i)^{-1} \tilde{\boldsymbol{Y}}_i^\top \mathbf{x}. \tag{27}$$

Using vector and matrix norm properties, we have

$$\begin{aligned} \|\tilde{\boldsymbol{a}}\|_1 &\leq \sqrt{d_i}\|\tilde{\boldsymbol{a}}\|_2 = \sqrt{d_i}\, \|(\tilde{\boldsymbol{Y}}_i^\top \tilde{\boldsymbol{Y}}_i)^{-1} \tilde{\boldsymbol{Y}}_i^\top \mathbf{x}\|_2 \\ &\leq \sqrt{d_i}\, \|(\tilde{\boldsymbol{Y}}_i^\top \tilde{\boldsymbol{Y}}_i)^{-1} \tilde{\boldsymbol{Y}}_i^\top\|_{2,2} \|\mathbf{x}\|_2 = \frac{\sqrt{d_i}}{\sigma_{d_i}(\tilde{\boldsymbol{Y}}_i)}\, \|\mathbf{x}\|_2, \end{aligned} \tag{28}$$

where $\sigma_{d_i}(\tilde{\boldsymbol{Y}}_i)$ denotes the d_i-th largest singular value of $\tilde{\boldsymbol{Y}}_i$. Thus, for the solution of the optimization problem in (16), we have

$$\|\boldsymbol{a}_i\|_1 \leq \min_{\tilde{\boldsymbol{Y}}_i \in \mathbb{W}_i} \|\tilde{\boldsymbol{a}}\|_1 \leq \min_{\tilde{\boldsymbol{Y}}_i \in \mathbb{W}_i} \frac{\sqrt{d_i}}{\sigma_{d_i}(\tilde{\boldsymbol{Y}}_i)} \|\mathbf{x}\|_2 \triangleq \beta_i, \tag{29}$$

which established the upper bound on the ℓ_1-norm of the solution of the optimization program in (16).

Step 2: Lower bound on the ℓ_1-norm of (17) For the solution of (17) we have $\mathbf{x} = \boldsymbol{Y}_{-i}\boldsymbol{a}_{-i}$. If we multiply both sides of this equation from left by $\mathbf{x}^\top$, we get

$$\|\mathbf{x}\|_2^2 = \mathbf{x}^\top \mathbf{x} = \mathbf{x}^\top \boldsymbol{Y}_{-i} \boldsymbol{a}_{-i}. \tag{30}$$

Applying the Holder's inequality ($|\boldsymbol{u}^\top \boldsymbol{v}| \leq \|\boldsymbol{u}\|_\infty \|\boldsymbol{v}\|_1$) to the above equation, we obtain

$$\|\mathbf{x}\|_2^2 \leq \|\boldsymbol{Y}_{-i}^\top \mathbf{x}\|_\infty \|\boldsymbol{a}_{-i}\|_1. \tag{31}$$

1. $\|\boldsymbol{Y}_{-i}\|_{1,2}$ is the maximum ℓ_2-norm of the columns of $\boldsymbol{Y}_{-i}$.

By recalling the definition of the smallest principal angle between two subspaces, we can write

$$\|\mathbf{x}\|_2^2 \leq \max_{j\neq i} \cos(\theta_{ij})\, \|\boldsymbol{Y}_{-i}\|_{1,2}\, \|\mathbf{x}\|_2\, \|\boldsymbol{a}_{-i}\|_1, \tag{32}$$

where θ_{ij} is the first principal angle between $\mathcal{S}_i$ and $\mathcal{S}_j$ and $\|\boldsymbol{Y}_{-i}\|_{1,2}$ is the maximum ℓ_2-norm of the columns of $\boldsymbol{Y}_{-i}$, i.e., data points in all subspaces except $\mathcal{S}_i$. We can rewrite (32) as

$$\beta_{-i} \triangleq \frac{\|\mathbf{x}\|_2}{\max_{j\neq i} \cos(\theta_{ij})\, \|\boldsymbol{Y}_{-i}\|_{1,2}} \leq \|\boldsymbol{a}_{-i}\|_1 \tag{33}$$

which establishes the lower bound on the ℓ_1 norm of the solution of the optimization program in (17). □

SOLVING THE SPARSE OPTIMIZATION

Note that the proposed convex programs can be solved using generic convex solvers such as CVX[2]. However, generic solvers typically have high computational costs and do not scale well with the dimension and the number of data points.

