Robust Non-Linear Matrix Factorization for Dictionary Learning, Denoising, and Clustering

Jicong Fan, Chengrun Yang, Madeleine Udell

Abstract—**Low dimensional nonlinear structure abounds in datasets across computer vision and machine learning. Kernelized matrix factorization techniques have recently been proposed to learn these nonlinear structures for denoising, classification, dictionary learning, and missing data imputation, by observing that the image of the matrix in a sufficiently large feature space is low-rank. However, these nonlinear methods fail in the presence of sparse noise or outliers. In this work, we propose a new robust nonlinear factorization method called Robust Non-Linear Matrix Factorization (RNLMF). RNLMF constructs a dictionary for the data space by factoring a kernelized feature space; a noisy matrix can then be decomposed as the sum of a sparse noise matrix and a clean data matrix that lies in a low dimensional nonlinear manifold. RNLMF is robust to sparse noise and outliers and scales to matrices with thousands of rows and columns. Empirically, RNLMF achieves noticeable improvements over baseline methods in denoising and clustering.**

Index Terms—**Matrix factorization, denoising, subspace clustering, dictionary learning, kernel method.**

I. Introduction

REAL data or signals are often corrputed by noise or outliers. As such, data denoising is a core task across applications in computer vision, machine learning, data mining and signal processing. Many denoising strategies exploit the difference between the distribution of the data (structured, coherent) and the distribution of the noise (independent). Perhaps the simplest latent structure is low-rank structure, which appears throughout a wide range of applications [1] and undergirds celebrated algorithms such as principal component analysis (PCA) [2], [3], robust PCA (RPCA) [4], and low-rank matrix completion [5], [6], [7]. For instance, RPCA aims to decompose a partially corrupted matrix as the sum of a low-rank matrix and a sparse matrix, thus separating the sparse noise from the low rank data. Another well-known latent structure is sparsity, or more specifically, the property that data vectors can be modeled as a sparse linear combination of basis elements. Sparse structure appears in compressed sensing [8], subspace clustering [9], [10], [11], [12], dictionary learning [13], [14], [15], [16], image classification [17], [18], noise/outlier identification [19], [20], and semi-supervised learning [21], [22]. For instance, the self-expressive models widely used in subspace clustering [9], [10], [23] exploit the fact that each data point can be represented as a linear combination of a few data points lying in the same subspace and hence are able to recognize noise and outliers when the data are drawn from a union of low-dimensional subspaces. In [22], the authors proposed to explicitly pursue structured (block-diagonal) sparsity for robust representation with partially labeled data. In recent years, a number of kernel methods [24], [25], [26], [27], [28] and deep learning methods [29], [30], [31], [32] have been proposed to remove noise from data with nonlinear low-dimensional latent structures. Nevertheless, the kernel denoising methods have high time and space complexities. Most of the deep-learning-based denoising methods require clean data samples (e.g. noiseless images), as supervision, to train the neural networks. In this paper, we focus on unsupervised denoising. More recently, a few kernelized factorization methods [33], [34], [35], [36] have been proposed for nonlinear dictionary learning but they are not able to handle sparse noise. To solve the problem, we in this paper provide a robust nonlinear matrix factorization method and apply it to dictionary learning, denoising, and clustering.

II. Related work and our contribution

A. Robust principal component analysis

Suppose a data matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ is partially corrupted by sparse noise $\boldsymbol{E} \in \mathbb{R}^{m\times n}$ to form noisy observations $\hat{\boldsymbol{X}} = \boldsymbol{X} + \boldsymbol{E}$. The robust PCA (RPCA) model in [4] assumes that $\boldsymbol{X}$ is low-rank and aims to solve

$$\underset{\boldsymbol{X},\boldsymbol{E}}{\text{minimize}} \;\; \|\boldsymbol{X}\|_* + \lambda\|\boldsymbol{E}\|_1, \quad \text{subject to } \boldsymbol{X}+\boldsymbol{E}=\hat{\boldsymbol{X}}, \quad (1)$$

where $\|\boldsymbol{X}\|_*$ denotes the matrix nuclear norm (sum of singular values; a convex relaxation of matrix rank) of $\boldsymbol{X}$. RPCA cannot effectively denoise data with nonlinear latent structure, as the corresponding matrix $\boldsymbol{X}$ is often of high rank. In [28], a robust kernel PCA (RKPCA) was proposed to remove sparse noise from nonlinear data. The space and time complexities of the naive algorithm are $O(n^2)$ and $O(n^3)$ respectively, to store an $n \times n$ kernel matrix and compute its singular value decomposition (SVD). Hence RKPCA cannot handle large-scale data. In addition, RKPCA has no *out-of-sample extension*: a method to reduce the noise of new data (rows of the data matrix $\boldsymbol{X}$) efficiently.

B. Robust dictionary learning and subspace clustering

Classical dictionary learning [37] and sparse coding algorithms [13], [38], [39], [40] denoise by projecting onto d

Jicong Fan is with the School of Data Science, The Chinese University of Hong Kong (Shenzhen) and Shenzhen Research Institute of Big Data, Shenzhen, China. Email: fanjicong@cuhk.edu.cn

Chengrun Yang and Madeleine Udell are with the School of Operations Research and Information Engineering, Cornell University, Ithaca, NY 14853, USA. Email: {cy438,udell}@cornell.edu

dictionary atoms. The dictionary matrix $\boldsymbol{D} \in \mathbb{R}^{m\times d}$ and sparse coefficient matrix $\boldsymbol{C} \in \mathbb{R}^{d\times n}$ are found by solving

$$\underset{\boldsymbol{D}\in\mathcal{S}_D,\boldsymbol{C}\in\mathcal{S}_C}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{D}\boldsymbol{C}\|_F^2, \tag{2}$$

$$\text{or} \qquad \underset{\boldsymbol{D}\in\mathcal{S}_D,\boldsymbol{C}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{D}\boldsymbol{C}\|_F^2 + \lambda\|\boldsymbol{C}\|_1, \tag{3}$$

where $\mathcal{S}_D := \{\boldsymbol{S} \in \mathbb{R}^{m\times d} : \|\boldsymbol{s}_j\| \leq 1, \forall j = 1,\ldots,d\}$[1] and $\mathcal{S}_C := \{\boldsymbol{S} \in \mathbb{R}^{d\times n} : \|\boldsymbol{s}_j\|_0 \leq k, \forall j = 1,\ldots,n\}$.

