Subspace Sparse Clustering: Algorithm, Theory, and Applications

Zeyu (Jerry) Wei

Department of Statistics, University of Washington Seattle, WA, 98195, USA

Abstract

This report examines the Sparse Subspace Clustering (SSC) algorithm proposed by Elhamifar and Vidal [2013]. The problem considered is to cluster data points into a set of low-dimensional subspaces. The main idea is to learn a sparse representation for each data point through a sparse optimization program. The sparse representation learnt leads to a graph, on which spectral clustering is applied to extract the connected components as the final clustering. Some practical extensions on the sparse optimization program are proposed to directly deal with data nuances and affine subspaces and to promote connectedness within each subspace. Under conditions on the subspace arrangements and the distribution of the data, the sparse optimization program is guaranteed to recover the desired sparse representation. Experiments on simulated data, motion segmentation task and face clustering problem show empirically the algorithm has the state-of-art performance.

1 Introduction

Many data nowadays are becoming high-dimensional, and many traditional clustering methods suffer from the curse of dimensionality and have unsatisfying performance in high-dimensional settings. However, data in a class or category are often observed to lie in a low-dimensional subspace of the high-dimensional ambient space, and hence the collection of multiple classes or categories can be perceived as a union of low-dimensional subspaces. Such phenomenon, for example, applies to a rigidly moving object in a video, which lies in a linear subspace of dimension at most 4; the phenomenon also applies to face image under varying illumination, which lie close to a linear subspace of dimension 9 according to Lambertian assumption. Taking advantage of the low-dimensional subspace structure,

subspace clustering aims to group the collection of data into the different low-dimensional subspaces. It can be seen as an extension of the standard clustering that not only output cluster memberships but also provide low-dimensional characterization.

1.1 Prior Work on Subspace Clustering

Iterative methods. Iterative approaches alternate between assigning points to subspaces and fitting a subspace to each cluster. Examples of this type of methods are K-subspaces [Tseng, 2000], [Ho et al., 2003] and median K-flats [Zhang et al., 2009]. The main shortcomings of iterative approaches are that the number and dimensions of the subspaces are required to be given a prior and that the algorithms are sensitive to initialization.

Algebraic approaches. *Factorization-based algebraic approaches* [Costeira and Kanade, 1998], [Gear, 1998] start by factorizing the data matrix to construct a similarity matrix and then find the initial segmentation by thresholding the entries in the similarity matrix. This class of methods are sensitive to noise and outliers. *Algebraic-geometric approach*, examplified by the Generalized Principal Component Analysis (GPCA) [Vidal et al., 2012], [Ma et al., 2008],fits the data with a polynomial and use the gradients at each point, which are the normal vectors to the subspaces, to cluster the points. This approach can deal with subspaces of different dimensions, but it is sensitive to noise and outliers,and the computation complexity increases exponentially in terms of the number and dimensions of subspaces.

Statistical methods. Statistical elements are also used by some subspace clustering methods. *Iterative statistical methods* makes parametric assumptions on data distribution and then iterate between clustering and subspace estimation by Expectation Maximization (EM). Examples of such method are Mixtures of Probabilistic PCA (MPPCA) [Archambeau et al., 2008] and Multi-Stage Learning (MSL) [Gruber and Weiss, 2004]. These methods need to specify the number and dimensions of the subspaces and are sensitive to initialization. Random Sample Consensus (RANSAC) [Fischler and Bolles, 1981], which can be categorized as *robust statistical approach*, uses random sets of d points to construct subspaces of dimension d until getting a subspace that have large enough number of inliers. Such inliers are taken as a cluster and the process is repeated for the remaining data. RANSAC is robust to noise and outliers, but all the subspaces need to have the same known dimension,

while the complexity increases exponentially in d. Agglomerative Lossy Compression (ALC) [Rao et al., 2010], as an example of *Information-theoretic statistical approaches*, finds the segmentation of the data that minimizes the coding length needed to fit a mixture of degenerate Gaussian to the points, up to a given distortion. One drawback is that the number of subspaces is dependent on the choice of distortion parameter.

Spectral clustering-based methods This class of methods first use local or global information to construct a similarity graph and then apply spectral clustering to extract group labels. *Local spectral clustering-based approaches* such as Local Subspace Affinity (LSA) [Yan and Pollefeys, 2006], Locally Linear Manifold Clustering (LLMC) [Goh and Vidal, 2007], and Spectral Local Best-fit Flats (SLBF) [Zhang et al., 2012], [Zelnik-Manor and Irani, 2003] use local information to measure similarity between data points. The performance of this type of methods depends on appropriate choice of local neighborhood. *Global spectral clustering-based approaches* use global information to build similarity graph. Spectral Curvature Clustering (SCC) [Chen and Lerman, 2009] utilizes multi-way similarities that capture the curvature of a collection of points within an affine subspace. SCC is robust to noise but assumes all the subspaces to be of the same dimension and needs the number and the dimension of the subspaces to be specified a prior. Low-Rank Recovery (LRR) [Liu et al., 2013],[Liu and Yan, 2011] and Low-Rank Subspace Clustering (LRSC) [Favaro et al., 2011], [Vidal and Favaro, 2013] pose global optimization algorithms to find a low-rank representation of the data as linear combination of other data points. The low-rank representation is then used to build a similarity graph on which spectral clustering is applied. These methods can deal with outliers and noise and do not need dimensions of the subspaces to be specified a prior. The Sparse Subspace Clustering (SSC) method discussed in this report is also a *global spectral clustering-based approach* and is similar to LRR and LRSC except that SSC aims to find a sparse representation instead of a low-rank representation.

1.2 Main Contribution

In this report, we study the Sparse Subspace Clustering method which aims to cluster a collection of data points lying in a union of low-dimensional subspaces. The basis of the approach is the *self-expressiveness property* that each data point can be efficiently repre-

sented as a linear or affine combination of other points in the data set. More specific, to utilize low-dimensional structure for subspace clustering, we aim to find the *subspace-sparse representation* that has each data point represented by *a few* other points *from the same subspace*. A global sparse optimization program is proposed to find such representation, and the solution is used in spectral clustering framework. For this method, the bases for the subspaces need not be specified a prior and the number of points in a subspace is arbitrary, and hence SSC can be seen an extension of block-sparse recovery.

Since sparse optimization with l_0 penalty is generally NP-hard, we consider the l_1 relaxation, which can be solved convex programming tools efficiently [Boyd et al., 2010] and does not require initialization. Theoretical analysis proves that, under suitable conditions on the subspaces and the distribution of the data, the proposed optimization program outputs the desired subspace-sparse representation. With some practical extensions, the algorithm can directly deal with noises, sparse outlying entries, and affine subspaces. Simulation study and real data experiments on motion segmentation and face clustering show the state-of-art performance of the SSC algorithm.

2 Methods

Let $\{S_l\}_{l=1}^n$ be an arrangement of n linear subspaces of dimensions $\{d_l\}_{l=1}^n$ lying in $\mathbb{R}^D$. For this section, we describe the method with observations that are noiseless. Let $\{\boldsymbol{y}_i\}_{i=1}^N$ be the collection of N data points that are free of noises and lie in the union of the n subspaces. Let N_l be the number of samples points in subspace S_l and we can denote the data matrix with all the data points as

$$\boldsymbol{Y} \equiv [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N] = [\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n]\boldsymbol{\Gamma}$$

where $\boldsymbol{Y}_l \in \mathbb{R}^{D \times N_l}$ is a matrix of all the points in S_l with $N_l > d_l$ and $\boldsymbol{\Gamma}$ is an unknown permutation matrix. The goal here is to find the number of subspaces, the corresponding dimensions, a basis for each subspace, and the segmentation of the data $\boldsymbol{Y}$.

To solve the subspace clustering problem, the Sparse Subspace Clustering algorithm is proposed, which consists mainly of two steps. In the first step, we solve a global sparse optimization program to represent each data points by a few other points in the same subspace.

Such sparse representation encodes the membership information of each point with respect to the underlying subspaces. In the second step, the learnt representation is used for spectral clustering to segment the data.

2.1 Sparse Optimization Program

The basic assumption that motivates the SSC algorithm is the so called *self-expressiveness property* of the data:

Definition 2.1. *We say a data has self-expressiveness property if each data point in a union of subspaces can be efficiently reconstructed by a combination of other points in the dataset.*

That is, each data point $\boldsymbol{y}_i \in \cup_{l=1}^{n} \mathcal{S}_l$ can have the representation

$$\boldsymbol{y}_i = \boldsymbol{Y}\boldsymbol{c}_i, \quad c_{ii} = 0 \tag{2.1}$$

where $\boldsymbol{c}_i \equiv (c_{i1}, c_{i2}, \ldots, c_{iN})^T$ is a vector of coefficients. The constraint $c_{ii} = 0$ excludes the trivial representation of $\boldsymbol{y}_i$ by itself. In other words, the collection of data points $\boldsymbol{Y}$ is presumed to be a self-expressive dictionary. Since it is assumed that $N_l > d_l$, the representation in (2.1) is not unique, and an arbitrary such representation does not provide much information for subspace clustering. The key idea for the proposed algorithm is to take a step further to find the *subspace-sparse representation*:

Definition 2.2. *A subspace-sparse representation of a data point is a linear combination of* ***a few*** *other points in* ***the same subspace****.*

