Exact Subspace Clustering in Linear Time

Shusen Wang, Bojun Tu, Congfu Xu

College of Computer Science and Technology,
Zhejiang University, Hangzhou, China
{wss, tubojun, xucongfu}@zju.edu.cn

Zhihua Zhang

Key Laboratory of Shanghai Education Commission
for Intelligent Interaction and Cognitive Engineering,
Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai, China
zhihua@sjtu.edu.cn

Abstract

Subspace clustering is an important unsupervised learning problem with wide applications in computer vision and data analysis. However, the state-of-the-art methods for this problem suffer from high time complexity—quadratic or cubic in n (the number of data instances). In this paper we exploit a data selection algorithm to speedup computation and the robust principal component analysis to strengthen robustness. Accordingly, we devise a scalable and robust subspace clustering method which costs time only *linear* in n. We prove theoretically that under certain mild assumptions our method solves the subspace clustering problem exactly even for grossly corrupted data. Our algorithm is based on very simple ideas, yet it is the only linear time algorithm with noiseless or noisy recovery guarantee. Finally, empirical results verify our theoretical analysis.

Introduction

Subspace clustering, also well known as subspace segmentation, is an unsupervised learning problem widely studied in the literature (Adler, Elad, and Hel-Or 2013; Babacan, Nakajima, and Do 2012; Elhamifar and Vidal 2009; Favaro, Vidal, and Ravichandran 2011; Li et al. 2012; Liu, Lin, and Yu 2010; Liu, Xu, and Yan 2011; Lu et al. 2012; 2013; Soltanolkotabi and Candès 2011; Vidal 2011; Vidal, Ma, and Sastry 2005; Wang et al. 2011; Wang and Xu 2013). Given n unlabeled data instances $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ drawn from k subspaces $\mathcal{S}_1, \cdots, \mathcal{S}_k \subset \mathbb{R}^d$, subspace clustering aims to find these k underlying subspaces. Subspace clustering has extensive applications in data analysis and computer vision, such as dimensionality reduction (Vidal, Ma, and Sastry 2005), rigid body motion segmentation (Rao et al. 2008; Tron and Vidal 2007; Vidal and Hartley 2004), image clustering under varying illuminations (Ho et al. 2003), image compression (Hong et al. 2006), image segmentation (Yang et al. 2008), system identification (Vidal et al. 2003), etc.

Many approaches have been proposed for dealing with the subspace clustering problem. Among them the most effective ones are the *optimization based methods*, such as sparse subspace clustering (SSC) (Adler, Elad, and Hel-Or 2013; Elhamifar and Vidal 2009; Soltanolkotabi and Candès 2011), low-rank representation (LRR) (Liu, Lin, and Yu 2010; Liu, Xu, and Yan 2011), subspace segmentation via quadratic programming (SSQP) (Wang et al. 2011), groupwise constrained reconstruction method (GCR) (Li et al. 2012), and the trace LASSO based method (Lu et al. 2013). These optimization based methods were illustrated to achieve the state-of-the-art performance in terms of clustering accuracy. In this paper we focus on the optimization based methods. For the readers who are interested in other subspace clustering methods, please refer to the review papers (Vidal 2011; Elhamifar and Vidal 2013).

The major drawback of the optimization-based methods is that their time complexity is high in n. Since the methods work on a so-called self-expression matrix of size $n \times n$, they require at least $\mathcal{O}(n^2)$ time to obtain the self-expression matrix. The time complexity for solving the low-rank representation model (LRR) of (Liu, Lin, and Yu 2010; Liu, Xu, and Yan 2011) is even higher — $\mathcal{O}(n^3)$. As a result, these methods are prohibitive when facing a large number of data instances. The application of the optimization based methods is thereby limited to small-scale datasets.

In this paper we propose a subspace clustering method which requires time only *linear* in n. Our method is based on the idea of data selection. When $n \gg d$, there is much redundancy in the data; a small number of the instances are sufficient for reconstructing the underlying subspaces. Accordingly, we propose to solve the subspace clustering problem in three steps: *selection, clustering,* and *classification*. Our approach first selects a few "informative" instances by certain rules, then performs some subspace clustering algorithm such as SSC or LRR over the selected data, and finally assigns each of the unselected instances to one of the classes. Theoretical analysis shows that our method can significantly reduce computational costs without affecting clustering accuracy. If the subspaces are linearly independent, our method is exact, as well as SSC (Elhamifar and Vidal 2009) and LRR (Liu, Lin, and Yu 2010).

Additionally, we observe that robust principal component analysis (RPCA) (Candès et al. 2011) exactly recovers the underlying sparse and low-rank matrices under certain assumptions which are much milder than that of all the theoretically guaranteed subspace clustering methods. So we propose to employ RPCA for denoising prior to subspace clustering when the data are noisy. Interestingly, this simple

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

idea leads to a much tighter theoretical error bound than the existing subspace clustering methods (Liu, Xu, and Yan 2011; Soltanolkotabi and Candès 2011). By combining RPCA and data selection, we are able to solve the subspace clustering problem exactly in linear time, even in presence of heavy data noise.

In sum, this paper offers the following contributions:

- We devise a data selection algorithm to speedup subspace clustering. Using data selection, subspace clustering can be solved in time linear in n (i.e. the number of data instance), while other optimization based methods all demands time square or cubic in n.
- We prove theoretically that subspace clustering with data selection is still exact on noiseless data. To the best of our knowledge, our work provides the only linear time algorithm with noiseless recovery guarantee.
- We propose to use the ensemble of data selection and RPCA for subspace clustering, such that subspace clustering can be solved exactly even when the data are heavily noisy, and the time complexity is still linear in n. This work is the first one to apply RPCA to subspace clustering; though simple and straightforward, this idea results in the strongest known error bound among all subspace clustering methods.

The remainder of this paper is organized as follows. We first present the notation used in this paper, and make two assumptions which are necessary for the correctness of subspace clustering. Then we we discuss some related work: the optimization based subspace clustering methods and RPCA. Next we describe our method in detail and show the theoretical correctness of our method. Finally we conduct empirical comparisons between our method and three state-of-the-art methods on both synthetic and real-world data.