Formulations (2) and (3) cannot handle sparse noise and outliers. Consequently, several robust dictionary learning (RDL) algorithms [19], [41], [42], [20], [43], [44] have been proposed. For instance, [19] proposed to solve

$$\underset{\boldsymbol{D}\in\mathcal{S}_D,\boldsymbol{C}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{D}\boldsymbol{C}\|_1 + \lambda\|\boldsymbol{C}\|_1, \tag{4}$$

using both batch and online optimization approaches. In [20], the authors proposed to set threshold ε and solve

$$\underset{\boldsymbol{D}\in\mathcal{S}_D,\boldsymbol{C}}{\text{minimize}} \quad \sum_{j=1}^{n}\min(\|\boldsymbol{x}_j-\boldsymbol{D}\boldsymbol{c}_j\|_2,\varepsilon) + \lambda\|\boldsymbol{C}\|_1, \tag{5}$$

to limit the contribution of outliers to the objective.

Problem (4) may fail when the data are corrupted by small dense noise. Problem (5) finds outliers but cannot recover the clean data because the hard-thresholding parameter ε eliminates the representation loss for the outliers and then sets the corresponding columns of $\boldsymbol{C}$ to zero.

In contrast, the following RDL formulation extended from (4) [19] simultaneously learns the dictionary and identifies the noise:

$$\underset{\boldsymbol{D}\in\mathcal{S}_D,\boldsymbol{C},\boldsymbol{E}}{\text{minimize}} \ \frac{1}{2}\|\hat{\boldsymbol{X}}-\boldsymbol{D}\boldsymbol{C}-\boldsymbol{E}\|_F^2 + \lambda_C\|\boldsymbol{C}\|_1 + \lambda_E\mathcal{R}(\boldsymbol{E}). \tag{6}$$

Here $\hat{\boldsymbol{X}}$ is the observed noisy matrix, $\boldsymbol{E}$ models the noise or outliers, and $\mathcal{R}(\boldsymbol{E})$ penalizes errors $\boldsymbol{E}$. For example, when $\hat{\boldsymbol{X}}$ contains sparse noise, we set $\mathcal{R}(\boldsymbol{E}) = \|\boldsymbol{E}\|_1$. The $\boldsymbol{D}$ obtained from (6) can be used to denoise new data efficiently.

The formulation (6) is closely related to a few other proposals. For example, by setting $\boldsymbol{D} = \hat{\boldsymbol{X}}$, one derives the self-expressive model used in sparse subspace clustering (SSC) [9]. Moreover, replacing $\|\boldsymbol{C}\|_1$ with $\|\boldsymbol{C}\|_*$, one gets the low-rank representation (LRR) [10] model. A few extensions of SSC and LRR for subspace clustering can be found in [45], [46], [47], [48]. We may use SSC and LRR to remove additive noise and outliers. For instance, when a few columns of $\boldsymbol{X}$ are outliers, SSC and LRR can identify the outliers by encouraging $\boldsymbol{E}$ to be column-wise sparse. When $\boldsymbol{X}$ is corrupted by sparse noise, we set $\mathcal{R}(\boldsymbol{E}) = \|\boldsymbol{E}\|_1$. The recovered matrix is $\hat{\boldsymbol{X}} - \boldsymbol{E}$. Compared to RDL (6), in terms of denoising, the major advantage of SSC and LRR is they are nonconvex. However, since the dictionary used in SSC and LRR is the noisy data matrix, the denoising performance may degrade. In addition, SSC and LRR are not effective in denoising new data.

C. Kernel dictionary learning

In recent years, several authors have proposed to augment dictionary learning with kernel methods to learn nonlinear structures [33], [34], [35], [36], [49]. For instance, [33] proposed to solve

$$\begin{aligned}\underset{\boldsymbol{D}\in\check{\mathcal{S}}_D,\boldsymbol{C}\in\mathcal{S}_C}{\text{minimize}} \ & \frac{1}{2}\|\phi(\hat{\boldsymbol{X}})-\phi(\hat{\boldsymbol{X}})\boldsymbol{D}\boldsymbol{C}-\phi(\hat{\boldsymbol{X}})\text{diag}(\boldsymbol{w})\|_F^2 \\ & + \lambda_C\|\boldsymbol{C}\|_0 + \lambda_w\|\boldsymbol{w}\|_0,\end{aligned} \tag{7}$$

where ϕ denotes the feature map induced by a kernel function and $\check{\mathcal{S}}_D := \{\boldsymbol{S} \in \mathbb{R}^{n\times d} : \|\boldsymbol{s}_j\| = 1, \forall j = 1,2,\ldots,d\}$. The nonzeros of $\boldsymbol{w}$ in (7) identify the outliers in $\hat{\boldsymbol{X}}$. In [36], the following problem was considered:

$$\underset{\boldsymbol{D}\in\bar{\mathcal{S}}_D,\boldsymbol{C}\in\mathcal{S}_C}{\text{minimize}} \quad \frac{1}{2}\|\phi(\hat{\boldsymbol{X}})-\phi(\boldsymbol{D})\boldsymbol{C}\|_F^2, \tag{8}$$

where $\bar{\mathcal{S}}_D := \{\boldsymbol{S} \in \mathbb{R}^{m\times d} : \|\boldsymbol{s}_j\| = 1, \forall j = 1,2,\ldots,d;\ \boldsymbol{s}_i^\top\boldsymbol{s}_j = \mu, \forall i \neq j\}$, and μ is a predefined parameter.

One advantage of (8) over (7) is that the computational cost was reduced if $n \gg m$. Nevertheless, (8) is vulnerable to outliers. Moreover, neither (7) nor (8) can identify sparse noise in $\hat{\boldsymbol{X}}$ and recover the clean data. The reason is that the denoised matrix itself does not appear in the objective functions.