Specifically, for $\boldsymbol{y}_i \in \mathcal{S}_l$ we want $\boldsymbol{c}_i$ in (2.1) to be sparse and the nonzero entries correspond to data points from $\mathcal{S}_l$. Such $\boldsymbol{c}_i$ naturally encodes the information of the underlying subspace $\boldsymbol{y}_i$ belongs to. The search for such subspace-sparse representation motivates the sparse optimization problem: for $i = 1, \ldots, N$

$$\min ||\boldsymbol{c}_i||_q \quad s.t. \quad \boldsymbol{y}_i = \boldsymbol{Y}\boldsymbol{c}_i, c_{ii} = 0 \tag{2.2}$$

where $||.||_q$ stands for the l_q norm and promotes sparsity in $\boldsymbol{c}_i$. The constraint ensures the self-expressiveness property. Ideally, we want l_0 norm to exactly control the nonzero

entries in $\boldsymbol{c}_i$ to find the sparsest representation. However, l_0 norm problem is NP-hard, and hence, for computational efficiency, l_1 norm as the tightest convex relaxation is used and the problem becomes, for $i = 1, \ldots, N$

$$\min ||\boldsymbol{c}_i||_1 \quad s.t. \quad \boldsymbol{y}_i = \boldsymbol{Y}\boldsymbol{c}_i, c_{ii} = 0 \tag{2.3}$$

In matrix form we have

$$\min ||\boldsymbol{C}||_1 \quad s.t. \quad \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C}, diag(\boldsymbol{C}) = 0 \tag{2.4}$$

where $\boldsymbol{C} \equiv [\boldsymbol{c}_1], \ldots, \boldsymbol{c}_N] \in \mathbb{R}^{N \times N}$ is the matrix whose i-th column corresponds to the representation coefficient for $\boldsymbol{y}_i$. In the subspace-sparse recovery theory section theoretical analysis are provided to show that under appropriate conditions, the convex optimization program in (2.3) is guaranteed to recover a subspace-sparse representation for each data point.

2.2 Spectral Clustering with Sparse Coefficients

The sparse optimization program in (2.3) has the sparse coefficient vector $\boldsymbol{c}_i$ for each data point $\boldsymbol{y}_i$ as the output, which ideally only has nonzero entries for points from the same subspace. To get the final clustering result, such coefficients are used to construct a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$, where the N data points compose the vertices $\mathcal{V}$, $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ denotes the set of edges between nodes, and $\boldsymbol{W}$ denotes the weights for the edges. Given the sparse coefficients and to make the weight matrix symmetric, we set

$$\boldsymbol{W} = |\boldsymbol{C}| + |\boldsymbol{C}|^T, \boldsymbol{W}_{ij} = |c_{ij}| + |c_{ji}| \tag{2.5}$$

With subspace-sparse representation, the constructed graph $\mathcal{G}$ has different subspaces as different connected components. Spectral clustering [Ng and Jordan, 2001] applied to the constructed graph $\mathcal{G}$ helps to extract connected components of a graph and gives the final segmentation of the data. That is, Kmeans algorithm is applied to the normalized rows of a matrix whose columns are the n bottom eigenvectors of the symmetric normalized graph Laplacian. Note that to deal with the influence of the different norms of the data points on the clustering procedure, we can normalize the sparse coefficients as $\boldsymbol{c}_i / ||\boldsymbol{c}_1||_\infty$. This

helps ensure that the largest edge weights for all the graph nodes are of the same scale, and consequently spectral clustering is not biased by the norm of the sparse coefficients.

The SSC algorithm is summarized in Algorithm 1 below:

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 $\{\mathcal{S}_l\}_{l=1}^n$.

1 Solve the sparse optimization program (2.3)

2 Normalize the columns of the resulting coefficients matrix $\boldsymbol{C}$ 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 $\boldsymbol{W} = |\boldsymbol{C}| + |\boldsymbol{C}|^T$

4 Apply spectral clustering to the similarity graph.

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

---

3 Subspace-Sparse Recovery Theory

The SSC algorithm depends heavily on the construction of a subspace-sparse representation. In this section, we study the conditions under which the sparse optimization program in (2.3) recovers a subspace-sparse representation. Note that we still assume all the data points to be noiseless. The main condition is on the arrangement of the subspaces that we want the subspaces to be "well separated". More specifically, we study two particular types of subspace arrangements: *independent subspaces* and *disjoint subspaces*, where the second type is a generalization of the first type.

Definition 3.1. 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.

An example is illustrated in Figure 1 (left) where $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ are all 1-dimensional subspaces that span the $\mathbb{R}^3$ spaces.

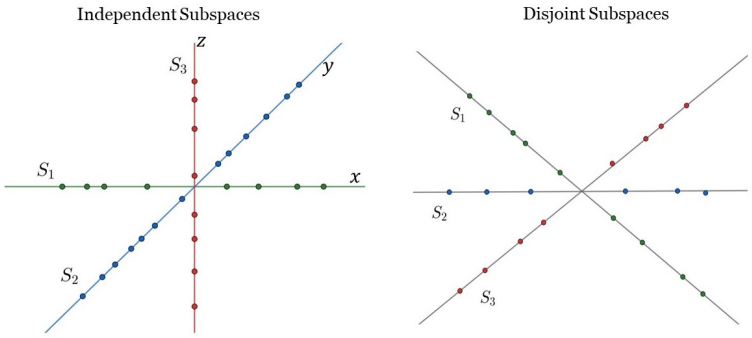

Figure 1: Types of subspaces. Left: three 1-dimensional subspaces are independent. Right: three 1-dimensional subspaces are disjoint.

Definition 3.2. 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 $dim(\mathcal{S}_i \oplus \mathcal{S}_j) = dim(\mathcal{S}_i) + dim(\mathcal{S}_j)$, where $\oplus$ denotes the direct sum operator.

We note that independent subspaces are disjoint while the converse does not necessarily hold. The three 1-dimensional subspaces illustrated in Figure 1 (right) are disjoint since each pair intersects at the origin. However, they only span $\mathbb{R}^2$ space and hence are not independent. Note that the concept of disjointedness essentials excludes the case of non-trivial intersection between subspaces. in the case of disjoint subspaces, we need a notion to measure "the separateness" of pairs of subspace. Such notion is the smallest principal angle:

Definition 3.3. The smallest principal angle between two subspaces $\mathcal{S}_i$ and $\mathcal{S}_j$, denoted by θ_{ij} , is defined as

$$cos(\theta_{ij}) \equiv \max_{\boldsymbol{v}_i \in \mathcal{S}_i, \boldsymbol{v}_j \in \mathcal{S}_j} \frac{\boldsymbol{v}_i^T \boldsymbol{v}_j}{||\boldsymbol{v}_i||_2 ||\boldsymbol{v}_j||_2} \tag{3.1}$$

Two disjoint subspaces intersect at the origin and hence we always have $cos(\theta_{ij}) \in [0, 1)$ for disjoint subspaces $\mathcal{S}_i$ and $\mathcal{S}_j$.

3.1 Independent Subspace Model

We first analyze the l_1 optimization program in (2.3) and more generally the l_q optimization program in (2.2) for $q < \infty$ under the condition of independent subspaces. We prove that subspace-sparse representation is guaranteed in this setting.

Theorem 3.4. *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 $rank(\boldsymbol{Y}_i) = d_i$, 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*

$$\begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{c}_-^* \end{bmatrix} = argmin \left\| \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix} \right\|_q \quad s.t. \quad \boldsymbol{y} = [\boldsymbol{Y}_i \quad \boldsymbol{Y}_{-i}] \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix} \tag{3.2}$$

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

The problem in (3.2) is a rewrite of the basic sparse optimization program in (2.3). Essentially, given independent subspaces, no other condition is needed to guarantee subspace-sparse recovery. However, this happens because the independent subspaces model is already restrictive enough. We then consider the more general disjoint subspaces model.

3.2 Disjoint Subspace Model

Here we show, in the setting of disjoint subspaces, under appropriate conditions on the relative configuration of the subspaces and the distribution of the data in each subspace, the l_1 optimization program in (2.3) recovers subspace-sparse representations. First we define some notions for the points in the intersection of subspaces. Let $\boldsymbol{x}$ be a nonzero vector in the intersection of $\mathcal{S}_i$ and $\oplus_{j \neq i} \mathcal{S}_j$. Let

$$\boldsymbol{a}_i \equiv argmin ||\boldsymbol{a}||_1 \quad s.t. \quad \boldsymbol{x} = \boldsymbol{Y}_i \boldsymbol{a} \tag{3.3}$$

$$\boldsymbol{a}_{-i} \equiv argmin ||\boldsymbol{a}||_1 \quad s.t. \quad \boldsymbol{x} = \boldsymbol{Y}_{-i} \boldsymbol{a} \tag{3.4}$$

Conceptually, $\boldsymbol{a}_i$ is the representation coefficient with respect to a single subspace, and $\boldsymbol{a}_{-i}$ is the representation coefficient with respect to multiple subspaces. We have the result:

Theorem 3.5. *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 $rank(\boldsymbol{Y}_i) = d_i$, and let $\boldsymbol{Y}_{-i}$ denote data points in all subspaces except $\mathcal{S}_i$. The l_1-minimization*

$$\begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{c}_-^* \end{bmatrix} = argmin \left\| \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix} \right\|_1 \quad s.t. \quad \boldsymbol{y} = [\boldsymbol{Y}_i \quad \boldsymbol{Y}_{-i}] \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix} \tag{3.5}$$

recovers a subspace-sparse representation, i.e., $\boldsymbol{c}^* \neq \boldsymbol{0}$ *and* $\boldsymbol{c}^*_- = \boldsymbol{0}$ *if and only if*

$$\forall \boldsymbol{x} \in \mathcal{S}_i \cap (\oplus_{j \neq i} \mathcal{S}_j), \boldsymbol{x} \neq 0 \implies ||\boldsymbol{a}_i||_1 < ||\boldsymbol{a}_{-i}||_1 \tag{3.6}$$