Notation and Assumptions

For a matrix $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{d\times n}$, let $\mathbf{a}_j$ be its j-th column and $|\mathbf{A}| = [|a_{ij}|]$ be the nonnegative matrix of $\mathbf{A}$. Let $[n]$ denote the set $\{1, \cdots, n\}$, $\mathcal{I} \subset [n]$ be an index set, $\bar{\mathcal{I}}$ be the set $[n] \setminus \mathcal{I}$, and $\mathbf{A}_{\mathcal{I}}$ be an $d \times |\mathcal{I}|$ matrix consisting of the columns of $\mathbf{A}$ indexed by $\mathcal{I}$. We let $\mathbf{A} = \mathbf{U}_{\mathbf{A}}\mathbf{\Sigma}_{\mathbf{A}}\mathbf{V}_{\mathbf{A}}^T$ be the singular value decomposition (SVD) of $\mathbf{A}$ and σ_i be the i-th top singular value of $\mathbf{A}$. Then we define the following matrix norms: $\|\mathbf{A}\|_1 = \sum_{i,j}|a_{ij}|$ denotes the ℓ_1-norm, $\|\mathbf{A}\|_{2,1} = \sum_{j=1}^{n}\|\mathbf{a}_j\|_2$ denotes the $\ell_{2,1}$-norm, $\|\mathbf{A}\|_F = (\sum_{i,j} a_{ij}^2)^{1/2}$ denotes the Frobenius norm, and $\|\mathbf{A}\|_* = \sum_i \sigma_i$ be the nuclear norm of $\mathbf{A}$. Furthermore, let $\mathbf{A}^\dagger = \mathbf{V}_{\mathbf{A}}\mathbf{\Sigma}_{\mathbf{A}}^{-1}\mathbf{U}_{\mathbf{A}}^T$ be the Moore-Penrose inverse of $\mathbf{A}$, and $\mathbf{B}\mathbf{B}^\dagger\mathbf{A}$ be the projection of $\mathbf{A}$ onto the column space of $\mathbf{B}$.

We make the following conventions throughout this paper: n is the number of data instances and can be very large; d is the dimension of each instance and is usually small; $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n]$ is the data matrix. The n instances are drawn from k subspaces $\mathcal{S}_1, \cdots, \mathcal{S}_k$, each of dimension $r_1, \cdots, r_k$, respectively. We let $r_1 + \cdots + r_k = r$. All of the provable methods require the following two assumptions (Elhamifar and Vidal 2009; Liu, Lin, and Yu 2010; Lu et al. 2012; Wang et al. 2011).

Assumption 1 (Inter-Subspace Independence)**.** *The k subspaces are linearly independent: $\mathcal{S}_i \cap \mathcal{S}_j = \{\mathbf{0}\}$ for all $i \neq j$. That is, no instance within a subspace can be expressed as a linear combination of instances on the rest subspaces. This assumption implies that $r = r_1 + \cdots + r_k = \mathrm{rank}(\mathbf{X})$.*

Assumption 2 (Inner-Subspace Dependence)**.** *For any subspace $\mathcal{S}_i$ ($i \in [k]$), the index set*

$$\mathcal{L} = \left\{ l \in [n] \;\middle|\; \mathbf{x}_l \in \mathcal{S}_i \right\}$$

cannot be partitioned into two non-overlapping sets $\mathcal{L}'$ and $\mathcal{L}''$ such that

$$\mathrm{Range}\big(\mathbf{X}_{\mathcal{L}'}\big) \bigcap \mathrm{Range}\big(\mathbf{X}_{\mathcal{L}''}\big) = \{\mathbf{0}\}.$$

Here $\mathrm{Range}(\mathbf{A})$ *denotes the column space of matrix* $\mathbf{A}$.

The two assumptions are necessary for the correctness of subspace clustering. If Assumption 1 is violated, which implies that some instances can lie on multiple subspaces simultaneously, there are ambiguities in such instances. If Assumption 2 is violated, there are at least $k+1$ linearly independent subspaces underlying the data, which leads to that the result of subspace clustering is not unique.

Related Work

Our work relies on the optimization-based subspace clustering methods (Elhamifar and Vidal 2009; Liu, Lin, and Yu 2010) and RPCA (Candès et al. 2011). We describe these methods in this section.

Optimization Based Subspace Clustering Methods

Optimization based methods are the most effective approaches to the subspace clustering problem. The methods include *sparse subspace clustering (SSC)* (Elhamifar and Vidal 2009; Soltanolkotabi and Candès 2011), *low-rank representation (LRR)* (Liu, Lin, and Yu 2010; Liu, Xu, and Yan 2011), *subspace segmentation via quadratic programming (SSQP)* (Wang et al. 2011), etc.

Typically, the optimization based methods are based on the idea of "self expression;" that is, each instance is expressed as a linear combination of the rest instances. Specifically, one represents $\mathbf{x}_i$ as $\mathbf{x}_i = \sum_{j\neq i} w_{ij}\mathbf{x}_j$ and imposes some regularization such that the resulting weight $w_{ij} = 0$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are drawn from different subspaces. By solving such an optimization problem, one obtains the $n \times n$ self-expression matrix $\mathbf{W}$, which reflects the affinity between the instances; i.e., large $|w_{ij}|$ implies that $\mathbf{x}_i$ and $\mathbf{x}_j$ are more likely on the same subspace. Finally, spectral clustering is applied upon the similarity matrix $\frac{1}{2}|\mathbf{W} + \mathbf{W}^T|$ to obtain the clustering results.

Under Assumptions 1 and 2, SSC and LRR solve the subspace clustering problem exactly (Elhamifar and Vidal 2009; Liu, Lin, and Yu 2010). By further assuming that the angles between the subspaces are large enough, SSQP also solves the problem exactly (Wang et al. 2011). If the data are contaminated by noise, only SSC and LRR are guaranteed theoretically, which was studied in (Soltanolkotabi and Candès 2011; Liu, Xu, and Yan 2011). Notice that SSC and LRR require relatively strong assumptions over the data

noise. For example, they allow only a small fraction of the n instances corrupted, and LRR additionally requires the magnitude of the noise be small.

In this paper we are mainly concerned with SSC and LRR, because they have been theoretically guaranteed for either clean data or contaminated data. But our method is not restricted to SSC and LRR; all subspace clustering methods can fit in our framework. SSC seeks to obtain the self-expression matrix $\mathbf{W}$ by solving the following optimization problem:

$$\min_{\mathbf{W}} \|\mathbf{X} - \mathbf{X}\mathbf{W}\|_F^2 + \lambda\|\mathbf{W}\|_1; \quad \text{s.t. } \mathsf{diag}(\mathbf{W}) = \mathbf{0}. \quad (1)$$

This takes at least $\mathcal{O}(n^2)$ time if d is regarded as a constant. LRR is built on the following optimization problem:

$$\min_{\mathbf{W}} \|\mathbf{X} - \mathbf{X}\mathbf{W}\|_{2,1} + \lambda\|\mathbf{W}\|_*, \quad (2)$$

which can be solved by the alternating direction method with multiplier (ADMM) in $\mathcal{O}(n^3)$ time (Lin, Liu, and Su 2011). Although another algorithm in (Lin, Liu, and Su 2011) solves this model in $\mathcal{O}(n^2 r)$ time, in our off-line experiments, it is sometimes numerically instable when r is not sufficiently small. So we use ADMM to solve LRR in this paper.