D. Contributions of this paper

Our contributions are three-fold. First, we propose a new robust nonlinear matrix factorization model together with an effective optimization algorithm that explicitly separates the sparse noise or outliers from the observed data. Second, we provide theory to prove correctness of our factorization in the feature space induced by kernels, and justify the use of squared Frobenius norm regularization on the feature matrix and coefficient matrix in the factorization model. Finally, based on the robust nonlinear matrix factorization model, we propose a new subspace clustering method. Extensive experiments on synthetic data, real image data, and real motion capture data showed that our proposed methods are more effective than baseline methods in dictionary learning, denoising, and subspace clustering. The MATLAB codes of the proposed methods are publicly available at *https://github.com/jicongfan/Robust-Nonlinear-Matrix-Factorization.*

III. Robust Non-Linear Matrix Factorization

A. Low-rank factorization in kernel feature space

Suppose the columns of a data matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ come from a generative model $\mathcal{M}$. Let $\phi : \mathbb{R}^m \to \mathbb{R}^l$ be the feature map induced by a kernel function

$$\mathcal{K}(\boldsymbol{x}_i,\boldsymbol{x}_j) = \langle\phi(\boldsymbol{x}_i),\phi(\boldsymbol{x}_j)\rangle = \phi(\boldsymbol{x}_i)^\top\phi(\boldsymbol{x}_j).$$

Two popular kernels are the polynomial (Poly) kernel and Gaussian radial basis function (RBF) kernel

$$\begin{aligned}\text{Poly}: \ & \mathcal{K}_{c,q}(\boldsymbol{x}_i,\boldsymbol{x}_j) = (\boldsymbol{x}_i^\top\boldsymbol{x}_j + c)^q \\ \text{RBF}: \ & \mathcal{K}_\sigma(\boldsymbol{x}_i,\boldsymbol{x}_j) = \exp\left(-\tfrac{1}{\sigma^2}\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2\right),\end{aligned}$$

[1] Throughout this paper, given a matrix $\boldsymbol{X}$, we denote its j-th column by $\boldsymbol{x}_j$, and denote its entry at location (i,j) by x_{ij}.

where c, q, and σ are hyper-parameters.

Let $\phi(\boldsymbol{X}) = [\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), \ldots, \phi(\boldsymbol{x}_n)]$. When $\phi(\boldsymbol{X})$ is low-rank or can be well approximated by a low-rank matrix, we will seek a factorization of the form

$$\phi(\boldsymbol{X}) \simeq \phi(\boldsymbol{D})\boldsymbol{C}, \tag{9}$$

where $\boldsymbol{D} \in \mathbb{R}^{m\times d}$ is a dictionary of d atoms, $\boldsymbol{C} \in \mathbb{R}^{d\times n}$ is the coefficient matrix, and $d < \min\{l, n\}$. This factorization model was also considered in [33], [36], [49].

We denote the feature map of the polynomial kernel by $\phi_{c,q}$. Then $\phi_{c,q}(\boldsymbol{X}) \in \mathbb{R}^{\binom{m+q}{q}\times n}$. In [49], [50], the authors assumed that the data generating model $\mathcal{M}$ is a union of p-degree polynomials with random coefficients, $f^{\{j\}} : \mathbb{R}^r \to \mathbb{R}^m$, $j = 1, \ldots, k$, $r \ll m$; for $j = 1, \ldots, k$, the n/k columns of $\boldsymbol{X}$ are given by $\boldsymbol{x} = f^{\{j\}}(\boldsymbol{z})$, and $\boldsymbol{z} \in \mathbb{R}^r$ consists of r uncorrelated random variables; see the following formulation

$$\begin{aligned}\boldsymbol{X} = [&f^{\{1\}}(\boldsymbol{z}_1), \ldots, f^{\{1\}}(\boldsymbol{z}_{n/k}),\\ &f^{\{2\}}(\boldsymbol{z}_{n/k+1}), \ldots, f^{\{2\}}(\boldsymbol{z}_{2n/k}), \ldots,\\ &f^{\{k\}}(\boldsymbol{z}_{n-n/k+1}), \ldots, f^{\{k\}}(\boldsymbol{z}_n)]\boldsymbol{P},\end{aligned}$$

where $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ is an unknown permutation matrix. They [49] showed that

$$\operatorname{rank}(\boldsymbol{X}) = \min\{m, n, k\tbinom{r+p}{p}\}, \tag{10}$$

and

$$\operatorname{rank}(\phi_{c,q}(\boldsymbol{X})) = \min\{\tbinom{m+q}{q}, n, k\tbinom{r+pq}{pq}\}. \tag{11}$$

Thus, when p is large, $\boldsymbol{X}$ is high rank; but $\phi_{c,q}(\boldsymbol{X})$ is low-rank provided that n is large enough.

Let ϕ_σ be the feature map of the Gaussian RBF kernel. The following reveals the connection between two types of kernels.

Lemma 1. *Define* $s_{ij} := \exp\left(-\frac{\|\boldsymbol{x}_i\|^2+\|\boldsymbol{x}_j\|^2+2c}{2\sigma^2}\right)$. *Then for any* $c \geq 0$, σ, $\boldsymbol{x}_i$, *and* $\boldsymbol{x}_j$,

$$\mathcal{K}_\sigma(\boldsymbol{x}_i, \boldsymbol{x}_j) = s_{ij}\sum_{u=0}^{\infty}\frac{\mathcal{K}_{c,u}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sigma^{2u}u!}.$$