Compared to theorem 3.4 for independent subspaces, we need the additional condition in (3.6) for disjoint subspaces, which says that, for any point in the intersection $\mathcal{S}_i \cap (\oplus_{j \neq i} \mathcal{S}_j)$ that is not the origin, the sparse representation by the single subspace $\mathcal{S}_i$ is always better. This if and only if condition guarantees the subspace-sparse recovery by l_1 optimization program, but does not provide a direct characterization about the arrangement of the subspaces and the distribution of the data. We then provide a sufficient condition for (3.6) to hold and hence also guarantees subspace-sparse recovery. Let $\|\boldsymbol{X}\|_{1,2}$ denotes the maximum l_2 norm of the columns of the matrix $\boldsymbol{X}$ and let $\sigma_i(\boldsymbol{X})$ denotes the i-th largest singular value of $\boldsymbol{X}$, we have the following theorem:

Theorem 3.6. *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* $rank(\boldsymbol{Y}_i) = d_i$*. If the condition*

$$\max_{\tilde{\boldsymbol{Y}}_i \in \mathbb{W}_i} \sigma_{d_i}(\tilde{\boldsymbol{Y}}_i) > \sqrt{d_i} \|\boldsymbol{Y}_{-i}\|_{1,2} \max_{j \neq i} cos(\theta_{ij}) \tag{3.7}$$

holds, then for every nonzero $\boldsymbol{y} \in \mathcal{S}_i$*, the* l_1*-minimization recovers a subspace-sparse solution.*

To make the connection to Theorem 3.5, let $\boldsymbol{x} \in \mathcal{S}_i \cap (\oplus_{j \neq i} \mathcal{S}_j)$ and define $\beta_i \equiv min_{\tilde{\boldsymbol{Y}}_i \in \mathbb{W}_i} \frac{\sqrt{d_i}}{\sigma_i(\tilde{\boldsymbol{Y}}_i)}$, $\beta_{-i} \equiv \frac{||\boldsymbol{x}||_2}{max_{j \neq i} cos(\theta_{ij}) ||\boldsymbol{Y}_{-i}||_{1,2}}$. Here we have β_i dependent on the singular values of data points in $\mathcal{S}_i$, and β_{-i} dependent on the principal angle between $\mathcal{S}_i$ and other subspaces. Condition (3.7) gives $||\boldsymbol{a}_i||_1 \leq \beta_i < \beta_{-i} \leq ||\boldsymbol{a}_{-1}||_1$. In appendix it is proved that

$$||\boldsymbol{a}_i||_1 \leq \beta_i, \quad \beta_{-i} \leq ||\boldsymbol{a}_{-1}||_1$$

and hence we have the subspace-sparse recovery from Theorem 3.5. When the data points are normalized to have unit Euclidean norms, which does not affect the data segmentation, the sufficient condition can be simplified as

$$\max_{\tilde{\boldsymbol{Y}}_i \in \mathbb{W}_i} \sigma_{d_i}(\tilde{\boldsymbol{Y}}_i) > \sqrt{d_i} \max_{j \neq i} cos(\theta_{ij}) \tag{3.8}$$

Loosely speaking, the theoretical analysis tells us that the SSC algorithm will be successful if the subspaces are "well separated" in the sense of having independent arrangement or having disjoint arrangement with pairwise principal angles sufficiently large.

4 Practical Extensions

So far, the SSC algorithm is proposed and analyzed in noiseless setting with linear subspaces. However, in reality, data nuisances such as noise, outliers, and missing entries are inevitable.We propose additional measures to directly deal with such data nuisances in this section. Moreover, we consider the more general affine subspaces which have linear subspaces as a special case. What is more, we also introduce new regularization terms to promote connectedness of points in the same subspace.

4.1 Noise and Sparse Outlying Entries

We first consider data with noises and sparse outlying entries. Noises and outliers can arise from all sorts of measurement errors during the data collection procedure. Let

$$\boldsymbol{y}_i = \boldsymbol{y}_i^0 + \boldsymbol{e}_i^0 + \boldsymbol{z}_i^0 \tag{4.1}$$

where $\boldsymbol{y}_i^0$ is the error-free part of observation and lies perfect on the underlying subspace, $\boldsymbol{e}_i^0 \in \mathbb{R}^D$ has sparse outlying entries with $||\boldsymbol{e}_i^0||_0 \leq k$ for some integer k, and $\boldsymbol{z}_i^0 \in \mathbb{R}^D$ is the noise component with bounded norm that $||\boldsymbol{z}_i^0||_2 \leq \zeta$ for some $\zeta > 0$. The noiseless components of the data points still have self-expressiveness property, and by algebraic manipulation we can represent $\boldsymbol{y}_i$ by other corrupted observations that

$$\boldsymbol{y}_i = \sum_{j \neq i} c_{ij} \boldsymbol{y}_j + \boldsymbol{e}_i + \boldsymbol{z}_i \tag{4.2}$$

$$\boldsymbol{e}_i = \boldsymbol{e}_i^0 - \sum_{j \neq i} c_{ij} \boldsymbol{e}_j^0 \tag{4.3}$$

$$\boldsymbol{z}_i = \boldsymbol{z}_i^0 - \sum_{j \neq i} c_{ij} \boldsymbol{z}_j^0 \tag{4.4}$$

As linear combinations of noises, $\boldsymbol{z}_i$ is again a noise vector. And given the coefficient vector $\boldsymbol{c}_i$ has only few entries nonzero, $\boldsymbol{e}_i$ again is a vector of sparse outlying entries. Combining $\boldsymbol{e}_i, \boldsymbol{z}_i$ as columns into matrices $\boldsymbol{E}$ and $\boldsymbol{Z}$, we want $\boldsymbol{E}$ to be sparse and want $\boldsymbol{Z}$ to have small entries. Therefore, a convex optimization program can be formulated as

$$\begin{aligned} &\min ||\boldsymbol{C}||_1 + \lambda_e ||\boldsymbol{E}||_1 + \frac{\lambda_z}{2} ||\boldsymbol{Z}||_F^2 \\ &s.t. \;\; \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C} + \boldsymbol{E} + \boldsymbol{Z}, \;\; diag(\boldsymbol{C}) = 0 \end{aligned} \tag{4.5}$$

where $\lambda_e, \lambda_z > 0$ are tuning parameters balancing the influence of the regularization terms in the objective function. This optimization problem is convex with respect to $(\boldsymbol{C}, \boldsymbol{E}, \boldsymbol{Z})$ and can be solved by the Alternating Direction Method of Multipliers (ADMM) [Boyd et al., 2010]. The detailed procedure for solving this convex problem is provided in Appendix.

With the introduced tuning parameters λ_e and λ_z, we suggest setting $\lambda_w = \alpha_e/\mu_e$, $\lambda_z = \alpha_z/\mu_z$ where $\alpha_e, \alpha_z > 1$ and

$$\mu_e \equiv \min_i \max_{j \neq i} ||\boldsymbol{y}_i||_1, \quad \mu_z \equiv \min_i \max_{j \neq i} |\boldsymbol{y}_i^T \boldsymbol{y}_j|$$

The reason for such parameter choice is stated in the following proposition:

Proposition 4.1. *Consider the optimization program with noise and sparse outlying entries in (4.5). Without the noise term $\boldsymbol{Z}$, if $\lambda_e \leq 1/\mu_e$, then there exists at least one data point $\boldsymbol{y}_l$ for which in the optimal solution we have $(\boldsymbol{c}_l, \boldsymbol{e}_l) = (\boldsymbol{0}, \boldsymbol{y}_l)$. That is, the data point is taken entirely as outlying entries and is not represented as combination of other data points. Similarly, without the sparse outlying term $\boldsymbol{E}$, if $\lambda_z \leq \mu_z$, then there exists at least one data point $\boldsymbol{y}_l$ for which $(\boldsymbol{c}_l, \boldsymbol{z}_l) = (\boldsymbol{0}, \boldsymbol{y}_l)$.*

Therefore, to ensure meaningful representation for every data point, it is suggested to have $\lambda_e \geq \alpha_e/\mu_e, \lambda_z \geq \alpha_z/\mu_z$ with $\alpha_e, \alpha_z > 1$.

4.2 Missing Entries

Here we address missing entries in the data for clustering purpose. When only a small fraction of the entries of each data point is missing, those missing entries can be taken as sparse outlying entries, and hence clustering of the incomplete data can be obtained through solving the program in (4.5). This approach is direct, but disregards the fact that we know the locations of the missing entries in the data. Also, if the columns with missing entries are small relative to the ambient space dimension D, another approach is to discard those columns and proceed with only complete entries.

4.3 Affine Space

In some real practices, data may lie in affine subspaces instead of linear subspaces. A d_l dimensional affine subspace can be considered as a subset of a d_l+1 dimensional linear subspace using homogeneous coordinates that includes $\mathcal{S}_l$ as the origin. Therefore, a straightforward approach is to ignore the affine subspace structure and proceed with the clustering for linear subspaces. However, this approach possibly increases the dimension of the intersection of two subspaces, which consequently can make distinctive subspaces indistinguishable. For example, take the three lines in Figure 1 (right) as affine subspaces in xy-plane, and they would lie in the same xyz-plane and become indistinguishable.

We then propose a method to directly deal with affine subspaces within the sparse optimization framework. Data point $\boldsymbol{y}_i$ in an affine subspace $\mathcal{S}_l$ with dimension d_l can be written as an affine combination of $d_l + 1$ other points from $\mathcal{S}_l$. That is, by the self-expressiveness property, we have the representation that

$$\boldsymbol{y}_i = \boldsymbol{Y}\boldsymbol{c}_i, c_{ii} = 0, \mathbf{1}^T\boldsymbol{c}_i = 1 \tag{4.6}$$

where $\boldsymbol{c}_i$ has $d_l + 1$ nonzero entries corresponding to points also in $\mathcal{S}_l$. To find such sparse representation for points in affine subspaces we can solve the optimization program:

$$\begin{aligned} &\min ||\boldsymbol{C}||_1 + \lambda_e||\boldsymbol{E}||_1 + \frac{\lambda_z}{2}||\boldsymbol{Z}||_F^2 \\ &s.t. \;\; \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C} + \boldsymbol{E} + \boldsymbol{Z}, \;\; diag(\boldsymbol{C}) = 0, \;\; \mathbf{1}^T\boldsymbol{C} = \mathbf{1}^T \end{aligned} \tag{4.7}$$

Compared to the program in (4.5), new equality constraint is introduced for affine subspace structure. Since linear subspaces are also affine subspaces, the program in (4.7) is more general than program in (4.5).