Our work is also related to the distributed low-rank subspace segmentation method of (Talwalkar et al. 2013), which is motivated by the divide-and-conquer matrix factorization idea (Mackey, Talwalkar, and Jordan 2011). This method can also perform subspace clustering exactly under some assumptions, but the theoretical result is weaker than ours.

Robust Principal Component Analysis (RPCA)

The RPCA model (Candès et al. 2011) is a powerful tool for low-rank and sparse matrix recovery. Assume that the observed data matrix $\mathbf{X}$ is the superposition of a low-rank matrix $\mathbf{L}_0$ and a sparse matrix $\mathbf{S}_0$. RPCA recovers the low-rank component and the sparse component respectively from the observation $\mathbf{X}$ by the following optimization model:

$$\min_{\mathbf{L},\mathbf{S}} \|\mathbf{S}\|_1 + \lambda\|\mathbf{L}\|_*; \qquad \text{s.t. } \mathbf{L} + \mathbf{S} = \mathbf{X}. \quad (3)$$

Many algorithms have been proposed for solving RPCA, among which ADMM is the most widely used. The ADMM algorithm of (Lin et al. 2009) runs in time $\mathcal{O}(\min\{n^2 d, nd^2\})$ for any $d \times n$ matrix $\mathbf{X}$.

The theoretical analysis of (Candès et al. 2011) shows that RPCA exactly recovers the low-rank component $\mathbf{L}_0$ and the sparse component $\mathbf{S}_0$ from the observation $\mathbf{X}$ with probability near 1, provided that the rank of $\mathbf{L}_0$ is not high and the number of nonzero entries in $\mathbf{S}_0$ is not large. We observe that the assumptions made by RPCA are much milder than those of SSC (Soltanolkotabi and Candès 2011) and LRR (Liu, Xu, and Yan 2011). The analysis for SSC (Soltanolkotabi and Candès 2011) and LRR (Liu, Xu, and Yan 2011) allows only a small fraction of the instances to be contaminated. The analysis for LRR (Liu, Xu, and Yan 2011) makes further assumptions on the magnitude of data noise. In contrast, RPCA does not make such requirements at all. We therefore suggest using RPCA for preprocessing before subspace clustering. Interestingly, this simple idea results in the strongest error bound for subspace clustering.

Algorithm 1 Exact Subspace Clustering in Linear Time

1: **Input:** data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$, classes number k.
2: **Denoising**: process $\mathbf{X}$ using RPCA when the data are noisy;
3: **Normalization**: for $l = 1, \cdots, n$, normalize $\mathbf{x}_l$ such that $\|\mathbf{x}_l\| = 1$;
4: **Selection**: $\mathbf{X}_{\mathcal{I}} \leftarrow$ select a fraction of columns of $\mathbf{X}$ by Algorithm 2;
5: **Clustering**: conduct subspace clustering for $\mathbf{X}_{\mathcal{I}}$ using a base method, e.g., SSC or LRR;
6: **Classification**: assign each column of $\mathbf{X}_{\bar{\mathcal{I}}}$ to one of the k classes.

Algorithm 2 Data Selection

Input: data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$, class number k, $r = \mathrm{rank}(\mathbf{X})$.
Initialize $\mathcal{I}_0 = \{p\}$ for an arbitrary index $p \in [n]$;
for $i = 1, \cdots, r-1$ **do**
    Compute the residual $\mathbf{D} = \mathbf{X} - \mathbf{X}_{\mathcal{I}_0}\mathbf{X}_{\mathcal{I}_0}^{\dagger}\mathbf{X}$;
    Select an arbitrary index $q \in \bar{\mathcal{I}}_0$ such that $\|\mathbf{d}_q\| > 0$;
    Add q to $\mathcal{I}_0$;
end for
$\mathcal{I} \leftarrow \mathcal{I}_0$; *(its cardinality is $|\mathcal{I}_0| = r$)*
Build a "*disjoint sets*" (Cormen et al. 2001, Chapter 21) data structure (denoted by *DS*) which initially contains r sets: $\mathcal{D}_1 = \{1\}, \cdots, \mathcal{D}_r = \{r\}$;
repeat
    Select an arbitrary index from $\bar{\mathcal{I}}$ without replacement, say l;
    Solve $\mathbf{x}_l = \mathbf{X}_{\mathcal{I}_0}\boldsymbol{\beta}$ to obtain $\boldsymbol{\beta}^{\text{opt}}$;
    Compute $\mathcal{J} = \left\{j \;\middle|\; |\beta_j^{\text{opt}}| > 0\right\}$;
    for each pair $\{i, j\} \subset \mathcal{J}$ **do**
        In *DS*, find the sets containing i and j;
        if the two sets are disjoint **then**
            Add l to $\mathcal{I}$ if $l \notin \mathcal{I}$, and union the two sets;
        end if
    end for
until there are no more than k sets in *DS*;
(Optional) Uniformly sample $\mathcal{O}(r)$ additional indices from $\bar{\mathcal{I}}$ and add them to $\mathcal{I}$;
return $\mathbf{X}_{\mathcal{I}}$.

Methodology

Our subspace clustering method is sketched in Algorithm 1 and described in detail in the following subsections. After the denoising and normalization preprocess, our method runs in three steps: *selection, clustering,* and *classification*. Our method costs $\mathcal{O}(nr^2 d + nd^2)$ time which is linear in n, so it is especially useful when $n \gg d, r$. We defer the theoretical analysis to the next section.

Denoising

We assume that the data are drawn from a union of low-rank subspaces and then perturbed by noise and outliers. We thus employ RPCA to process the observed data $\mathbf{X}$. The RPCA model (3) can be solved in $\mathcal{O}(nd^2)$ time by ADMM (Lin et al. 2009). After solving (3), we replace $\mathbf{X}$ by its low-rank component $\mathbf{L}$. Under some mild assumptions, $\mathbf{L}$ is exactly the uncontaminated data drawn from the low-rank subspaces.