It follows from Lemma 1 that

$$\phi_\sigma(\boldsymbol{x}_i) = s_i\left[w_0\phi_{c,0}^\top(\boldsymbol{x}_i), w_1\phi_{c,1}^\top(\boldsymbol{x}_i), \ldots, w_\infty\phi_{c,\infty}^\top(\boldsymbol{x}_i)\right]^\top,$$

where $s_i = \exp\left(-\frac{\|\boldsymbol{x}_i\|^2+c}{2\sigma^2}\right)$ and $w_u = 1/(\sigma^u\sqrt{u!})$ for $u = 0, 1, \ldots, \infty$. Therefore, $\phi_\sigma(\boldsymbol{X}) \in \mathbb{R}^{\infty\times n}$ is full rank, according to (11). Let $\boldsymbol{S}_X = \operatorname{diag}(s_1, \ldots, s_n)$ and $\boldsymbol{S}_D = \operatorname{diag}\left(\exp(-\frac{\|\boldsymbol{d}_1\|^2+c}{2\sigma^2}), \ldots, \exp(-\frac{\|\boldsymbol{d}_d\|^2+c}{2\sigma^2})\right)$. We have

Lemma 2. *Define* $\kappa_1 = \max\{0.5n, \sqrt{dn}\|\boldsymbol{C}\|_F, 0.5d\|\boldsymbol{C}\|_2\|\boldsymbol{C}\|_F\}$ *and* $\kappa_2 = \max\{\max_i\|\boldsymbol{x}_i\|^2, \max_j\|\boldsymbol{d}_j\|^2\}$. *Suppose* $\sigma^2 > \kappa_2 + c$. *Then for any* $q \geq 0$,

$$\begin{aligned}&\frac{1}{2}\|\phi_\sigma(\boldsymbol{X}) - \phi_\sigma(\boldsymbol{D})\boldsymbol{C}\|_F^2\\ =&\sum_{u=0}^{q}\frac{w_u^2}{2}\|\phi_{c,u}(\boldsymbol{X})\boldsymbol{S}_X - \phi_{c,u}(\boldsymbol{D})\boldsymbol{S}_D\boldsymbol{C}\|_F^2 + R,\end{aligned}$$

where $|R| \leq \dfrac{3\kappa_1\exp(-\frac{c}{\sigma^2})}{q!}\left(\dfrac{\kappa_2+c}{\sigma^2}\right)^q$.

Lemma 2 shows the factorization error of the feature matrix induced by the RBF kernel can be well approximated by the weighted sum of the factorization errors of the feature matrices induced by polynomial kernels of different degrees if σ and q are sufficiently large. Notice that $\operatorname{rank}(\phi_{c,u}(\boldsymbol{X})\boldsymbol{S}_X) = \operatorname{rank}(\phi_{c,u}(\boldsymbol{X}))$ because $\boldsymbol{S}_X$ is diagonal with positive diagonal entries. The following corollary of Lemma 2 provides an upper bound on the factorization error using the RBF kernel:

Corollary 1. *Suppose* $d \geq \operatorname{rank}(\phi_{c,q}(\boldsymbol{X}))$ *and* $\sigma^2 > \kappa_2 + c$. *Then for any* $q \geq 1$, *there exist* $\boldsymbol{D}$ *and* $\boldsymbol{C}$ *such that*

$$\frac{1}{2}\|\phi_\sigma(\boldsymbol{X}) - \phi_\sigma(\boldsymbol{D})\boldsymbol{C}\|_F^2 \leq \frac{3\kappa_1\exp(-\frac{c}{\sigma^2})}{q!}\left(\frac{\kappa_2+c}{\sigma^2}\right)^q.$$

Remark 1. Since κ_1 relates to $\|\boldsymbol{C}\|_F$, the bound in Corollary 1 may be tighter when $\|\boldsymbol{C}\|_F^2$ is smaller.

Hence, when $d \geq k\binom{r+pq}{pq}$ and n, σ are sufficiently large, our factorization model (9) holds as follows:

$$\phi_{c,q}(\boldsymbol{X}) = \phi_{c,q}(\boldsymbol{D})\boldsymbol{C} \quad\text{and}\quad \phi_\sigma(\boldsymbol{X}) \approx \phi_\sigma(\boldsymbol{D})\boldsymbol{C}.$$

When using polynomial kernel or Gaussian RBF kernel, we can exactly or approximately factorize $\phi(\boldsymbol{X})$ into $\phi(\boldsymbol{D})$ and $\boldsymbol{C}$, where d could be much smaller than n. This property enables us to extract nonlinear features (rows of $\boldsymbol{C}$), find useful basis elements (columns of $\boldsymbol{D}$), or remove noise from $\boldsymbol{X}$.

B. Robustness in data space

1) General objective function: Suppose a data matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ is partially corrupted by sparse noise $\boldsymbol{E} \in \mathbb{R}^{m\times n}$ to form noisy observations

$$\hat{\boldsymbol{X}} = \boldsymbol{X} + \boldsymbol{E}, \tag{12}$$

where the locations of nonzero entries of $\boldsymbol{E}$ are uniform and random. We wish to recover $\boldsymbol{X}$ from $\hat{\boldsymbol{X}}$. Using (9) and (12), we define the factorization loss as

$$\begin{aligned}\mathcal{L}(\boldsymbol{D}, \boldsymbol{C}, \boldsymbol{E}) :=& \frac{1}{2}\|\phi(\hat{\boldsymbol{X}} - \boldsymbol{E}) - \phi(\boldsymbol{D})\boldsymbol{C}\|_F^2 \qquad (13)\\ =&\frac{1}{2}\operatorname{Tr}\left(\phi(\hat{\boldsymbol{X}} - \boldsymbol{E})^\top\phi(\hat{\boldsymbol{X}} - \boldsymbol{E})\right) - \operatorname{Tr}\left(\boldsymbol{C}^\top\phi(\boldsymbol{D})^\top\phi(\hat{\boldsymbol{X}} - \boldsymbol{E})\right)\\ &+\frac{1}{2}\operatorname{Tr}\left(\boldsymbol{C}^\top\phi(\boldsymbol{D})^\top\phi(\boldsymbol{D})\boldsymbol{C}\right),\end{aligned}$$