4.4 Graph Connectivity

Previously, we focused on obtaining subspace-sparse representation, which ensures that points from different subspaces will be disconnected in the constructed graph. This is in the spirit of Hartigan Consistency for clustering problems. However, for the clustering to be accurate, we also need to ensure the connectedness of points in the same subspace in the constructed graph. Unfortunately, for subspaces of dimensions greater than or equal

to 4, under odd distribution of data, it is possible that points in the same subspace form multiple components in the constructed graph [Nasihatkon and Hartley, 2011]. To promote connectedness, we introduce a regularization term in the sparse optimization program that leads points in the same subspace to be all presented by a few common points. Let

$$\|\boldsymbol{C}\|_{r,0} \equiv \sum_{i=1}^{N} I(\|\boldsymbol{c}^i\|_2 > 0) \tag{4.8}$$

where $\boldsymbol{c}^i$ denotes the i-th row of the matrix $\boldsymbol{C}$. The regularization term shows the number of data points that are used to represent other points. Penalization on this term encourages the representations to use a few common data points and hence improves connectivity. Again the use of indicator function gives a NP-hard problem, and hence we consider the convex relaxation that

$$\|\boldsymbol{C}\|_{r,1} \equiv \sum_{i=1}^{N} \|\boldsymbol{c}^i\|_2 \tag{4.9}$$

Thus, to promote the connectivity of data points from the same subspace, the optimization program to consider is

$$\min ||\boldsymbol{C}||_1 + \lambda_r \|\boldsymbol{C}\|_{r,1} \quad s.t. \quad \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C}, diag(\boldsymbol{C}) = 0 \tag{4.10}$$

where $\lambda_r > 0$ sets the trade-off between the sparsity of the solution and the connectivity of the graph.

5 Experiments

In this section we first use simulation study to test the performance of the SSC algorithm on subspaces that are not disjoint and illustrate that the angle between subspaces in such setting greatly affect the accuracy of the SSC algorithm. We then apply SSC algorithm to real data on motion segmentation task and face clustering task to demonstrate that it has state-of-art performance in real practice.

5.1 Simulation Experiment

In section 3 we only have subspace-sparse recovery guarantee for independent or disjoint arrangement of the subspaces, and Theorem 3.6 indicates the importance of the principal

angle between the subspaces. Here we use synthetic data to test the performance of SSC algorithm when the subspaces are not disjoint and see how the angles between subspaces affect the algorithm. We look at noiseless data of ambient dimension $D = 50$ from $n = 3$ subspaces, $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$, each has dimension $d = 4$. We first randomly generate a 4 dimensional space, and the other two subspaces are obtained by rotating the initial space by θ and 2θ in the first two dimensions. That is, let U_1 a 50×4 dimensional orthonormal matrix as basis for $\mathcal{S}_1$, let U_2, U_3 denote the orthonormal matrix representing $\mathcal{S}_2$ and $\mathcal{S}_3$, and let T be the rotation matrix, then we have

$$T = \begin{bmatrix} cos(\theta) & sin(\theta) & & & \\ -sin(\theta) & cos(\theta) & & & \\ & & 1 & & \\ & & & 1 & \end{bmatrix}, \quad U_2 = TU_1, \quad U_3 = TU_2$$

For this report, θ takes 10 equally spaced values from $\pi/90$ to $\pi/2$. With the generated subspaces, N points are randomly sampled from each subspace, where N takes 10 equally spaced integers from 10 to 172. Note that the simulation setting here is different from that in [Elhamifar and Vidal, 2013], where disjoint subspaces are constructed.

With the generated data, the basic form of SSC algorithm (2.4) is applied and two types of errors are measured. Denote the data points as $\{\boldsymbol{y}_i\}_{i=1}^{3N}$ and denote the sparse representation of $\boldsymbol{y}_i$ by $\boldsymbol{c}_i = [\boldsymbol{c}_{i1}^T \ \ \boldsymbol{c}_{i2}^T \ \ \boldsymbol{c}_{i3}^T]^T$ with $\boldsymbol{c}_{ik}$ corresponds to points in $\mathcal{S}_k$. Also let $k(i) \in \{1, 2, 3\}$ be the true membership of point $\boldsymbol{y}_i$. The *subspace-sparse recovery error* is defined as

$$\textbf{recovery error} = 1 - \frac{1}{3N} \sum_{i=1}^{3N} \left(\frac{||\boldsymbol{c}_{ik(i)}||_1}{||\boldsymbol{c}_i||_1} \right) \tag{5.1}$$

where in the summation is the fraction of l_1-norm from points in the same subspace. Also, the clustering result given by SSC algorithm is measured by *subspace clustering error*

$$\textbf{clustering error} = \frac{\#\text{of misclassified points}}{\text{total \# of points}} \tag{5.2}$$

For each combination (θ, N), the data simulation and clustering process is repeated 100 times, and the averages of the errors in (5.1) and (5.2) are reported in Figure 2.

When θ is large and N is large we have both small subspace-sparse recovery error and small clustering error. However, when θ is small or N is small, both kinds of errors are

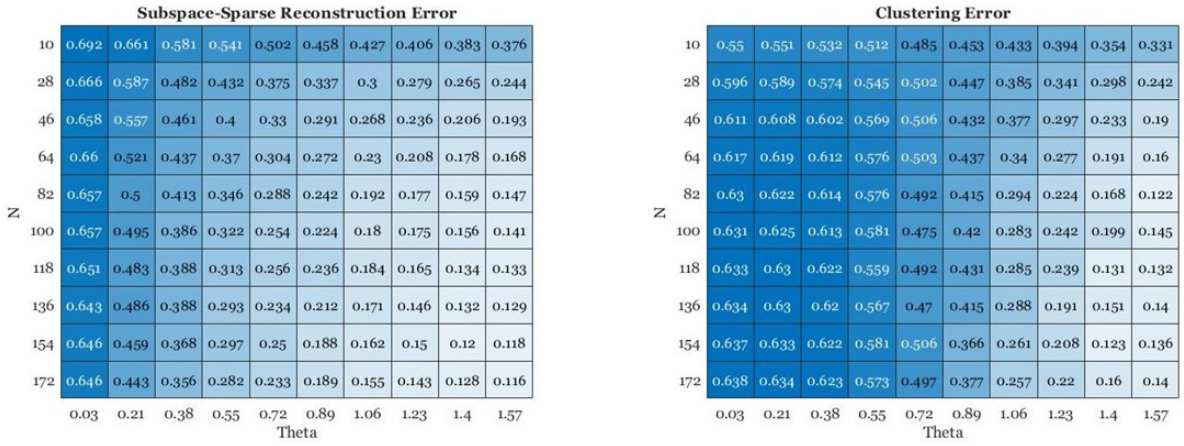

Figure 2: Subspace-sparse recovery error (left) and clustering error (right) for simulated data with different angle between subspaces and sample size.

large. This verifies that the success of SSC algorithm depends on the "separateness" of the underlying subspaces. However, note that for small θ, the clustering error actually increases as the sample size N increases, and the reconstruction error is not decreasing with increasing N. This is because, when the subspaces are close, a point is likely to be represented by points from other subspaces, and the larger the number of points, the more the undesirable connections between points that are in different subspaces, which leads to worse clustering results. These phenomena are similar to that observed in [Elhamifar and Vidal, 2013] despite that we let the intersections between subspaces to be non-trivial. Points in the intersection are challenging for all clustering methods, but with this simulation study we illustrated that SSC has good performance when sample size is sufficiently large and the angles between subspaces are large.

5.2 Real Data Experiments

In this section we apply SSC algorithm to two real-world problems: motion segmentation for videos and clustering human face images. We compare the performance of SSC with other subspace clustering algorithms: LSA, RANSAC, LRSC, and K-Subspace.

The SSC algorithm is implemented using ADMM. For motion segmentation tasks, the most general version of the SSC (4.7) is implemented with $\lambda_e = 0$ and $\lambda_z = \alpha_z/\mu_z$, where

$\alpha_z = 800$ when comparing with other methods. We also include the clustering result with differing α_z in Appendix. For face clustering, we use the noisy version of SSC (4.5) that does not have affine constraint with $\lambda_e = \alpha_e/\mu_e$ and $\lambda_z = 0$, where $\alpha_e = 20$ when comparing with other methods. We also include the clustering result with differing α_e in Appendix.

For we implemented LSA with $k = 8, d = 4$ for motion segmentation and $k = 7, d = 5$ for face clustering, where k is the number of nearest neighbors to look at and d is the dimension to fit local subspaces. For RANSAC, when applied to face clustering we set threshold to be 0.00002 and $d = 9$, where d is dimension of the subspaces to find. For LRSC, we set the parameter $\tau = 420, \alpha = 3000, \gamma = 5$ for motion segmentation and $\tau = 0.045, \alpha = 0.045, \gamma = 0.01$ for face clustering. For K-Subspace, we set the maximum number of the iteration to be 128 with random orthogonal initial subspaces and set the dimension to be 4 for motion segmentation and 9 for face clustering.