Data Selection

Our data selection algorithm is sketched in Algorithm 2. It selects $\mathcal{O}(r)$ instances in $\mathcal{O}(nr^2d)$ time. Here r is the sum of the dimensions of the subspaces.

The subspace clustering problem requires recovering the k subspaces from $\mathbf{X}$. If the selected data $\mathbf{X}_\mathcal{I}$ also satisfy Assumptions 1 and 2, then the underlying k subspaces will be recovered exactly from $\mathbf{X}_\mathcal{I}$. So the aim of the data selection algorithm is to make $\mathbf{X}_\mathcal{I}$ satisfy Assumptions 1 and 2. When $n \gg d$, we show in Theorem 1 that a small subset of the n instances suffices for reconstructing the k subspaces.

Theorem 1. *Let $\mathbf{X}_\mathcal{I}$ be the matrix consists of the data selected by Algorithm 2. We have that* $\mathrm{Range}(\mathbf{X}_\mathcal{I}) = \mathrm{Range}(\mathbf{X})$. *Furthermore, if the given data $\mathbf{X}$ satisfies Assumptions 1 and 2, the selected data $\mathbf{X}_\mathcal{I}$ also satisfies Assumptions 1 and 2. Algorithm 2 runs in $\mathcal{O}(nr^2d)$ time.*

More specifically, $\mathrm{Range}(\mathbf{X}_\mathcal{I}) = \mathrm{Range}(\mathbf{X})$ is achieved by Lines 2 to 7 in the algorithm. Since $\mathbf{X}$ satisfies Assumption 1, $\mathbf{X}_\mathcal{I}$ satisfies Assumption 1 trivially. Furthermore, the selection procedure in Lines 9 to 20 makes $\mathbf{X}_\mathcal{I}$ satisfy Assumption 2.

Empirically, we find it better to implement the selection operations in Line 5 and 11 randomly than deterministically, and uniform sampling works well in practice. Furthermore, Line 21 is optional and it does not affect the correctness of the algorithm. When the data are clean, Line 21 is not necessary; but when the data are perturbed by noise, Line 21 can make the algorithm more robust.

Base Method

We have discussed previously that some methods fulfill subspace clustering exactly when the data satisfy Assumptions 1 and 2. Theorem 1 ensures that, if the given data $\mathbf{X}$ satisfy Assumptions 1 and 2, then the selected data instances $\mathbf{X}_\mathcal{I}$ also satisfy Assumptions 1 and 2. So we can safely apply such a subspace clustering method upon the selected data $\mathbf{X}_\mathcal{I}$ to reconstruct the underlying subspaces. We refer to such a subspace clustering method as *base method*. For example, SSC or LRR can be employed as a base method.

Classifying the Unselected Data Instances

After clustering the selected instances, the k subspaces are reconstructed. The remaining problem is how to classify the unselected instances into the k classes. It can be solved by linear regression.

Let $\mathcal{I}$ be the index set corresponding to the selected instances, and let $\mathcal{I}_1, \cdots, \mathcal{I}_k$ be the clustering results of Line 5 in Algorithm 1. We have $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset$ for all $i \neq j$, and $\mathcal{I}_1 \cup \cdots \cup \mathcal{I}_k = \mathcal{I}$. Without loss of generality, we can rearrange the columns of $\mathbf{X}_\mathcal{I}$ to be $\mathbf{X}_\mathcal{I} = \left[\mathbf{X}_{\mathcal{I}_1}, \cdots, \mathbf{X}_{\mathcal{I}_k}\right]$. Then for each of the unselected instances, say $l \in \bar{\mathcal{I}}$, we solve the least-squares regression problem

$$\mathbf{b}^{\text{opt}} = \operatorname*{argmin}_{\mathbf{b}} \left\|\mathbf{x}_l - \mathbf{X}_\mathcal{I}\mathbf{b}\right\|^2 = \operatorname*{argmin}_{\mathbf{b}} \left\|\mathbf{x}_l - \sum_{i=1}^{k} \mathbf{X}_{\mathcal{I}_i}\mathbf{b}_{(i)}\right\|^2.$$

Letting $\mathbf{b}^{\text{opt}^T} = \left[\mathbf{b}_{(1)}^{\text{opt}\,T}, \cdots, \mathbf{b}_{(k)}^{\text{opt}\,T}\right]$, we assign $\mathbf{x}_l$ to the j-th class where

$$j = \operatorname*{argmax}_{i \in [k]} \left\|\mathbf{b}_{(i)}^{\text{opt}}\right\|.$$

In addition, if the column number of $\mathbf{X}_\mathcal{I}$, i.e. the number of selected instances, is greater than d, we impose a small ℓ_2-norm penalty on $\mathbf{b}$ in the least square regression, which has a closed form solution

$$\mathbf{b}^{\text{opt}} = \left(\mathbf{X}_\mathcal{I}^T\mathbf{X}_\mathcal{I} + \epsilon\mathbf{I}\right)^{-1}\mathbf{X}_\mathcal{I}^T\mathbf{x}_l,$$

where $\epsilon > 0$ is the weight of the ℓ_2-norm penalty. Here one can set ϵ to be a small constant.

Theoretical Analysis

In this section we show theoretically that our method solves the subspace clustering problem exactly under certain mild assumptions. We present our theoretical analysis for noise-free data and noisy data respectively in the two subsections. In both cases, our method requires only $\mathcal{O}(n)$ time as d and r stay constant.

Subspace Clustering for Noise-free Data

If the data are noise-free, our method has the same theoretical guarantee as SSC (Elhamifar and Vidal 2009) and LRR (Liu, Lin, and Yu 2010), which is stronger than SSQP (Wang et al. 2011).

Theorem 2 (Exact Subspace Clustering for Noise-free Data)**.** *Let Assumptions 1 and 2 hold for the given data $\mathbf{X}$. If the base method (Line 5 in Algorithm 1) solves the subspace clustering problem exactly, then Algorithm 1 (without the denoising step) also solves the problem exactly for $\mathbf{X}$. Moreover, the time complexity of Algorithm 1 is linear in n.*

Subspace Clustering for Noisy Data

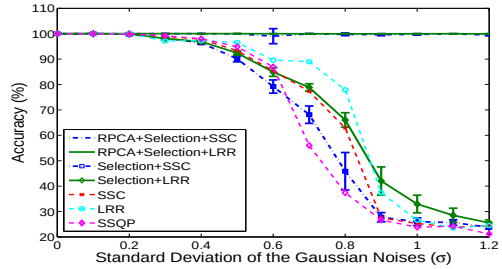
If a small proportion of the entries in the data matrix are contaminated by data noise, no matter what the magnitude of the noise is, our method is guaranteed to solve the subspace clustering problem exactly.