where $\operatorname{Tr}(\cdot)$ denotes the matrix trace. Using the kernel, we have $\phi(\hat{\boldsymbol{X}} - \boldsymbol{E})^\top\phi(\hat{\boldsymbol{X}} - \boldsymbol{E}) = \mathcal{K}(\hat{\boldsymbol{X}} - \boldsymbol{E}, \hat{\boldsymbol{X}} - \boldsymbol{E}) \in \mathbb{R}^{n\times n}$, $\phi(\boldsymbol{D})^\top\phi(\hat{\boldsymbol{X}} - \boldsymbol{E}) = \mathcal{K}(\boldsymbol{D}, \hat{\boldsymbol{X}} - \boldsymbol{E}) \in \mathbb{R}^{d\times n}$, and $\phi(\boldsymbol{D})^\top\phi(\boldsymbol{D}) = \mathcal{K}(\boldsymbol{D}, \boldsymbol{D}) \in \mathbb{R}^{d\times d}$. Hence from (13),

$$\begin{aligned}\mathcal{L}(\boldsymbol{D}, \boldsymbol{C}, \boldsymbol{E}) =& \frac{1}{2}\operatorname{Tr}\left(\mathcal{K}(\hat{\boldsymbol{X}} - \boldsymbol{E}, \hat{\boldsymbol{X}} - \boldsymbol{E})\right)\\ &- \operatorname{Tr}\left(\boldsymbol{C}^\top\mathcal{K}(\boldsymbol{D}, \hat{\boldsymbol{X}} - \boldsymbol{E})\right) + \frac{1}{2}\operatorname{Tr}\left(\boldsymbol{C}^\top\mathcal{K}(\boldsymbol{D}, \boldsymbol{D})\boldsymbol{C}\right).\end{aligned}$$

In addition, we define the regularization as

$$\mathcal{R}(\boldsymbol{D}, \boldsymbol{C}, \boldsymbol{E}) := \lambda_D\mathcal{R}(\boldsymbol{D}) + \lambda_C\mathcal{R}(\boldsymbol{C}) + \lambda_E\mathcal{R}(\boldsymbol{E}),$$

where λ_D, λ_C, and λ_E are penalty parameters. Then we propose to solve

$$\underset{\boldsymbol{D},\boldsymbol{C},\boldsymbol{E}}{\text{minimize}}\ \ \mathcal{L}(\boldsymbol{D}, \boldsymbol{C}, \boldsymbol{E}) + \mathcal{R}(\boldsymbol{D}, \boldsymbol{C}, \boldsymbol{E}) \triangleq \mathcal{J}(\boldsymbol{D}, \boldsymbol{C}, \boldsymbol{E}). \tag{14}$$

Remark 2. In (14), we can introduce a constraint $\boldsymbol{D}=\hat{\boldsymbol{X}}-\boldsymbol{E}$ to eliminate $\boldsymbol{D}$, which leads to a self-expressive model, i.e. represent $\phi(\hat{\boldsymbol{X}}-\boldsymbol{E})$ with itself multiplied by $\boldsymbol{C}$; then we only need to compute $\boldsymbol{C}$ and $\boldsymbol{E}$. However, as $\boldsymbol{C}\in\mathbb{R}^{n\times n}$, the space and time complexities will increase significantly.

2) Noise-specific penalty on $\boldsymbol{E}$: We suggest choosing penalties of $\boldsymbol{E}$ from below, based on the suspected noise distribution.

- When all entries of $\boldsymbol{X}$ are corrupted by Gaussian noise, i.e. $E_{ij}\sim\mathcal{N}(0,\epsilon^2)$, $\forall\ (i,j)\in[m]\times[n]$, we set $\mathcal{R}(\boldsymbol{E})=\frac{1}{2}\|\boldsymbol{E}\|_F^2$.
- When $\boldsymbol{X}$ is partially and randomly corrupted, i.e., $\boldsymbol{E}$ is a sparse matrix and $\mathbb{P}[E_{ij}\neq 0]=\rho$, we set $\mathcal{R}(\boldsymbol{E})=\|\boldsymbol{E}\|_1$, where $0<\rho<1$ and $\|\cdot\|_1$ denotes the ℓ_1 norm of vector or matrix serving as a convex relaxation of the ℓ_0 norm.
- When a few columns of $\boldsymbol{X}$ are corrupted by Gaussian noise, we set $\mathcal{R}(\boldsymbol{E})=\|\boldsymbol{E}\|_{2,1}$, where $\|\boldsymbol{E}\|_{2,1}=\sum_{j=1}^n\|\boldsymbol{e}_j\|_2$ is a convex relaxation of the number of nonzero columns of $\boldsymbol{E}$.

Next, we detail the choices of $\mathcal{R}(\boldsymbol{D})$ and $\mathcal{R}(\boldsymbol{C})$.

3) Smooth penalty on $\boldsymbol{D}$ and $\boldsymbol{C}$: We can set $\mathcal{R}(\boldsymbol{D})=\frac{1}{2}\|\phi(\boldsymbol{D})\|_F^2$ and $\mathcal{R}(\boldsymbol{C})=\frac{1}{2}\|\boldsymbol{C}\|_F^2$. We have $\mathcal{R}(\boldsymbol{D})=\frac{1}{2}\text{Tr}\left(\phi(\boldsymbol{D})^\top\phi(\boldsymbol{D})\right)=\frac{1}{2}\text{Tr}\left(\mathcal{K}(\boldsymbol{D},\boldsymbol{D})\right)$. In this case, we often require that d is equal to (or a little bit larger than) the rank or approximate rank of $\phi(\boldsymbol{X})$. Otherwise, the recovery error will be large.

Lemma 3. *Let $\boldsymbol{X}\in\mathbb{R}^{m\times n}$, $\boldsymbol{D}\in\mathbb{R}^{m\times d}$, and $\boldsymbol{C}\in\mathbb{R}^{d\times n}$. For any $\phi:\mathbb{R}^m\to\mathbb{R}^l$, suppose $d\geq\text{rank}(\phi(\boldsymbol{X}))$, then*

$$\min_{\phi(\boldsymbol{D})\boldsymbol{C}=\phi(\boldsymbol{X})}\frac{1}{2}\|\phi(\boldsymbol{D})\|_F^2+\frac{1}{2}\|\boldsymbol{C}\|_F^2\geq\|\phi(\boldsymbol{X})\|_*.$$

Lemma 3 shows using factorization we can minimize an upper bound of $\|\phi(\boldsymbol{X})\|_*$, which may eventually reduce the value of $\|\phi(\boldsymbol{X})\|_*$. The following corollary implies that solving problem (14) may find a low-nuclear-norm matrix $\phi(\boldsymbol{D})\boldsymbol{C}$ to approximate $\phi(\hat{\boldsymbol{X}}-\boldsymbol{E})$.