In this section, we study efficient implementations of the proposed sparse optimizations using an Alternating Direction Method of Multipliers (ADMM) method. We fist consider the most general optimization program

$$\begin{aligned} &\min_{(\boldsymbol{C},\boldsymbol{E},\boldsymbol{Z})} \ \|\boldsymbol{C}\|_1 + \lambda_e \|\boldsymbol{E}\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{Z}\|_F^2 \\ &\text{s.t.} \quad \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C} + \boldsymbol{E} + \boldsymbol{Z}, \ \ \boldsymbol{C}^\top \mathbf{1} = \mathbf{1}, \quad \operatorname{diag}(\boldsymbol{C}) = \mathbf{0}, \end{aligned} \tag{34}$$

and present an ADMM algorithm to solve it.

First, note that using the equality constraint in (34), we can eliminate $\boldsymbol{Z}$ from the optimization program and equivalently solve

$$\begin{aligned} &\min_{(\boldsymbol{C},\boldsymbol{E})} \ \|\boldsymbol{C}\|_1 + \lambda_e \|\boldsymbol{E}\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{C} - \boldsymbol{E}\|_F^2 \\ &\text{s.t.} \quad \boldsymbol{C}^\top \mathbf{1} = \mathbf{1}, \quad \operatorname{diag}(\boldsymbol{C}) = \mathbf{0}. \end{aligned} \tag{35}$$

The overall procedure of the ADMM algorithm is to introduce appropriate auxiliary variables into the optimization program, augment the constraints into the objective function, and iteratively minimize the Lagrangian with respect to the primal variables and maximize it with respect to the Lagrange multipliers. With an abuse of notation, throughout this section, we denote by $\operatorname{diag}(\boldsymbol{C})$ both a vector whose elements are the diagonal entries of $\boldsymbol{C}$ and a diagonal matrix whose diagonal elements are the diagonal entries of $\boldsymbol{C}$.

To start, we introduce an auxiliary matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ and consider the optimization program

$$\begin{aligned} &\min_{(\boldsymbol{C},\boldsymbol{E},\boldsymbol{A})} \ \|\boldsymbol{C}\|_1 + \lambda_e \|\boldsymbol{E}\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{A} - \boldsymbol{E}\|_F^2 \\ &\text{s.t.} \quad \boldsymbol{A}^\top \mathbf{1} = \mathbf{1}, \quad \boldsymbol{A} = \boldsymbol{C} - \operatorname{diag}(\boldsymbol{C}). \end{aligned} \tag{36}$$

whose solution for $(\boldsymbol{C}, \boldsymbol{E})$ coincides with the solution of (35). As we will see shortly, introducing $\boldsymbol{A}$ helps to obtain efficient updates on the optimization variables. Next, using a parameter $\rho > 0$, we add to the objective function of (36) two penalty terms corresponding to the constraints $\boldsymbol{A}^\top \mathbf{1} = \mathbf{1}$ and $\boldsymbol{A} = \boldsymbol{C} - \operatorname{diag}(\boldsymbol{C})$ and consider the following optimization program

$$\begin{aligned} \min_{(\boldsymbol{C},\boldsymbol{E},\boldsymbol{A})} \ & \|\boldsymbol{C}\|_1 + \lambda_e \|\boldsymbol{E}\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{A} - \boldsymbol{E}\|_F^2 \\ & + \frac{\rho}{2}\|\boldsymbol{A}^\top \mathbf{1} - \mathbf{1}\|_2^2 \quad + \frac{\rho}{2}\|\boldsymbol{A} - (\boldsymbol{C} - \operatorname{diag}(\boldsymbol{C}))\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{A}^\top \mathbf{1} = \mathbf{1}, \quad \boldsymbol{A} = \boldsymbol{C} - \operatorname{diag}(\boldsymbol{C}). \end{aligned} \tag{37}$$

Note that adding the penalty terms to (36) do not change its optimal solution, i.e., both (36) and (37) have the same solutions, since for any feasible solution of (37) that satisfies the constraints, the penalty terms vanish. However, adding the penalty terms makes the objective function strictly convex in terms of the optimization variables $(\boldsymbol{C}, \boldsymbol{E}, \boldsymbol{A})$, which allows using the ADMM approach.