Suppose we are given a $d \times n$ data matrix $\mathbf{X} = \mathbf{X}_0 + \mathbf{S}$, $(n \geq d)$, where $\mathbf{X}_0$ denotes the n instances drawn from k subspaces of dimensions $r_1, \cdots, r_k$, and $\mathbf{S}$ captures the noise. Theorem 3 follows directly from Theorem 2 and (Candès et al. 2011, Theorem 1.1).

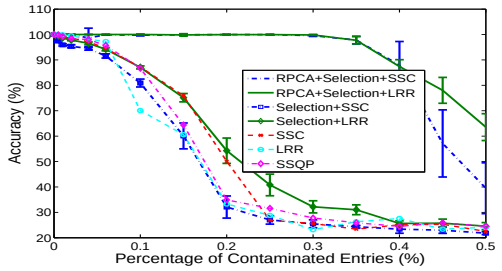
Theorem 3 (Exact Subspace Clustering for Noisy Data)**.** *Suppose that Assumptions 1 and 2 hold for $\mathbf{X}_0$, and the support of $\mathbf{S}$ is uniformly distributed among all sets of cardinality m. Then there exists a constant c such that with probability at least $1 - cn^{-10}$, Algorithm 1 solves the subspace clustering problem exactly for $\mathbf{X}$, provided that*

$$r_1 + \cdots + r_k \leq \rho_r\mu^{-1}d(\log n)^{-2} \quad \textit{and} \quad m \leq \rho_s nd,$$

where ρ_r and ρ_s are constants, and μ is the incoherence constant for $\mathbf{X}_0$. Moreover, the time complexity of Algorithm 1 is linear in n.



(a) Results on the data with 30% entries contaminated by i.i.d. Gaussian noise $\mathcal{N}(0, \sigma^2)$.



(b) Results on the data contaminated by a proportion of independent Bernoulli noise ± 1.

Figure 1: Subspace clustering accuracy on the synthetic data.

Theorem 3 is the tightest known error bound for the subspace clustering problem. Theorem 3 is much stronger than the error bounds of SSC (Soltanolkotabi and Candès 2011) and LRR (Liu, Xu, and Yan 2011). Both SSC and LRR allow only a small fraction, not all, of the n instances to be contaminated. Additionally, LRR requires the magnitude of the noise be small. In contrast, our method does not make such requirements at all.

Experiments

In this section we empirically evaluate our method in comparison with three state-of-the-art methods which have theoretical guarantees: sparse subspace clustering (SSC) (Elhamifar and Vidal 2009), low-rank representation method (LRR) (Liu, Lin, and Yu 2010), and subspace segmentation via quadratic programming (SSQP) (Wang et al. 2011). Although some other methods like the groupwise constrained reconstruction (Li et al. 2012) and spectral local best-fit flats (Zhang et al. 2012) achieves comparable accuracy with SSC and LRR, they have no theoretical guarantee on subspace clustering accuracy, so we do not compare with them. We conduct experiments on both synthetic data and real-world data. We denote our method by "Selection+SSC" if it employs SSC as the base method and "RPCA+Selection+SSC" if it further invokes RPCA for denoising; the notations "Selection+LRR" and "RPCA+Selection+LRR" are similarly defined.

Setup

We conduct experiments on a workstation with Intel Xeon 2.4GHz CPU, 24GB memory, and 64bit Windows Server 2008 system. Our method is implemented in MATLAB; for the three compared methods, we use the MATLAB code released by their authors. To compare the running time, we set MATLAB in the single thread mode. Each of the SSC, LRR and SSQP methods has one tuning parameter, while our method has two tuning parameters: one for RPCA and one for the base method. The parameters of each method are all tuned best for each input dataset.

More specifically, our method has the following settings. We use data selection only when the instance number n is large, say $n > 20r$. In the data selection step, Line 21 in Algorithm 2 is used such that exactly $\min\{20r, n\}$ instances are selected. The selection operations in Line 5 and 11 in Algorithm 2 are uniformly at random. In the classification step (Line 6 in Algorithm 1), we use the least square regression method (with a very small fixed ℓ_2-norm penalty when the number of selected instances is greater than d).

We report the clustering accuracy and elapsed time of the compared methods. When the data selection algorithm is applied, the results are nondeterministic, so we repeat the algorithms ten times and report the means and standard deviations of the clustering accuracy.

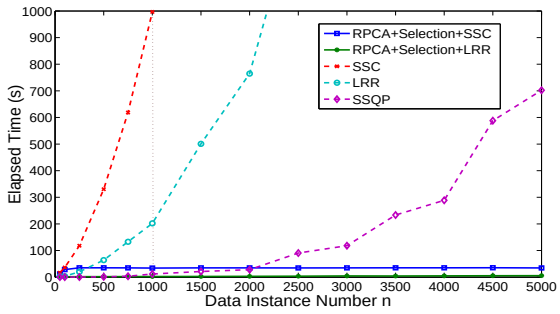


Figure 2: The growth of lapsed time with data instance number n.

Synthetic Datasets

We generate synthetic data in the same way as that in (Liu, Lin, and Yu 2010; Wang et al. 2011). The bases of each subspace are generated by $\mathbf{U}^{(l+1)} = \mathbf{T}\mathbf{U}^{(l)} \in \mathbb{R}^{d \times r_l}$, where $\mathbf{T} \in \mathbb{R}^{d \times d}$ is a random rotation matrix, and $\mathbf{U}^{(1)}$ is a random matrix with orthogonal columns. Then we randomly sample $\frac{n}{k}$ instances from the l-th subspace by $\mathbf{X}_0^{(l)} = \mathbf{U}^{(l)}\mathbf{Q}^{(l)}$, where the sampling matrix $\mathbf{Q}^{(l)} \in \mathbb{R}^{r_l \times \frac{n}{k}}$ is a standard Gaussian random matrix. We denote $\mathbf{X}_0 = [\mathbf{X}_0^{(1)}, \cdots, \mathbf{X}_0^{(k)}]$.

In the first set of experiments, we fix the data size to be $n = 1000$, $d = 100$, $k = 5$, $r_1 = \cdots = r_5 = 4$, and then add varying noise to $\mathbf{X}_0$ to form data matrix $\mathbf{X}$. The noise is generated in the following ways.