Corollary 2. *For any $\phi:\mathbb{R}^m\to\mathbb{R}^l$,*

$$\frac{\lambda_D}{2}\|\phi(\boldsymbol{D})\|_F^2+\frac{\lambda_C}{2}\|\boldsymbol{C}\|_F^2\geq\sqrt{\lambda_C\lambda_D}\|\phi(\boldsymbol{D})\boldsymbol{C}\|_*.$$

Nevertheless, we may not achieve the equality in Lemma 3 and Corollary 2 because of the presence of ϕ. Notice that with RBF kernel, $\text{Tr}\big(\mathcal{K}(\boldsymbol{D},\boldsymbol{D})\big)\equiv d$, which means $\lambda_D\mathcal{R}(\boldsymbol{D})$ has no effect on the minimization and can be discarded; thus the number of penalty parameters is reduced. The following lemma explains the connection between $\|\boldsymbol{C}\|_F^2$ and $\|\phi(\boldsymbol{D})\boldsymbol{C}\|_*$ when using RBF kernel.

Lemma 4. *Suppose ϕ is induced by RBF kernel. Then for any $\boldsymbol{D}\in\mathbb{R}^{m\times d}$ and $\boldsymbol{C}\in\mathbb{R}^{d\times n}$,*

$$\|\boldsymbol{C}\|_F\geq\|\phi(\boldsymbol{D})\boldsymbol{C}\|_*/\sqrt{d}.$$

4) Non-smooth penalty on $\boldsymbol{D}$ and $\boldsymbol{C}$: For example, we set $\mathcal{R}(\boldsymbol{D})=\|\phi(\boldsymbol{D})\|_*$, where $\|\cdot\|_*$ denotes the matrix nuclear norm. We have $\mathcal{R}(\boldsymbol{D})=\text{Tr}\big(\mathcal{K}(\boldsymbol{D},\boldsymbol{D})^{1/2}\big)$. Such penalty on $\boldsymbol{D}$ will encourage $\phi(\boldsymbol{D})$ to be low-rank. Similarly, we can penalize $\boldsymbol{C}$ to be low-rank by $\mathcal{R}(\boldsymbol{C})=\|\boldsymbol{C}\|_*$. Moreover, we may set $\mathcal{R}(\boldsymbol{C})=\|\boldsymbol{C}\|_1$, which encourages $\boldsymbol{C}$ to be sparse. The motivation is the same as dictionary learning: each column of $\phi(\boldsymbol{X})$ can be represented by a linear combination of a few columns of $\phi(\boldsymbol{D})$. Nevertheless, the nonsmooth $\mathcal{R}(\boldsymbol{D})$ and $\mathcal{R}(\boldsymbol{C})$ will increase the difficulty of optimization.

In the remaining of this paper, we will focus on Gaussian RBF kernel because of the following reasons. First, we only need to determine one parameter σ in Gaussian RBF kernel, compared to two parameters c and q in polynomial kernel. Second, Gaussian RBF kernel is easier to optimize and the parameter σ controls the weights of low-order features and high-order features effectively. In addition, as mentioned above, when using Gaussian RBF kernel and $\mathcal{R}(\boldsymbol{D})=\frac{1}{2}\|\phi(\boldsymbol{X})\|_F^2$, we do not need the parameter λ_D.

IV. Optimization for RNLMF

Problem (14) is nonconvex and nonsmooth and has three blocks of variables. We propose to initialize $\boldsymbol{D}$ randomly and initialize $\boldsymbol{E}$ with zeros, then update $\boldsymbol{C}$, $\boldsymbol{D}$, and $\boldsymbol{E}$ alternately. In the numerical results, we show that the alternating minimization always provide satisfactory denoising performance, provided that parameters such as λ_E are properly determined.

A. Update $\boldsymbol{C}$ by closed-form solution or proximal gradient method

At iteration t, suppose we have got $\boldsymbol{D}_{t-1}$ and $\boldsymbol{E}_{t-1}$. Let

$$\begin{aligned}\mathcal{L}(\boldsymbol{C})=&-\text{Tr}\left(\boldsymbol{C}^\top\mathcal{K}(\boldsymbol{D}_{t-1},\hat{\boldsymbol{X}}-\boldsymbol{E}_{t-1})\right)\\&+\frac{1}{2}\text{Tr}\Big(\boldsymbol{C}^\top\mathcal{K}(\boldsymbol{D}_{t-1},\boldsymbol{D}_{t-1})\boldsymbol{C}\Big).\end{aligned}$$

We aim to solve

$$\underset{\boldsymbol{C}}{\text{minimize}}\quad\mathcal{L}(\boldsymbol{C})+\lambda_C\mathcal{R}(\boldsymbol{C}).\tag{15}$$

When $\mathcal{R}(\boldsymbol{C})=\frac{1}{2}\|C\|_F^2$, by letting $\nabla_{\boldsymbol{C}}[\mathcal{L}(\boldsymbol{C})+\lambda_C\mathcal{R}(\boldsymbol{C})]=0$, we update $\boldsymbol{C}$ to the solution of (15):

$$\boldsymbol{C}_t=(\mathcal{K}(\boldsymbol{D}_{t-1},\boldsymbol{D}_{t-1})+\lambda_C\boldsymbol{I}_d)^{-1}\mathcal{K}(\boldsymbol{D}_{t-1},\hat{\boldsymbol{X}}-\boldsymbol{E}_{t-1}),\tag{16}$$

where $\boldsymbol{I}_d\in\mathbb{R}^{d\times d}$ is an identity matrix. The problem is actually closely related to the kernel ridge regression, in which the feature map is performed only on the regressors $\boldsymbol{D}$ and $\phi(\hat{\boldsymbol{X}}-\boldsymbol{E})$ is replaced by the dependent variables.