5.2.1 Motion Segmentation

In this section we evaluate the performance of the SSC algorithm on motion segmentation task. Generally, data presented for motion segmentation has videos that have frames $f : 1, \ldots, F$, and a set of N feature points (crosses in Figure 3) $\{\boldsymbol{x}_{fi} \in \mathbb{R}^2\}_{i=1}^N$ are tracked across the frames. For analysis, each feature trajectory $\boldsymbol{y}_i$ is taken as a data point, where $\boldsymbol{y}_i$ is obtained by stacking the feature points x_{fi} in the video as

$$\boldsymbol{y}_i \equiv [\boldsymbol{x}_{1i}^T, \boldsymbol{x}_{2i}^T, \ldots, \boldsymbol{x}_{Fi}^T]^T \in \mathbb{R}^{2F}$$

Figure 3: Motion segmentation example from Hopkins 155 dataset where two rigidly moving objects are tracked in multiple frames of a video.

The dataset considered is the Hopkins 155 dataset [Tron and Vidal, 2007], which consists of 120 two-motion videos and 35 three-motion videos. On average, each sequence of 2

motions has $N = 266$ feature trajectories and $F = 30$ frames, while videos of 3 motions have $N = 398$ feature trajectories and $F = 29$ frames. For experiments, we first apply the subspace clustering algorithms onto the $2F$-dimensional feature trajectories. For another set of experiments we use PCA to project the data into $4n$-dimensional subspace, where $n = 2, 3$ is the number of motions (subspaces), and then apply the subspace clustering algorithms. The results are shown in Table 1. The performance of SSC presented is similar to that in [Elhamifar and Vidal, 2013] and achieves state-of-art performance. Noticeably, LRSC, which is similar to SSC but focus on low-rank representation and directly restrain the coefficient matrix to be symmetric, also achieve great performance and in practice has much faster computation. LSA also have good performance with dimension reduction as the preprocessing step. However, applying PCA does not improve the performance of SSC. That is, projecting the data points onto low-dimensional space does not help finding subspace-sparse representations and the following spectral clustering.

Table 1: Clustering error (%) of different algorithms on the Hopkins 155 dataset

Algorithm	SSC	LSA	LRSC	K-Subspace
2 Motions, without PCA				
Mean	**2.23**	20.88	3.98	28.42
Median	**0**	14.72	**0**	33.72
3 Motions, without PCA				
Mean	**5.78**	21.09	7.96	32.27
Median	**0.95**	22.81	3.40	29.62
2 Motions, with PCA				
Mean	**2.32**	3.01	3.89	19.78
Median	**0**	0.25	**0**	1.62
3 Motions, with PCA				
Mean	5.78	**5.19**	7.88	21.34
Median	**0.95**	1.11	3.39	3.19

5.2.2 Face Clustering

In this section we apply SSC algorithm to face clustering problem. The data used is the Extended Yale B dataset [Lee et al., 2005], which consists of 192×168 pixel cropped face images for $n = 38$ individuals. For each individual, $N_i = 64$ frontal face pictures are taken under fixed pose but varying illumination. To reduce computation, all the images are down-

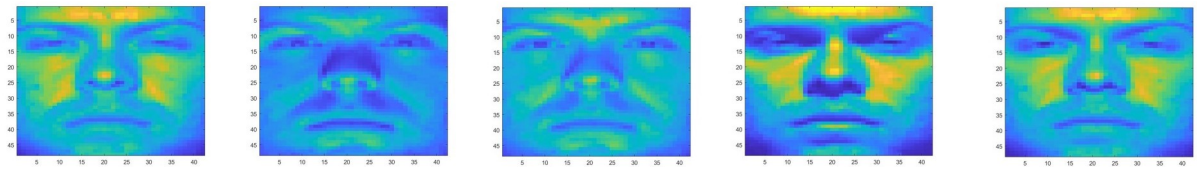

Figure 4: Example of a face under different illumination from Extended Yale B dataset .

sampled to 48×42 pixels, and each vectorized image is treated as a data point. Therefore, we have $D = 48 \times 42 = 2016$ dimensional data. To see how the number of subspaces affect the clustering performance, the following experimental setting is implemented: the 38 individuals are divided into 3 groups of 10 and 1 group of 8, and for each group we consider all the combinations of $n \in \{2, 3, 5, 8, 10\}$ subjects (for the last group only up to combination of 8 subjects). Each combination of n subjects lead to a collection of data with n subspaces, and the subspace clustering algorithms are applied to such constructed datasets. The results of those trails are summarized in Table 2. We note that SSC achieves state-of-art performance for each experimental setting, and the results we have here are similar to that presented in [Elhamifar and Vidal, 2013]. Also, as expected, the clustering error is increasing with more number of subspace. Notably, although SSC is faster than iterative type of methods (K-Subspace and RANSAC), it is slower than the other spectral clustering-based methods (LSA and LRSC) considered here.

6 Discussion

In this report we proposed the SSC algorithm, analyzed theoretically the subspace-sparse recovery conditions, presented some practical extensions of the algorithm, and illustrated its state-of-art performance with data experiments. The SSC algorithm has the great advantage to directly incorporate noise, sparse outlying entries, and affine subspace constraints into the sparse optimization program. Also SSC works for subspaces of different dimensions and hence have very general applications.

However, there are several points that can be of the interest of further research. First, there is no theoretical guarantee for subspace-sparse recovery in the case of corrupted data. Such theoretical guarantee is only for the basic form of SSC in (2.3) with noiseless data in

Table 2: Clustering error (%) and average running time (sec.) of different algorithms on the Extended Yale B dataset

Algorithm	SSC	LSA	RANSAC	LRSC	KSubspace
2 Subjects					
Mean	**1.87**	33.47	37.45	11.41	45.41
Median	**0**	46.09	39.06	11.25	46.88
Time	57.5	13.7	4240.4	2.0	149.2
3 Subjects					
Mean	**3.30**	53.03	49.39	13.97	59.18
Median	**1.04**	51.06	50.52	13.87	59.90
Time	81.1	19.5	6815.0	3.1	536.0
5 Subjects					
Mean	**4.32**	58.82	62.10	21.58	67.59
Median	**2.50**	57.19	63.12	21.56	67.19
Time	135.8	29.4	12721.3	11.2	1115.9
8 Subjects					
Mean	**5.89**	57.41	787	34.73	72.00
Median	**4.59**	57.81	5415	34.37	71.58
Time	216.0	65.0	15901.7	19.5	2030.1
10 Subjects					
Mean	**7.40**	56.04	71.4	51.06	72.03
Median	**5.63**	60.47	72.5	50.78	72.34
Time	326.0	95.3	16393.7	59	3248.0

linear subspaces, but in practice we use the general form in (4.6) to deal with corrupted data in affine subspaces. Theoretical analysis is needed to justify such empirical applications and may provide more guide on parameter selection. Secondly, it is of interest to also have theoretical analysis on the connectivity of the resulted graph in a probabilistic framework. Thirdly, although the sparse optimization program can apply when the number of subspaces is unknown, the spectral clustering step in the SSC algorithm does take the number of clusters as an input when using Kmeans. Further research can generalize the SSC algorithm to settings when the number of clusters is unknown. In such situation, the proposed extension for improving graph connectivity can play a significant role. Moreover, in case of contaminated data, SSC uses corrupted dictionary for calculation, which may implicitly communicate errors in the clustering procedure. It may be helpful to come up with new methods that simultaneously construct a clean dictionary and cluster the data with clean dictionary. Finally, making the algorithm applicable to very large dataset can be of interest.

Acknowledgment

I would love to thank Marina Meila and Vincent Roulet for the help on this report.

References

Cédric Archambeau, Nicolas Delannay, and Michel Verleysen. Mixtures of robust probabilistic principal component analyzers. *Neurocomputing*, 71(7-9):1274–1282, 2008. ISSN 09252312. doi: 10.1016/j.neucom.2007.11.029.

S Boyd, N Parikh, E Chu, J Eckstein, Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends R in Machine Learning*, 3(1):1–122, 2010. doi: 10.1561/2200000016.

Guangliang Chen and Gilad Lerman. Spectral Curvature Clustering (SCC). *International Journal of Computer Vision*, 81(3):317–330, 2009. ISSN 09205691. doi: 10.1007/s11263-008-0178-9.

João Paulo Costeira and Takeo Kanade. A Multibody Factorization Method for Independently Moving Objects. *International Journal of Computer Vision*, 29(3):159–179, 1998. ISSN 09205691. doi: 10.1023/A:1008000628999.

Ehsan Elhamifar and Rene Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11): 2765–2781, 2013. ISSN 01628828. doi: 10.1109/TPAMI.2013.57.

Paolo Favaro, René Vidal, and Avinash Ravichandran. A closed form solution to robust subspace estimation and clustering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (June 2014):1801–1807, 2011. ISSN 10636919. doi: 10.1109/CVPR.2011.5995365.

Martin a Fischler and Robert C Bolles. Paradigm for Model. *Communications of the ACM*, 24(6):381–395, 1981. doi: 10.1145/358669.358692.

C. W. Gear. Multibody Grouping from Motion Images. *International Journal of Computer Vision*, 29(2):133–150, 1998. ISSN 09205691. doi: 10.1023/A:1008026310903.

Alvina Goh and René Vidal. Segmenting motions of different types by unsupervised manifold clustering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007. ISSN 10636919. doi: 10.1109/CVPR.2007.383235.

Amit Gruber and Yair Weiss. Multibody factorization with uncertainty and missing data using the EM algorithm. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:0–7, 2004. ISSN 10636919. doi: 10.1109/cvpr.2004.1315101.

Jeffrey Ho, Ming Hsuan Yang, Jongwoo Lim, Kuang Chih Lee, and David Kriegman. Clustering appearances of objects under varying illumination conditions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1, 2003. ISSN 10636919. doi: 10.1109/cvpr.2003.1211332.