- Noises with increasing magnitude. We add Gaussian noise i.i.d. from $\mathcal{N}(0, \sigma^2)$ to 30% entries of $\mathbf{X}_0$ to construct $\mathbf{X}$. We vary σ from 0 to 1.2 and report the clustering accuracy in Figure 1(a).

Table 1: Descriptions of the Hopkins155 Database and the Pen-Based Handwritten Digit Dataset.

Datasets in Hopkins155 (Tron and Vidal 2007)			Handwritten Digit Dataset (Alimoglu and Alpaydin 1996)		
#instance	#feature	#class	#instance	#feature	#class
hundreds	tens	2 or 3	$10,992$	16	10

Table 2: Clustering accuracy and elapsed time of each method on the Hopkins155 Database and the Pen-Based Handwritten Digit Dataset.

	Selection+SSC	Selection+LRR	SSC	LRR	SSQP
Hopkins155 database (Tron and Vidal 2007)					
Accuracy (%)	97.15 ± 0.45	96.80 ± 0.39	98.71	97.87	98.50
Time (s)	$4,508$	217	$12,750$	$11,516$	$3,144$
Pen-Based Handwritten Digit Dataset (Alimoglu and Alpaydin 1996)					
Accuracy (%)	71.07 ± 4.78	71.96 ± 2.79	73.98	75.90	73.84
Time (s)	172	39	$42,596$	1.17×10^6	3.19×10^5

- Noises added to an increasing proportion of entries. We then add i.i.d. Bernoulli noise of ± 1 to $p\%$ entries of $\mathbf{X}_0$ to construct $\mathbf{X}$. We vary $p\%$ from 0 to 50% and report the clustering accuracy in Figure 1(b).

In the second set of experiments, we analyze the growth of the computation time with n. Here we fix $d = 100$, $k = 5$, $r_1 = \cdots = r_5 = 4$, but vary n. We generate an $d \times n$ data matrix $\mathbf{X}_0$ and then add Gaussian noise i.i.d. from $\mathcal{N}(0, 0.5^2)$ to 30% entries of the data matrix $\mathbf{X}_0$. We vary n from 50 to $5,000$, and plot the running time of each method in Figure 2.

The empirical results fully agree with our theoretical analysis. That is,

- The results in Figure 2 demonstrate the efficiency of our method. Figure 2 shows that the elapsed time of our method grows linearly (not constantly) in n, which is much lower than the growth of SSC, LRR, and SSQP.
- The results in Figure 1 show that using RPCA for denoising significantly improves the clustering accuracy. When RPCA is invoked, the clustering results are not affected by the magnitude of noise (see Figure 1(a)), and the methods can still work well when all of the n instances are contaminated (see Figure 1(b)).
- The results in Figure 1 also show that, for the noisy data (i.e. when RPCA is not invoked), using data selection does not do much harm to clustering accuracy.

In sum, the above empirical results show that using data selection and RPCA in subspace clustering is a great success. Data selection brings tremendous speedup, and RPCA makes subspace clustering more robust. Specifically, when $n \gg d, r$, using data selection for subspace clustering leads to significant speedup without affecting the accuracy much. Moreover, when all of the n data instances are contaminated or when the magnitude of noise is large, SSC and LRR are liable to fail, while the RPCA+Selection ensemble method still works well. This verifies our theoretical analysis.

Real-World Datasets

We also evaluate the subspace clustering methods on real-world datasets—the Hopkins155 Database and the Pen-Based Handwritten Digit Dataset—which are described in Table 1 . The Hopkins155 Database contains 155 datasets with varying instance numbers and feature numbers. We choose the datasets because they satisfy that $n \gg d$, which is the assumption of our method.

For the Hopkins155 Database, we set up the experiments according to (Elhamifar and Vidal 2009; Liu, Lin, and Yu 2010; Wang et al. 2011). For SSC, SSQP, and our method, we project the data onto 12-dimension subspace by PCA; for LRR, we use the original data without projection; such preprocess leads to higher accuracy, as was reported in (Elhamifar and Vidal 2009; Liu, Lin, and Yu 2010; Wang et al. 2011). Additionally, for our method, we do not use RPCA (Line 2 in Algorithm 1) because the data are nearly noise free. We report in Table 2 the average clustering accuracy and total elapsed time on the 155 datasets.

For the Pen-Based Handwritten Digit Dataset, we directly apply the subspace clustering methods without preprocessing the data. We do not use RPCA for our method because the feature size $d = 16$ is too small to satisfy the requirements by Theorem 3. The clustering accuracy and elapsed time are also reported in Table 2.

The results on the real-world data demonstrate the effectiveness and efficiency of our method. Our method achieves clustering accuracy nearly as good as the state-of-the-art methods, while the computational costs are dramatically reduced. Especially, on the Handwritten Digit Dataset where n is large, LRR spends 13 days to fulfill the task, while our "Selection+LRR" costs less than a minute to achieve a comparable clustering accuracy!

Conclusions

In this paper we have proposed a subspace clustering method which has lower time complexity and tighter theoretical bound than the state-of-the-art methods. The method employs data selection for speeding up computation and RPCA for denoising, and the ensemble of the two techniques yields an exact solution in time linear in n even for grossly corrupted data. The experiments have further verified the effectiveness and efficiency of our method.

Acknowledgement

Wang is supported by Microsoft Research Asia Fellowship 2013 and the Scholarship Award for Excellent Doctoral

Student granted by Chinese Ministry of Education. Xu is supported by the National Natural Science Foundation of China (No. 61272303) and the National Program on Key Basic Research Project of China (973 Program, No. 2010CB327903). Zhang is supported by the National Natural Science Foundation of China (No. 61070239).

References

Adler, A.; Elad, M.; and Hel-Or, Y. 2013. Probabilistic subspace clustering via sparse representations. *Signal Processing Letters, IEEE* 20(1):63–66.

Alimoglu, F., and Alpaydin, E. 1996. Methods of combining multiple classifiers based on different representations for pen-based handwriting recognition. In *Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium*.

Babacan, S.; Nakajima, S.; and Do, M. 2012. Probabilistic low-rank subspace clustering. In *Advances in Neural Information Processing Systems (NIPS)*.

Candès, E. J.; Li, X.; Ma, Y.; and Wright, J. 2011. Robust principal component analysis? *Journal of the ACM* 58(3):11:1–11:37.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2001. *Introduction to algorithms, 2ed*. MIT press.

Elhamifar, E., and Vidal, R. 2009. Sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Elhamifar, E., and Vidal, R. 2013. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(11):2765–2781.