When $\mathcal{R}(\boldsymbol{C})=\|C\|_1$ or $\|C\|_*$, (15) has no closed-form solution. We use first order approximation to find a majorizer of $\mathcal{L}(\boldsymbol{C})$ at $\boldsymbol{C}_{t-1}$

$$\mathcal{L}(\boldsymbol{C})\leq\mathcal{L}(\boldsymbol{C}_{t-1})+\langle\nabla_{\boldsymbol{C}}\mathcal{L}(\boldsymbol{C}_{t-1}),\boldsymbol{C}-\boldsymbol{C}_{t-1}\rangle+\frac{\tau_C^t}{2}\|\boldsymbol{C}-\boldsymbol{C}_{t-1}\|_F^2$$

and then solve

$$\underset{\boldsymbol{C}}{\text{minimize}}\ \frac{\tau_C^t}{2}\|\boldsymbol{C}-\boldsymbol{C}_{t-1}+\nabla_{\boldsymbol{C}}\mathcal{L}(\boldsymbol{C}_{t-1})/\tau_C^t\|_F^2+\lambda_C\mathcal{R}(\boldsymbol{C}),\tag{17}$$

where $\nabla_{\boldsymbol{C}}\mathcal{L}(\boldsymbol{C}_{t-1})=-\mathcal{K}(\boldsymbol{D}_{t-1},\hat{\boldsymbol{X}}-\boldsymbol{E}_{t-1})+\mathcal{K}(\boldsymbol{D}_{t-1},\boldsymbol{D}_{t-1})\boldsymbol{C}_{t-1}$. Here we need $\tau_C^t>\|\mathcal{K}(\boldsymbol{D}_{t-1},\boldsymbol{D}_{t-1})\|_2$ to ensure that (17) is non-expansive. Consequently, the closed-form solution of (17) as well as

(16) are shown in Table I. In the table, Θ denotes the soft-thresholding operator defined by

$$\Theta_u(v) = \frac{|v|}{v}\max(|v| - u, 0).$$

In addition, Ψ denotes the singular value thresholding operator [51] defined by

$$\Psi_u(\boldsymbol{M}) = \boldsymbol{U}\Theta_u(\boldsymbol{S})\boldsymbol{V}^\top,$$

where $\boldsymbol{U}$, $\boldsymbol{S}$, and $\boldsymbol{V}$ are given by the SVD $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top$.

TABLE I: Solution for $\boldsymbol{C}_t$ with respect to different $\mathcal{R}(\boldsymbol{C})$

$\mathcal{R}(\boldsymbol{C})$	$\boldsymbol{C}_t$
$\frac{1}{2}\Vert\boldsymbol{C}\Vert_F^2$	$(\mathcal{K}(\boldsymbol{D}_{t-1}, \boldsymbol{D}_{t-1}) + \lambda_C \boldsymbol{I}_d)^{-1}\mathcal{K}(\boldsymbol{D}_{t-1}, \hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1})$
$\Vert\boldsymbol{C}\Vert_1$	$\Theta_{\lambda_C/\tau_C^t}(\boldsymbol{C}_{t-1} - \nabla_{\boldsymbol{C}}\mathcal{L}(\boldsymbol{C}_{t-1})/\tau_C^t)$
$\Vert\boldsymbol{C}\Vert_*$	$\Psi_{\lambda_C/\tau_C^t}(\boldsymbol{C}_{t-1} - \nabla_{\boldsymbol{C}}\mathcal{L}(\boldsymbol{C}_{t-1})/\tau_C^t)$

The following lemma shows that the update of $\boldsymbol{C}$ when $\mathcal{R}(\boldsymbol{C}) = \|\boldsymbol{C}\|_1$ or $\|\boldsymbol{C}\|_*$ ensures the objective function is nonascending.

Lemma 5. *Let $\boldsymbol{C}_t$ be the solution of* (17) *with $\mathcal{R}(\boldsymbol{C}) = \|\boldsymbol{C}\|_1$ or $\|\boldsymbol{C}\|_*$. Denote $L_C^t = \|\mathcal{K}(\boldsymbol{D}_{t-1}, \boldsymbol{D}_{t-1})\|_2$. Then*

$$\begin{aligned}&\mathcal{J}(\boldsymbol{D}_{t-1}, \boldsymbol{C}_t, \boldsymbol{E}_{t-1}) - \mathcal{J}(\boldsymbol{D}_{t-1}, \boldsymbol{C}_{t-1}, \boldsymbol{E}_{t-1})\\ \le& -\frac{\tau_C^t - L_C^t}{2}\|\boldsymbol{C}_t - \boldsymbol{C}_{t-1}\|_F^2,\end{aligned}$$

where $\tau_C^t > L_C^t$.

B. Update $\boldsymbol{D}$ by relaxed Newton method

Having obtained $\boldsymbol{E}_{t-1}$ and $\boldsymbol{C}_t$, we let

$$\begin{aligned}\mathcal{L}(\boldsymbol{D}) = &-\text{Tr}\left(\boldsymbol{C}_t^\top \mathcal{K}(\boldsymbol{D}, \hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1})\right)\\ &+ \frac{1}{2}\text{Tr}\left(\boldsymbol{C}_t^\top \mathcal{K}(\boldsymbol{D}, \boldsymbol{D})\boldsymbol{C}_t\right).\end{aligned}$$