Introducing a vector $\boldsymbol{\delta} \in \mathbb{R}^N$ and and a matrix $\boldsymbol{\Delta} \in \mathbb{R}^{N\times N}$ of Lagrange multipliers for the two equality constraints in (37), we can write the Lagrangian function of (37) as

$$\begin{aligned} \boldsymbol{\mathcal{L}}(\boldsymbol{C}, \boldsymbol{A}, \boldsymbol{E}, \boldsymbol{\delta}, \boldsymbol{\Delta}) = \ & \|\boldsymbol{C}\|_1 + \lambda_e \|\boldsymbol{E}\|_1 + \frac{\lambda_z}{2}\|\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{A} - \boldsymbol{E}\|_F^2 \\ & + \frac{\rho}{2}\|\boldsymbol{A}^\top \mathbf{1} - \mathbf{1}\|_2^2 + \frac{\rho}{2}\|\boldsymbol{A} - (\boldsymbol{C} - \operatorname{diag}(\boldsymbol{C}))\|_F^2 \\ & + \boldsymbol{\delta}^\top (\boldsymbol{A}^\top \mathbf{1} - \mathbf{1}) + \operatorname{tr}(\boldsymbol{\Delta}^\top (\boldsymbol{A} - \boldsymbol{C} + \operatorname{diag}(\boldsymbol{C}))), \end{aligned} \tag{38}$$

where $\operatorname{tr}(\cdot)$ denotes the trace operator of a given matrix. The ADMM approach then consists of an iterative procedure as follows: Denote by $(\boldsymbol{C}^{(k)}, \boldsymbol{E}^{(k)}, \boldsymbol{A}^{(k)})$ the optimization variables at iteration k, and by $(\boldsymbol{\delta}^{(k)}, \boldsymbol{\Delta}^{(k)})$ the Lagrange multipliers at iteration k and

- Obtain $\boldsymbol{A}^{(k+1)}$ by minimizing $\boldsymbol{\mathcal{L}}$ with respect to $\boldsymbol{A}$, while $(\boldsymbol{C}^{(k)}, \boldsymbol{E}^{(k)}, \boldsymbol{\delta}^{(k)}, \boldsymbol{\Delta}^{(k)})$ are fixed. Note that computing the derivative of $\boldsymbol{\mathcal{L}}$ with respect to $\boldsymbol{A}$ and setting it to zero, we obtain

$$\begin{aligned} (\lambda_z \boldsymbol{Y}^\top \boldsymbol{Y} + \rho \boldsymbol{I} + \rho \mathbf{1}\mathbf{1}^\top)\boldsymbol{A}^{(k+1)} = \ & \lambda_z \boldsymbol{Y}^\top (\boldsymbol{Y} - \boldsymbol{E}^{(k)}) \\ & + \rho(\mathbf{1}\mathbf{1}^\top + \boldsymbol{C}^{(k)}) - \mathbf{1}\boldsymbol{\delta}^{(k)\top} - \boldsymbol{\Delta}^{(k)}. \end{aligned} \tag{39}$$