Favaro, P.; Vidal, R.; and Ravichandran, A. 2011. A closed form solution to robust subspace estimation and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ho, J.; Yang, M.-H.; Lim, J.; Lee, K.-C.; and Kriegman, D. 2003. Clustering appearances of objects under varying illumination conditions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hong, W.; Wright, J.; Huang, K.; and Ma, Y. 2006. Multi-scale hybrid linear models for lossy image representation. *IEEE Transactions on Image Processing* 15(12):3655–3671.

Li, R.; Li, B.; Zhang, K.; Jin, C.; and Xue, X. 2012. Group-wise constrained reconstruction for subspace clustering. In *International Conference on Machine Learning (ICML)*.

Lin, Z.; Chen, M.; Wu, L.; and Ma, Y. 2009. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *UIUC Technical Report, UILU-ENG-09-2215*.

Lin, Z.; Liu, R.; and Su, Z. 2011. Linearized alternating direction method with adaptive penalty for low-rank representation. In *Advances in Neural Information Processing Systems (NIPS)*.

Liu, G.; Lin, Z.; and Yu, Y. 2010. Robust subspace segmentation by low-rank representation. In *International Conference on Machine Learning (ICML)*.

Liu, G.; Xu, H.; and Yan, S. 2011. Exact subspace segmentation and outlier detection by low-rank representation. *CoRR* abs/1109.1646.

Lu, C. Y.; Zhu, L.; Yan, S.; and Huang, D. 2012. Robust and efficient subspace segmentation via least squares regression. In *European Conference on Computer Vision (ECCV)*.

Lu, C.; Feng, J.; Lin, Z.; and Yan, S. 2013. Correlation adaptive subspace segmentation by trace lasso. In *International Conference on Computer Vision (ICCV)*.

Mackey, L. W.; Talwalkar, A.; and Jordan, M. I. 2011. Divide-and-conquer matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*.

Rao, S.; Tron, R.; Ma, Y.; and Vidal, R. 2008. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Soltanolkotabi, M., and Candès, E. 2011. A geometric analysis of subspace clustering with outliers. *arXiv preprint arXiv:1112.4258*.

Talwalkar, A.; Mackey, L.; Mu, Y.; Chang, S.-F.; and Jordan, M. I. 2013. Distributed low-rank subspace segmentation. In *International Conference on Computer Vision (ICCV)*.

Tron, R., and Vidal, R. 2007. A benchmark for the comparison of 3-d motion segmentation algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Vidal, R., and Hartley, R. 2004. Motion segmentation with missing data using power factorization and gpca. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Vidal, R.; Soatto, S.; Ma, Y.; and Sastry, S. 2003. An algebraic geometric approach to the identification of a class of linear hybrid systems. In *Proceeding of the 42nd IEEE Conference on Decision and Control*.

Vidal, R.; Ma, Y.; and Sastry, S. 2005. Generalized principal component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(12):1–15.

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

Wang, Y.-X., and Xu, H. 2013. Noisy sparse subspace clustering. In *International Conference on Machine Learning (ICML)*.

Wang, S.; Yuan, X.; Yao, T.; Yan, S.; and Shen, J. 2011. Efficient subspace segmentation via quadratic programming. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Yang, A.; Wright, J.; Ma, Y.; and Sastry, S. 2008. Unsupervised segmentation of natural images via lossy data compression. *Computer Vision and Image Understanding* 110(2):212–225.

Zhang, T.; Szlam, A.; Wang, Y.; and Lerman, G. 2012. Hybrid linear modeling via local best-fit flats. *International journal of computer vision* 100(3):217–240.

Proof of Theorems

Key Lemmas

Lemma 4. *Let $\mathbf{X}_{\mathcal{I}}$ be the matrix consists of the data selected by Algorithm 2. We have that*

$$\mathrm{Range}\big(\mathbf{X}_{\mathcal{I}_0}\big) = \mathrm{Range}\big(\mathbf{X}_{\mathcal{I}}\big) = \mathrm{Range}\big(\mathbf{X}\big).$$

Proof. It is easy to show that the column space of $\mathbf{X}_{\mathcal{I}_0} \in \mathbb{R}^{d\times r}$ is exactly the same as $\mathbf{X}$, i.e., $\mathrm{Range}\big(\mathbf{X}_{\mathcal{I}_0}\big) = \mathrm{Range}\big(\mathbf{X}\big)$, because $\mathbf{X}$ has a rank at most r. And since $\mathcal{I}_0 \subset \mathcal{I} \subset [n]$, we have that

$$\mathrm{Range}\big(\mathbf{X}_{\mathcal{I}_0}\big) \subset \mathrm{Range}\big(\mathbf{X}_{\mathcal{I}}\big) \subset \mathrm{Range}\big(\mathbf{X}\big),$$

and thus $\mathrm{Range}\big(\mathbf{X}_{\mathcal{I}_0}\big) = \mathrm{Range}\big(\mathbf{X}_{\mathcal{I}}\big) = \mathrm{Range}\big(\mathbf{X}\big)$. □

Lemma 5. *Assume the given data $\mathbf{X}$ satisfies Assumptions 1 and 2. Lines 9 to 20 in Algorithm 2 select at most $r-k$ columns. Finally, in the disjoint sets data structure, the instances on the same subspace are in the same set, and instances on different subspaces are in different sets. This procedure costs $\mathcal{O}(nr^2d)$ time.*

Proof. Here we study Lines 9 to 20 in Algorithm 2. After Line 9, there are r sets in the disjoint set data structure *DS*. Now we show that, under Assumptions 1 and 2, the r sets will be combined into k disjoint sets.