Seung-Jean Kim, K Koh, M Lustig, Stephen Boyd, and Dimitry Gorinevsky. An Interior-Point Method for Large-Scale 1-Regularized Least Squares. *IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING*, 1(4), 2007. doi: 10.1109/JSTSP.2007.910971.

Kuang Chih Lee, Jeffrey Ho, and David J. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):684–698, may 2005. ISSN 01628828. doi: 10.1109/TPAMI.2005.92.

Guangcan Liu and Shuicheng Yan. Latent low-rank representation for subspace segmentation and feature extraction. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1615–1622, 2011. doi: 10.1109/ICCV.2011.6126422.

Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):171–184, 2013. ISSN 01628828. doi: 10.1109/TPAMI.2012.88.

Yi Ma, Allen Y. Yang, Harm Derksen, and Robert M. Fossum. Estimation of subspace arrangements with applications in modeling and segmenting mixed data. *SIAM Review*, 50(3):413–458, 2008. ISSN 00361445. doi: 10.1137/060655523.

Behrooz Nasihatkon and Richard Hartley. Graph Connectivity In Sparse Subspace Clustering. 2011.

Andrew Y Ng and Michael I Jordan. On Spectral Clustering: Analysis and an algorithm. 2001.

Shankar Rao, Roberto Tron, Rene Vidal, and Yi Ma. Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1832–1845, 2010. ISSN 01628828. doi: 10.1109/TPAMI.2009.191.

Roberto Tron and René Vidal. A benchmark for the comparison of 3-D motion segmentation algorithms. 2007. ISSN 10636919. doi: 10.1109/CVPR.2007.382974.

P Tseng. Nearest q-Flat to m Points 1,2. *Journal of Optimization Theory and Applications*, 105(1):249–252, 2000.

René Vidal and Paolo Favaro. Low rank subspace clustering (LRSC). 2013. doi: 10.1016/j.patrec.2013.08.006. URL `http://dx.doi.org/10.1016/j.patrec.2013.08.006`.

Rene Vidal, Yi Ma, and Shankar Sastry. Generalized Principal Component Analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1945–1959, feb 2012. URL `http://arxiv.org/abs/1202.4002`.

J Yan and M Pollefeys. A General Framework for Motion Segmentation : Degenerate and Non-degenerate. *Computer Vision – ECCV 2006*, 3954(Chapter 8):94–106, 2006. ISSN 978-3-540-33838-3. doi: $10.1007/11744085_8$.

Lihi Zelnik-Manor and Michal Irani. Degeneracies, dependencies and their implications in multi-body and multi-sequence factorizations. *Proceedings of the IEEE Computer Society*

Conference on Computer Vision and Pattern Recognition, 2, 2003. ISSN 10636919. doi: 10.1109/cvpr.2003.1211482.

Teng Zhang, Arthur Szlam, and Gilad Lerman. Median K-flats for hybrid linear modeling with many outliers. *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops 2009*, pages 234–241, 2009. doi: 10.1109/ICCVW.2009.5457695.

Teng Zhang, Arthur Szlam, Yi Wang, and Gilad Lerman. Hybrid linear modeling via local best-fit flats. *International Journal of Computer Vision*, 100(3):217–240, 2012. ISSN 09205691. doi: 10.1007/s11263-012-0535-6.

7 Appendix

7.1 Empirical Parameter Choice

Here we present how the mean clustering error of SSC algorithm changes with different parameter setting. In Figure 5, we vary α_z from 20 to 1000 while all other implementations are the same as in section 5 ($\lambda_z = \alpha_z/\mu_z, \lambda_e = 0$). The performance is in general good with $\alpha_z \geq 80$. The graph also illustrates that clustering 3 motions is harder than clustering 2 motions.

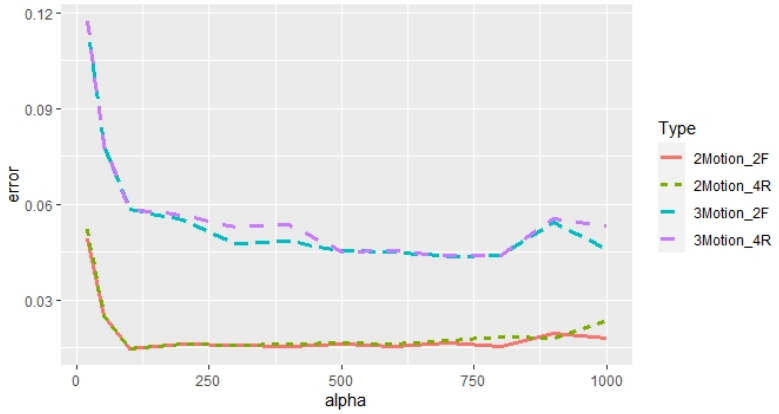

Figure 5: Motion segmentation error v.s. α_z

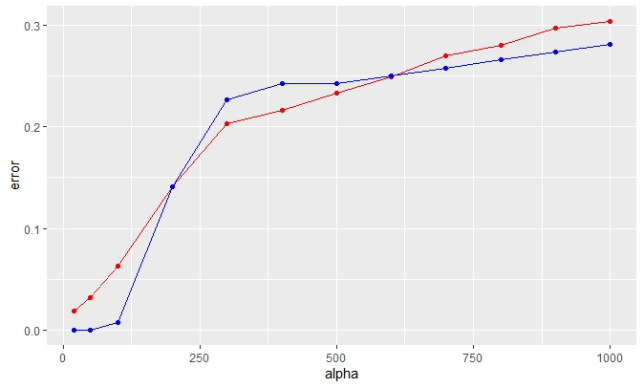

Figure 6: Face clustering error v.s. α_e

In Figure 6 we show the mean and median clustering error changes with α_e on combinations of 2 subjects in Extended Yale B dataset. The implementation is the same as in section 5. We note that generally small α_e is preferred.

7.2 Proof of Theorem 3.4

Theorem. *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 $rank(\boldsymbol{Y}_i) = d_i$, 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*

$$\begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{c}_-^* \end{bmatrix} = argmin \left\| \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix} \right\|_q \quad s.t. \quad \boldsymbol{y} = [\boldsymbol{Y}_i \quad \boldsymbol{Y}_{-i}] \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix} \tag{7.1}$$

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

Proof. We prove by contradiction. Assume in the optimal solution $\boldsymbol{c}_-^* \neq \boldsymbol{0}$, then we can write

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

Since $\boldsymbol{y} \in \mathcal{S}_i$ and $N_i > d_i$, there is a $\boldsymbol{c}$ such that $\boldsymbol{y} = \boldsymbol{Y}_i \boldsymbol{c}$. Combining with above we have

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

Note that $\boldsymbol{Y}_i(\boldsymbol{c}-\boldsymbol{c}^*) \in \mathcal{S}_i$ while $\boldsymbol{Y}_{-i}\boldsymbol{c}^*_- \in \oplus_{j\neq i}\mathcal{S}_j$, where by independent subspaces we have $\mathcal{S}_i$ and $\oplus_{j\neq i}\mathcal{S}_j$ only intersects at the origin. Therefore, we must have $\boldsymbol{Y}_{-i}\boldsymbol{c}^*_- = 0$, and $[\boldsymbol{c}^{*T} \;\; \boldsymbol{0}^T]^T$ is a feasible solution. Consequently,

$$||[\boldsymbol{c}^{*T} \;\; \boldsymbol{0}^T]^T||_q < ||[\boldsymbol{c}^{*T} \;\; \boldsymbol{c}^{*T}_-]^T||_q$$

contradicts the optimalility of $[\boldsymbol{c}^{*T} \;\; \boldsymbol{c}^{*T}_-]^T$. □

7.3 Proof of Theorem 3.5

Theorem. *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 $rank(\boldsymbol{Y}_i) = d_i$, and let $\boldsymbol{Y}_{-i}$ denote data points in all subspaces except $\mathcal{S}_i$. The l_1 minimization*

$$\begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{c}^*_- \end{bmatrix} = argmin \left\| \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix} \right\|_1 \;\; s.t. \;\; \boldsymbol{y} = [\boldsymbol{Y}_i \;\; \boldsymbol{Y}_{-i}] \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c}_- \end{bmatrix} \tag{7.2}$$

recovers a subspace-sparse representation, i.e., $\boldsymbol{c^*} \neq \boldsymbol{0}$ and $\boldsymbol{c}^*_- = \boldsymbol{0}$ if and only if

$$\forall \boldsymbol{x} \in \mathcal{S}_i \cap (\oplus_{j\neq i}\mathcal{S}_j), \boldsymbol{x} \neq 0 \implies ||\boldsymbol{a}_i||_1 < ||\boldsymbol{a}_{-i}||_1 \tag{7.3}$$

Proof. Assume that the condition in (7.3) does not hold that $\exists \boldsymbol{x} \in\in \mathcal{S}_i \cap (\oplus_{j\neq i}\mathcal{S}_j), \boldsymbol{x} \neq 0$ that $||\boldsymbol{a}_i||_1 \geq ||\boldsymbol{a}_{-i}||_1$. Consequently, solving the l_1-optimization in (7.2) gives a representation by points from all other subspaces except except $\mathcal{S}_i$, which contradicts the subspace-sparse recovery.