  In other words, $\boldsymbol{A}^{(k+1)}$ is obtained by solving an $N \times N$ system of linear equations. When N is not very large, one can simply matrix inversion to obtain $\boldsymbol{A}^{(k+1)}$ from (39). For large values of N, conjugate gradient methods should be employed to solve for $\boldsymbol{A}^{(k+1)}$.
- Obtain $\boldsymbol{C}^{(k+1)}$ by minimizing $\boldsymbol{\mathcal{L}}$ with respect to $\boldsymbol{C}$, while $(\boldsymbol{A}^{(k)}, \boldsymbol{E}^{(k)}, \boldsymbol{\delta}^{(k)}, \boldsymbol{\Delta}^{(k)})$ are fixed. Note that the update on $\boldsymbol{C}$ also has a closed-form solution given by

$$\boldsymbol{C}^{(k+1)} = \boldsymbol{J} - \operatorname{diag}(\boldsymbol{J}), \tag{40}$$

$$\boldsymbol{J} \triangleq \mathcal{T}_{\frac{1}{\rho}}(\boldsymbol{A}^{(k+1)} + \boldsymbol{\Delta}^{(k)}/\rho), \tag{41}$$

  where $\mathcal{T}_\eta(\cdot)$ is the shrinkage-thresholding operator acting on each element of the given matrix, and is defined as

$$\mathcal{T}_\eta(v) = (|v| - \eta)_+ \operatorname{sgn}(v). \tag{42}$$

  The operator $(\cdot)_+$ returns its argument if it is non-negative and returns zero otherwise.
- Obtain $\boldsymbol{E}^{(k+1)}$ by minimizing $\boldsymbol{\mathcal{L}}$ with respect to $\boldsymbol{E}$, while $(\boldsymbol{C}^{(k+1)}, \boldsymbol{A}^{(k+1)}, \boldsymbol{\delta}^{(k)}, \boldsymbol{\Delta}^{(k)})$ are fixed. The update on $\boldsymbol{E}$ can also be computed in closed-form as

$$\boldsymbol{E}^{(k+1)} = \mathcal{T}_{\frac{\lambda_e}{\lambda_z}}(\boldsymbol{Y}\boldsymbol{A}^{(k+1)} - \boldsymbol{Y}), \tag{43}$$

2. CVX is a Matlab-based software for convex programming and can be downloaded from http://cvxr.com.

Algorithm 1 : Solving (34) **via an ADMM Algorithm**

Initialization: Set maxIter $= 10^4$, $k = 0$, and Terminate $\leftarrow$ False. Initialize $\boldsymbol{C}^{(0)}, \boldsymbol{A}^{(0)}, \boldsymbol{E}^{(0)}, \boldsymbol{\delta}^{(0)}$, and $\boldsymbol{\Delta}^{(0)}$ to zero.

while (Terminate == False) **do**
  update $\boldsymbol{A}^{(k+1)}$ by solving the following system of linear equations

$$(\lambda_z \boldsymbol{Y}^\top \boldsymbol{Y} + \rho \boldsymbol{I} + \rho \mathbf{1}\mathbf{1}^\top)\boldsymbol{A}^{(k+1)} = \lambda_z \boldsymbol{Y}^\top (\boldsymbol{Y} - \boldsymbol{E}^{(k)}) + \rho(\mathbf{1}\mathbf{1}^\top + \boldsymbol{C}^{(k)}) - \mathbf{1}\boldsymbol{\delta}^{(k)\top} - \boldsymbol{\Delta}^{(k)},$$