Let $\mathcal{S}_1, \cdots, \mathcal{S}_k$ be the k underlying subspaces. Assumption 1 ensures that we can partition the set $\mathcal{I}_0$ into k non-overlapping sets $\mathcal{I}_0^{(1)}, \cdots, \mathcal{I}_0^{(k)}$, such that the columns of $\mathbf{X}$ indexed by $\mathcal{I}_0^{(p)}$ are all on the subspace $\mathcal{S}_p$. Lemma 4 ensures $\mathrm{Range}\big(\mathbf{X}_{\mathcal{I}_0}\big) = \mathrm{Range}\big(\mathbf{X}\big)$, so we have that for any $l \in [n]$ there is some $\boldsymbol{\beta} \in \mathbb{R}^r$ such that $\mathbf{x}_l = \mathbf{X}_{\mathcal{I}_0}\boldsymbol{\beta}$. Without loss of generality, we can rearrange the indices such that

$$\mathbf{X}_{\mathcal{I}_0} = [\mathbf{X}_{\mathcal{I}_0^{(1)}}, \cdots, \mathbf{X}_{\mathcal{I}_0^{(k)}}], \qquad \boldsymbol{\beta}^T = [\boldsymbol{\beta}_{(1)}^T, \cdots, \boldsymbol{\beta}_{(k)}^T].$$

If $\mathbf{x}_l \in \mathcal{S}_p$, Assumption 1 implies that

$$\mathbf{x}_l = \sum_{i=1}^{k} \mathbf{X}_{\mathcal{I}_0^{(i)}}\boldsymbol{\beta}_{(i)} = \mathbf{X}_{\mathcal{I}_0^{(p)}}\boldsymbol{\beta}_{(p)},$$

that is,

$$\boldsymbol{\beta}_{(i)} = \mathbf{0} \ \text{ for all } \ i \neq p.$$

Thus instances from different subspaces can never appear in the same set (in the disjoint set data structure *DS*) according to the criterion in Line 14. It remains to be shown that data instances on the same subspace will surely be grouped into the same set (in the disjoint set data structure).

If the set $\mathcal{I}_0^{(p)}$ could be further partitioned into two non-overlapping sets $\mathcal{J}$ and $\mathcal{K}$ such that

$$\mathrm{Range}\big(\mathbf{X}_{\mathcal{J}}\big) \bigcap \mathrm{Range}\big(\mathbf{X}_{\mathcal{K}}\big) = \{0\}$$

and the instances in $\{\mathbf{x}_i \,|\, i \in [n], \mathbf{x}_i \in \mathcal{S}_p\}$ were in either $\mathrm{Range}(\mathbf{X}_{\mathcal{J}})$ or $\mathrm{Range}(\mathbf{X}_{\mathcal{K}})$, then Assumption 2 would be violated. Thus there does not exist two non-overlapping index sets $\mathcal{J}$ an $\mathcal{K}$ such that for any index $l \in \big\{i \in [n] \,|\, \mathbf{x}_i \in \mathcal{S}_p, i \notin \mathcal{I}_0^{(p)}\big\}$, the solution $\boldsymbol{\beta}_{(p)}^{\text{opt}}$ to the linear system

$$\mathbf{x}_l = \mathbf{X}_{\mathcal{I}_0^{(p)}}\boldsymbol{\beta}_{(p)}$$

is supported on either $\mathcal{J}$ or $\mathcal{K}$. Therefore, finally in the data structure *DS*, the instances on the same subspace are in the same set; otherwise $\mathcal{I}_0^{(p)}$ could be partitioned into such two non-overlapping sets $\mathcal{J}$ and $\mathcal{K}$.

Hence after Line 20, in the disjoint set data structure *DS*, there are exactly k sets, and the instances on the same subspace are in the same set.

After Line 4 there are r non-overlapping sets. If a column is selected in Line 17, then at least two sets are merged, and the number of sets in *DS* decreases by at least 1. Therefore, after at most $r-k$ columns being selected, there are k disjoint sets in *DS*.

Now we analyze the time complexity. Solving the linear equations in Line 12 requires $\mathcal{O}(r^2d)$ time. Since $\boldsymbol{\beta}$ is a r dimension vector, so there are no more than r^2 pairs in Line 14. The disjoint set data structure support union and find in near constant time (Cormen et al. 2001, Chapter 21), so Lines 14 to 19 require $\mathcal{O}(r^2)$ time. This procedure runs in at most $n-r$ loops, so the overall time complexity is $\mathcal{O}(nr^2d)$. □

Proof of Theorem 1

Proof. Firstly, Lemma 4 shows that $\mathrm{Range}\big(\mathbf{X}_{\mathcal{I}_0}\big) = \mathrm{Range}\big(\mathbf{X}_{\mathcal{I}}\big) = \mathrm{Range}\big(\mathbf{X}\big)$.

Assumption 1 holds for the data $\mathbf{X}_{\mathcal{I}}$ trivially. Now we prove Assumption 2 holds for $\mathbf{X}_{\mathcal{I}}$. Let $\mathcal{S}_1, \cdots, \mathcal{S}_k$ be the k underlying subspaces. We can partition the set $\mathcal{I}$ into k non-overlapping sets $\mathcal{I}^{(1)}, \cdots, \mathcal{I}^{(k)}$ according to the k sets in the data structure *DS*. Lemma 5 indicates that the instances indexed by $\mathcal{I}^{(p)}$ are on a same subspace $\mathcal{S}_p$, for any $p \in [k]$. The construction in the proof of Lemma 5 ensure that $\mathcal{I}^{(p)}$ cannot be partitioned into two non-overlapping sets $\mathcal{J}$ and $\mathcal{K}$ such that

$$\mathrm{Range}\big(\mathbf{X}_{\mathcal{J}}\big) \bigcap \mathrm{Range}\big(\mathbf{X}_{\mathcal{K}}\big) = \{0\}.$$

Thus Assumption 2 also holds for the data $\mathbf{X}_{\mathcal{I}}$.

Lines 2 to 7 run in r loops; each loop costs $\mathcal{O}(nrd)$ time. Lines 9 to 20 costs $\mathcal{O}(nr^2d)$ time, which is shown by Lemma 5. Thus Algorithm 2 costs totally $\mathcal{O}(nr^2d)$ time. □

Proof of Theorem 2

Proof. When the data are noise free, Algorithm 1 simply skips the denoising step. Theorem 1 shows that the column space of $\mathbf{X}_{\mathcal{I}}$ contains all of the k subspaces, and that the selected data instances $\mathbf{X}_{\mathcal{I}}$ satisfies Assumptions 1 and 2. So the clustering result of Line 5 exactly reconstructs the k subspaces. Then we need only to assign the unselected data, i.e., those indexed by $\bar{\mathcal{I}}$, to their nearest subspaces. This can be done in $\mathcal{O}(nr^2d)$ time, and the result is also exact because the subspaces are independent.

In Algorithm 1, Line 2 costs $\mathcal{O}(nd^2)$ time, Line 3 costs $\mathcal{O}(nd)$ time, Line 4 costs $\mathcal{O}(nr^2d)$ time, Line 5 costs $\mathcal{O}(r^2)$ or $\mathcal{O}(r^3)$ time, and Line 6 cost $\mathcal{O}(nr^2d)$ time. Thus the time complexity of our method is $\mathcal{O}(nr^2d+nd^2)$, which is linear in n. □