For the other direction, assume in the optimal solution $\boldsymbol{c}^*_- \neq \boldsymbol{0}$ and define

$$\boldsymbol{x} \equiv \boldsymbol{y} - \boldsymbol{Y}_i\boldsymbol{c}^* = \boldsymbol{Y}_{-i}\boldsymbol{c}^*_-$$

Note that $\boldsymbol{y} - \boldsymbol{Y}_i\boldsymbol{c}^* \in \mathcal{S}_i$ and $\boldsymbol{Y}_{-i}\boldsymbol{c}^*_- \in \oplus_{j\neq i}\mathcal{S}_j$ and hence $\boldsymbol{x} \in \mathcal{S}_i \cap (\oplus_{j\neq i}\mathcal{S}_j)$. Let

$$\boldsymbol{a}_i = argmin||\boldsymbol{a}||_1 \;\; s.t. \;\; \boldsymbol{x} = \boldsymbol{Y}_i\boldsymbol{a}$$

$$\boldsymbol{a}_{-i} = argmin||\boldsymbol{a}||_1 \;\; s.t. \;\; \boldsymbol{x} = \boldsymbol{Y}_{-i}\boldsymbol{a}$$

Combining with above we have

$$\boldsymbol{Y}_i\boldsymbol{a}_i = \boldsymbol{x} = \boldsymbol{y} - \boldsymbol{Y}_i\boldsymbol{c}^* \implies \boldsymbol{y} = \boldsymbol{Y}_i(\boldsymbol{c}^* + \boldsymbol{a}_i)$$

$$\boldsymbol{Y}_{-i}\boldsymbol{a}_{-i} = \boldsymbol{x} = \boldsymbol{y} - \boldsymbol{Y}_i\boldsymbol{c}^* \implies \boldsymbol{y} = \boldsymbol{Y}_i\boldsymbol{c}^* + \boldsymbol{Y}_{-i}\boldsymbol{a}_{-i}$$

Also note that by optimality of $\boldsymbol{a}_{-i}$ we have $||\boldsymbol{a}_{-i}||_1 \leq ||\boldsymbol{c}^*_-||_1$. We have that both $\begin{bmatrix} \boldsymbol{c}^* + \boldsymbol{a}_i \\ \boldsymbol{0} \end{bmatrix}$ and $\begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{a}_{-i} \end{bmatrix}$ are feasible solutions of the original optimization program in (7.2). By condition (7.3) we have

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

which contradicts the optimality of $\begin{bmatrix} \boldsymbol{c}^* \\ \boldsymbol{c}^*_- \end{bmatrix}$. □

7.4 Proof of Theorem 3.6

Theorem. *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* $\mathbb{W}_i$ *be the set of all full-rank submatrices* $\tilde{\boldsymbol{Y}}_i \in \mathbb{R}^{D \times d_i}$ *of* $\boldsymbol{Y}_i$, *where* $rank(\boldsymbol{Y}_i) = d_i$. *If the condition*

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

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

$\|\boldsymbol{X}\|_{1,2}$ denotes the maximum l_2 norm of the columns of the matrix $\boldsymbol{X}$.

Proof. Let $\boldsymbol{x} \in \mathcal{S}_i \cap (\oplus_{j \neq i} \mathcal{S}_j)$. Define $\beta_i \equiv min_{\tilde{\boldsymbol{Y}}_i \in \mathbb{W}_i} \frac{\sqrt{d_i}}{\sigma_i(\tilde{\boldsymbol{Y}}_i)}$ and let $\beta_{-i} \equiv \frac{||\boldsymbol{x}||_2}{max_{j \neq i} cos(\theta_{ij}) ||\boldsymbol{Y}_{-i}||_{1,2}}$, $\sigma_i(\tilde{\boldsymbol{Y}}_i)$ denotes the i-th largest singular value of $\tilde{\boldsymbol{Y}}_i$). We will show that

$$||\boldsymbol{a}_i||_1 \leq \beta_i, \quad \beta_{-i} \leq ||\boldsymbol{a}_{-1}||_1$$

Then, by condition in the theorem we have $||\boldsymbol{a}_i||_1 \leq \beta_i < \beta_{-i} \leq ||\boldsymbol{a}_{-1}||_1$ and we have the desired result by Theorem 3.5.

$||\boldsymbol{a}_i||_1 \leq \beta_i$: Write $\boldsymbol{x} \in \mathcal{S}_i \cap (\oplus_{j \neq i} \mathcal{S}_j)$ as

$$\boldsymbol{x} = \tilde{\boldsymbol{Y}}_i \tilde{\boldsymbol{a}} \implies \tilde{\boldsymbol{a}} = (\tilde{\boldsymbol{Y}}_i^T \tilde{\boldsymbol{Y}}_i)^{-1} \tilde{\boldsymbol{Y}}_i^T \boldsymbol{x}$$

Then using property of norms we have

$$||\tilde{\boldsymbol{a}}||_1 \leq \sqrt{d_i}||\tilde{\boldsymbol{a}}||_2 = \sqrt{d_i}||(\tilde{\boldsymbol{Y}}_i^T\tilde{\boldsymbol{Y}}_i)^{-1}\tilde{\boldsymbol{Y}}_i^T\boldsymbol{x}||_2 \leq \sqrt{d_i}||(\tilde{\boldsymbol{Y}}_i^T\tilde{\boldsymbol{Y}}_i)^{-1}\tilde{\boldsymbol{Y}}_i^T||_{2,2}||\boldsymbol{x}||_2 = \frac{\sqrt{d_i}}{\sigma_{d_i}(\tilde{\boldsymbol{Y}}_i)}||\boldsymbol{x}||_2$$

where $\sigma_{d_i}(\tilde{\boldsymbol{Y}}_i)$ denotes the d_i-th largest singular value of $\tilde{\boldsymbol{Y}}_i$). Hence we have

$$||\boldsymbol{a}_i||_1 \leq min_{\tilde{\boldsymbol{Y}}_i \in \mathbb{W}_i}||\tilde{\boldsymbol{a}}_i||_1 \leq min_{\tilde{\boldsymbol{Y}}_i \in \mathbb{W}_i}\frac{\sqrt{d_i}}{\sigma_i(\tilde{\boldsymbol{Y}}_i)} = \beta_i$$

$\beta_{-i} \leq ||\boldsymbol{a}_{-1}||_1$: We have another representation $\boldsymbol{x} = \boldsymbol{Y}_{-1}\boldsymbol{a}_{-1}$ and hence

$$\boldsymbol{x}^T\boldsymbol{x} = ||\boldsymbol{x}||_2^2 = \boldsymbol{x}^T\boldsymbol{Y}_{-1}\boldsymbol{a}_{-1}$$

By Hölder's inequality with $p, q = 1, \infty$ we have

$$||\boldsymbol{x}||_2^2 \leq ||\boldsymbol{Y}_{-1}^T\boldsymbol{x}||_\infty||\boldsymbol{a}_{-1}||_1$$

And by the definition of smallest principal angle between two subspaces, we can write

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

and hence

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

□

7.5 Proof of Proposition 4.1

$$\mu_e \equiv min_i max_{j\neq i}||\boldsymbol{y}_i||_1, \quad \mu_z \equiv min_i max_{j\neq i}|\boldsymbol{y}_i^T\boldsymbol{y}_j|$$

Lemma 7.1. *The optimization program*

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

has solution $\boldsymbol{c} = \boldsymbol{0}$ *if* $\lambda < ||\boldsymbol{A}^T\boldsymbol{y}||_\infty$

This lemma is proved in [Kim et al., 2007].

Proposition. *Consider the optimization program with noise and sparse outlying entries in (4.5). Without the noise term $\boldsymbol{Z}$, if $\lambda_e \leq 1/\mu_e$, then there exists at least one data point $\boldsymbol{y}_l$ for which in the optimal solution we have $(\boldsymbol{c}_l, \boldsymbol{e}_l) = (\boldsymbol{0}, \boldsymbol{y}_l)$. That is, the data point is taken entirely as outlying entries and is not represented as combination of other data points. Similarly, without the sparse outlying term $\boldsymbol{E}$, if $\lambda_z \leq \mu_z$, then there exists at least one data point $\boldsymbol{y}_l$ for which $(\boldsymbol{c}_l, \boldsymbol{z}_l) = (\boldsymbol{0}, \boldsymbol{y}_l)$.*

Proof. Consider the optimization program without the noise term $\boldsymbol{Z}$ and denote the objective function for a single data point as

$$\phi(\boldsymbol{c}_i, \boldsymbol{e}_i) \equiv ||\boldsymbol{c}_i||_1 + \lambda_e ||\boldsymbol{e}_i||_1$$

By matrix norm properties, for any feasible solution $(\boldsymbol{c}_i, \boldsymbol{e}_i)$ we have

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

Hence we have

$$\phi(\boldsymbol{0}, \boldsymbol{y_i}) = \lambda_e ||\boldsymbol{y}_i||_1 = (\lambda_e max_{j \neq i} ||\boldsymbol{y}_j||_i) ||\boldsymbol{c}_i||_1 + \lambda_e ||\boldsymbol{e}_i||_1$$

Note that if $\lambda_e < \frac{1}{max_{j \neq i} ||\boldsymbol{y}_j||_1}$ then $\phi(\boldsymbol{0}, \boldsymbol{y_i}) \leq \phi(\boldsymbol{c}_i, \boldsymbol{e}_i)$, and hence there exists $l \in \{1, \ldots, N\}$ that we have $(\boldsymbol{0}, \boldsymbol{y}_l)$ as the optimal solution.