  update $\boldsymbol{C}^{(k+1)}$ as $\boldsymbol{C}^{(k+1)} = \boldsymbol{J} - \mathrm{diag}(\boldsymbol{J})$, where $\boldsymbol{J} \triangleq \mathcal{T}_{\frac{1}{\rho}}(\boldsymbol{A}^{(k+1)} + \boldsymbol{\Delta}^{(k)}/\rho)$,
  update $\boldsymbol{E}^{(k+1)}$ as $\boldsymbol{E}^{(k+1)} = \mathcal{T}_{\frac{\lambda_e}{\lambda_z}}(\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{A}^{(k+1)})$,
  update $\boldsymbol{\delta}^{(k+1)}$ as $\boldsymbol{\delta}^{(k+1)} = \boldsymbol{\delta}^{(k)} + \rho\,(\boldsymbol{A}^{(k+1)\top}\mathbf{1} - \mathbf{1})$,
  update $\boldsymbol{\Delta}^{(k+1)}$ as $\boldsymbol{\Delta}^{(k+1)} = \boldsymbol{\Delta}^{(k)} + \rho\,(\boldsymbol{A}^{(k+1)} - \boldsymbol{C}^{(k+1)})$,
  $k \leftarrow k + 1$,
  if $(\|\boldsymbol{A}^{(k)\top}\mathbf{1} - \mathbf{1}\|_\infty \leq \epsilon$ and $\|\boldsymbol{A}^{(k)} - \boldsymbol{C}^{(k)}\|_\infty \leq \epsilon$ and $\|\boldsymbol{A}^{(k)} - \boldsymbol{A}^{(k-1)}\|_\infty \leq \epsilon$ and $\|\boldsymbol{E}^{(k)} - \boldsymbol{E}^{(k-1)}\|_\infty \leq \epsilon$ or $(k \geq$ maxIter$)$ **then**
    Terminate $\leftarrow$ True
  end if
end while

Output: Optimal sparse coefficient matrix $\boldsymbol{C}^* = \boldsymbol{C}^{(k)}$.

- Having $(\boldsymbol{C}^{(k+1)}, \boldsymbol{A}^{(k+1)}, \boldsymbol{E}^{(k+1)}$ fixed, perform a gradient ascent update with the step size of ρ on the Lagrange multipliers as

$$\boldsymbol{\delta}^{(k+1)} = \boldsymbol{\delta}^{(k)} + \rho\,(\boldsymbol{A}^{(k+1)\top}\mathbf{1} - \mathbf{1}), \quad (44)$$

$$\boldsymbol{\Delta}^{(k+1)} = \boldsymbol{\Delta}^{(k)} + \rho\,(\boldsymbol{A}^{(k+1)} - \boldsymbol{C}^{(k+1)}). \quad (45)$$

These three steps are repeated until convergence is achieved or the number of iterations exceeds a maximum iteration number. Convergence is achieved when we have $\|\boldsymbol{A}^{(k)\top}\mathbf{1} - \mathbf{1}\|_\infty \leq \epsilon$, $\|\boldsymbol{A}^{(k)} - \boldsymbol{C}^{(k)}\|_\infty \leq \epsilon$, $\|\boldsymbol{A}^{(k)} - \boldsymbol{A}^{(k-1)}\|_\infty \leq \epsilon$ and $\|\boldsymbol{E}^{(k)} - \boldsymbol{E}^{(k-1)}\|_\infty \leq \epsilon$, where ϵ denotes the error tolerance for the primal and dual residuals. In practice, the choice of $\epsilon = 10^{-4}$ works well in real experiments. In summary, Algorithm 1 shows the updates for the ADMM implementation of the optimization program (34).

COMPUTATIONAL TIME COMPARISON

TABLE 1
Average computational time (sec.) of the algorithms on the Extended Yale B dataset as a function of the number of subjects.

	LSA	SCC	LRR	LRSC	SSC
2 Subjects	5.2	262.8	1.6	1.1	1.8
3 Subjects	13.4	451.5	2.2	1.9	3.29
5 Subjects	62.7	630.3	7.6	5.7	11.4
8 Subjects	180.2	1020.5	22.1	16.3	42.6
10 Subjects	405.3	1439.8	255.0	96.9	160.3

Table 1 shows the computational time of different algorithms on the Extended Yale B dataset as a function of the number of subjects. Note that these computational times are based on the codes of the algorithms used by their authors. It is important to mention that LRR and SSC can be implemented using faster optimization solvers. More specifically, LRR can be made faster using LADMAP method proposed in: "Z. Lin and R. Liu and Z. Su, Linearized Alternating Direction Method with Adaptive Penalty for Low-Rank Representation, NIPS 2011." Also, SSC can be made faster using LADM method proposed in "J. Yang and Y. Zhang. Alternating direction algorithms for ℓ_1 problems in compressive sensing. SIAM J. Scientific Computing, 2010."