Then consider the optimization program without the sparse outlying entries term. That is, we have $\boldsymbol{z}_i = \boldsymbol{y}_i - \boldsymbol{Y}\boldsymbol{c}_i$. Hence the optimization problem can be rewritten as

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

Then by Lemma 7.1, for $\lambda_z < \frac{1}{max_{j \neq i} |\boldsymbol{y}_j^T \boldsymbol{y}_i}$, the optimal solution is $(\boldsymbol{0}, \boldsymbol{y}_i)$. □

7.6 ADMM Solver for the Sparse Optimization Program

In this section we implement the Alternating Direction Method of Multipliers to solve the most general form of sparse optimization program as presented in (4.6):

$$\begin{aligned} &\min_{\boldsymbol{C}, \boldsymbol{E}, \boldsymbol{Z}} ||\boldsymbol{C}||_1 + \lambda_e ||\boldsymbol{E}||_1 + \frac{\lambda_z}{2} ||\boldsymbol{Z}||_F^2 \\ &s.t. \;\; \boldsymbol{Y} = \boldsymbol{Y}\boldsymbol{C} + \boldsymbol{E} + \boldsymbol{Z}, \;\; diag(\boldsymbol{C}) = 0, \;\; \boldsymbol{1}^T \boldsymbol{C} = \boldsymbol{1}^T \end{aligned} \tag{7.4}$$

First, eliminate $\boldsymbol{Z}$ with the equality constraint we can rewrite the problem as

$$\begin{aligned} &\min_{\boldsymbol{C},\boldsymbol{E}} ||\boldsymbol{C}||_1 + \lambda_e||\boldsymbol{E}||_1 + \frac{\lambda_z}{2}||\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{C} - \boldsymbol{E}||_F^2 \\ &s.t. \;\; diag(\boldsymbol{C}) = 0, \;\; \mathbf{1}^T\boldsymbol{C} = \mathbf{1}^T \end{aligned} \tag{7.5}$$

We then introduce an auxiliary matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ and consider the optimization program

$$\begin{aligned} &\min_{\boldsymbol{C},\boldsymbol{E},\boldsymbol{A}} ||\boldsymbol{C}||_1 + \lambda_e||\boldsymbol{E}||_1 + \frac{\lambda_z}{2}||\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{A} - \boldsymbol{E}||_F^2 \\ &s.t. \;\; \boldsymbol{A} = \boldsymbol{C} - diag(\boldsymbol{C}), \;\; \mathbf{1}^T\boldsymbol{A} = \mathbf{1}^T \end{aligned} \tag{7.6}$$

which has the same solution for $(\boldsymbol{C}, \boldsymbol{E})$ with the original problem but helps to make efficient updates. Then we add the constraints into the objective with a parameter $\rho > 0$ to get

$$\begin{aligned} &\min_{\boldsymbol{C},\boldsymbol{E},\boldsymbol{A}} ||\boldsymbol{C}||_1 + \lambda_e||\boldsymbol{E}||_1 + \frac{\lambda_z}{2}||\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{A} - \boldsymbol{E}||_F^2 \\ &+ \frac{\rho}{2}||\boldsymbol{A} - \boldsymbol{C} + diag(\boldsymbol{C})||_2^2 + \frac{\rho}{2}||\mathbf{1}^T\boldsymbol{A} - \mathbf{1}^T||_2^2 \\ &s.t. \;\; \boldsymbol{A} = \boldsymbol{C} - diag(\boldsymbol{C}), \;\; \mathbf{1}^T\boldsymbol{A} = \mathbf{1}^T \end{aligned} \tag{7.7}$$

The regularization terms vanish with feasible solutions. Therefore, adding those terms does not change the optimal solution, but does make the objective function strictly convex in terms of $(\boldsymbol{C}, \boldsymbol{E}, \boldsymbol{A})$. Let $\boldsymbol{\Delta} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{\delta} \in \mathbb{R}^N$ the Lagrange multipliers for the equality constraints, we have the Lagrangian of (7.7) as

$$\begin{aligned} \mathcal{L}(\boldsymbol{C}, \boldsymbol{E}, \boldsymbol{A}, \boldsymbol{\Delta}, \boldsymbol{\delta}) = &||\boldsymbol{C}||_1 + \lambda_e||\boldsymbol{E}||_1 + \frac{\lambda_z}{2}||\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{A} - \boldsymbol{E}||_F^2 \\ &+ \frac{\rho}{2}||\boldsymbol{A} - \boldsymbol{C} + diag(\boldsymbol{C})||_2^2 + \frac{\rho}{2}||\mathbf{1}^T\boldsymbol{A} - \mathbf{1}^T||_2^2 \\ &+ tr(\boldsymbol{\Delta}^T(\boldsymbol{A} - \boldsymbol{C} + diag(\boldsymbol{C}))) + \boldsymbol{\delta}(\mathbf{1}^T\boldsymbol{A} - \mathbf{1}^T) \end{aligned} \tag{7.8}$$

where $tr(.)$ gets the trace of a matrix. The ADMM procedure then iterates as:
Let $(\boldsymbol{C}^{(k)}, \boldsymbol{E}^{(k)}, \boldsymbol{A}^{(k)})$ be the variables at iteration k and $\boldsymbol{\Delta}^{(k)}$, and $\boldsymbol{\delta}^{(k)}$ the Lagrange multipliers at iteration k, then the variables are updates as:
1: Update $\boldsymbol{A}^{(k+1)}$ by minimizing $\mathcal{L}$ with respect the $\boldsymbol{A}$ with all other variables $(\boldsymbol{C}^{(k)}, \boldsymbol{E}^{(k)}, \boldsymbol{\Delta}^{(k)}, \boldsymbol{\delta}^{(k)})$ fixed. By Calculus $\boldsymbol{A}^{(k+1)}$ satisfies the following equation:

$$(\lambda_z\boldsymbol{Y}^T\boldsymbol{Y} + \rho\boldsymbol{I} + \rho\mathbf{1}\mathbf{1}^T)\boldsymbol{A}^{(k+1)} = \lambda_z\boldsymbol{Y}^T(\boldsymbol{Y} - \boldsymbol{E}^{(k+1)}) + \rho(\mathbf{1}\mathbf{1}^T + \boldsymbol{C}^{(k)}) - \mathbf{1}\boldsymbol{\delta}^{(k)T} - \boldsymbol{\Delta}^{(k)} \tag{7.9}$$

2: Update $\boldsymbol{C}^{(k+1)}$ by minimizing $\mathcal{L}$ with respect the $\boldsymbol{C}$ with all other variables $(\boldsymbol{A}^{(k+1)}, \boldsymbol{E}^{(k)}, \boldsymbol{\Delta}^{(k)}, \boldsymbol{\delta}^{(k)})$ fixed. By Calculus $\boldsymbol{C}^{(k+1)}$ has the following closed-form solution:

$$\boldsymbol{C}^{(k+1)} = \boldsymbol{J} - diag(\boldsymbol{J}) \tag{7.10}$$

where

$$\mathcal{T}_{\eta}(v) = (|v| - \eta)_{+} sgn(v) \tag{7.11}$$

$$\boldsymbol{J} = \mathcal{T}_{\frac{1}{\rho}}(\boldsymbol{C}^{(k+1)} + \boldsymbol{\Delta}^{(k)}/\rho) \tag{7.12}$$

$$\tag{7.13}$$

3: Update $\boldsymbol{E}^{(k+1)}$ by minimizing $\mathcal{L}$ with respect the $\boldsymbol{E}$ with all other variables $(\boldsymbol{C}^{(k+1)}, \boldsymbol{A}^{(k+1)}, \boldsymbol{\Delta}^{(k)}, \boldsymbol{\delta}^{(k)})$ fixed. By Calculus $\boldsymbol{E}^{(k+1)}$ has the following closed-form solution:

$$\boldsymbol{E}^{(k+1)} = \mathcal{T}_{\frac{\lambda_e}{\lambda_z}}(\boldsymbol{Y}\boldsymbol{A}^{(k+1)} - \boldsymbol{Y}) \tag{7.14}$$

with the same shrinkage-thresholding operator defined above.

4: Gradient ascent with step size ρ on the Lagrange multipliers as

$$\boldsymbol{\Delta}^{(k+1)} = \boldsymbol{\Delta}^{(k)} + \rho(\boldsymbol{A}^{(k+1)} - \boldsymbol{C}^{(k+1)}) \tag{7.15}$$

$$\boldsymbol{\delta}^{(k+1)} = \boldsymbol{\delta}^{(k)} + \rho(\boldsymbol{A}^{(k+1)T}\mathbf{1} - \mathbf{1}) \tag{7.16}$$

The steps are repeated until the changes are smaller than the error tolerance ε or the number of interactions exceeds a manimum iteration number M. In practice we have $\varepsilon = 10^{-4}$ and $N = 150$.

Algorithm 2: ADMM Procedure to solve (4.6)

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

While (Terminate == False) **do**

1 Update $\boldsymbol{A}^{(k+1)}$ by solving

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

2 Update $\boldsymbol{C}^{(k+1)} = \boldsymbol{J} - diag(\boldsymbol{J})$, where $\boldsymbol{J} = \mathcal{T}_{\frac{1}{\rho}}(\boldsymbol{C}^{(k+1)} + \boldsymbol{\Delta}^{(k)}/\rho)$,

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

4 Update $\boldsymbol{\Delta}^{(k+1)} = \boldsymbol{\Delta}^{(k)} + \rho(\boldsymbol{A}^{(k+1)} - \boldsymbol{C}^{(k+1)})$,

5 Update $\boldsymbol{\delta}^{(k+1)} = \boldsymbol{\delta}^{(k)} + \rho(\boldsymbol{A}^{(k+1)T}\mathbf{1} - \mathbf{1})$,

6 **if** $(max\{||\boldsymbol{A}^{(k+1)} - \boldsymbol{C}^{(k+1)}||_\infty, ||\boldsymbol{A}^{(k+1)T}\mathbf{1} - \mathbf{1}||_\infty, ||\boldsymbol{A}^{(k+1)} - \boldsymbol{A}^{(k)}||_\infty, ||\boldsymbol{E}^{(k+1)} - \boldsymbol{E}^{(k)}||_\infty\} \leq$ ε or $k + 1 \geq$ maxIter):
Terminate = True
end if

7 $k = k + 1$,

end while Output: Sparse coefficient matrix $\boldsymbol{C} = \boldsymbol{C}^{(k)}$