Because of the presence of kernel function, the minimization of $\mathcal{L}(\boldsymbol{D})$ has no closed-form solution. The gradient[2] is

$$\begin{aligned}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}) =&\tfrac{1}{\sigma^2}((\hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1})\boldsymbol{W}_D - \boldsymbol{D}\bar{\boldsymbol{W}}_D)\\ &+ \tfrac{2}{\sigma^2}(\boldsymbol{D}\boldsymbol{Q}_D - \boldsymbol{D}\bar{\boldsymbol{Q}}_D),\end{aligned}$$

where $\boldsymbol{W}_D = -\boldsymbol{C}_t^\top \odot \mathcal{K}(\hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1}, \boldsymbol{D})$, $\boldsymbol{Q}_D = (0.5\boldsymbol{C}_t\boldsymbol{C}_t^\top) \odot \mathcal{K}(\boldsymbol{D}, \boldsymbol{D})$, $\bar{\boldsymbol{W}}_D = \text{diag}(\boldsymbol{1}_n^\top \boldsymbol{W}_D)$, and $\bar{\boldsymbol{Q}}_D = \text{diag}(\boldsymbol{1}_d^\top \boldsymbol{Q}_D)$. One straightforward approach is to update $\boldsymbol{D}$ by gradient descent with backtracking line search, which however requires evaluating $\mathcal{L}(\boldsymbol{D})$ for multiple times and hence increases the computational cost. In addition, one may consider using second-order information to accelerate the optimization. Note that $\frac{\partial[\boldsymbol{W}_D]_{ij}}{\partial[\boldsymbol{D}]_{:j}} = [\boldsymbol{C}_t^\top]_{ij}[\mathcal{K}(\hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1}, \boldsymbol{D})]_{ij}(\hat{\boldsymbol{x}}_i - [\boldsymbol{E}_{t-1}]_{:j})/\sigma^2$. Thus, when σ is large, we regard $\boldsymbol{W}_D$ (also $\bar{\boldsymbol{W}}_D$, $\boldsymbol{Q}_D$, and $\bar{\boldsymbol{Q}}_D$) as a constant independent of $\mathbf{D}$ and consider the derivative of $\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D})$ at $\mathbf{D}_{t-1}$. Consequently, we define

$$\boldsymbol{H}_{t-1} = \tfrac{1}{\sigma^2}(-\bar{\boldsymbol{W}}_{D_{t-1}} + 2\boldsymbol{Q}_{D_{t-1}} - 2\bar{\boldsymbol{Q}}_{D_{t-1}}),$$

where $\mu \ge 0$ is large enough such that $\boldsymbol{H}_{t-1} + \mu\boldsymbol{I}$ is positive definite.Then we update $\boldsymbol{D}$ by a relaxed Newton step

$$\boldsymbol{D}_t = \boldsymbol{D}_{t-1} - \tfrac{1}{\tau_D^t}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})(\boldsymbol{H}_{t-1} + \mu\boldsymbol{I})^{-1}, \qquad (18)$$

where $\tau_D^t \ge 1$ controls the step size. The effectiveness of (18) is justified by the following lemma.

Lemma 6. *Suppose $\boldsymbol{D}_t$ is given by* (18) *and τ_D^t and μ are sufficiently large. Then*

$$\begin{aligned}&\mathcal{J}(\boldsymbol{D}_t, \boldsymbol{C}_t, \boldsymbol{E}_{t-1}) - \mathcal{J}(\boldsymbol{D}_{t-1}, \boldsymbol{C}_t, \boldsymbol{E}_{t-1})\\ \le& -\tfrac{1}{2\tau_D^t}\text{Tr}\left(\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})(\boldsymbol{H}_{t-1} + \mu\boldsymbol{I})^{-1}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})^\top\right) \le 0.\end{aligned}$$

Remark 3. *Empirically, in our experiments, we found $\boldsymbol{H}$ was always positive definite. Hence we set $\mu = 0$ in all experiments. In addition, $\tau_D = 1$ works well in practice.*

We can also incorporate momentum into the update of $\boldsymbol{D}$:

$$\boldsymbol{\Delta}_t = \eta\boldsymbol{\Delta}_{t-1} + \tfrac{1}{\tau_D^t}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})(\boldsymbol{H}_{t-1} + \mu\boldsymbol{I})^{-1}, \qquad (19)$$

where $0 < \eta < 1$; then

$$\boldsymbol{D}_t = \boldsymbol{D}_{t-1} - \boldsymbol{\Delta}_t. \qquad (20)$$

The following corollary shows that when η is sufficiently small, the objective function is non-increasing.

Corollary 3. *Suppose $\boldsymbol{D}_t$ is given by* (20) *and τ_D^t is sufficiently large. Then*

$$\begin{aligned}&\mathcal{J}(\boldsymbol{D}_t, \boldsymbol{C}_t, \boldsymbol{E}_{t-1}) - \mathcal{J}(\boldsymbol{D}_{t-1}, \boldsymbol{C}_t, \boldsymbol{E}_{t-1})\\ \le& -\frac{1}{2\tau_D^t}\text{Tr}\left(\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})(\boldsymbol{H}_{t-1} + \mu\boldsymbol{I})^{-1}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})^\top\right)\\ &+ \frac{\eta^2\tau_D^t}{2}\text{Tr}\left(\boldsymbol{\Delta}_{t-1}(\boldsymbol{H}_{t-1} + \mu\boldsymbol{I})\boldsymbol{\Delta}_{t-1}^\top\right).\end{aligned} \qquad (21)$$

When d is large, to avoid the high computational cost of the inverse of $\boldsymbol{H}_{t-1}$, we suggest replacing (20) with

$$\boldsymbol{D}_t = \boldsymbol{D}_{t-1} - \bar{\boldsymbol{\Delta}}_t, \qquad (22)$$

where $\bar{\boldsymbol{\Delta}}_t = \eta\bar{\boldsymbol{\Delta}}_{t-1} + \frac{1}{\tau_D^t}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})/\|\boldsymbol{H}_{t-1}\|_2$.

Algorithm 1 Optimization for RNLMF

```
Input: $\hat{\boldsymbol{X}}$, $d$, $\lambda_C$, $\lambda_E$, $\sigma$, $\eta$, $t_{\text{iter}}$.
Initialize: $\boldsymbol{E} = \boldsymbol{0}$, $\boldsymbol{C} = \boldsymbol{0}$, $\boldsymbol{D} \sim \mathcal{N}(0, 1)$, $\boldsymbol{\Delta} = \boldsymbol{0}$, $t = 0$.
repeat
    $t \leftarrow t + 1$.
    Update $\boldsymbol{C}$ using Table I.
    Update $\boldsymbol{\Delta}$ using (19)
    Update $\boldsymbol{D}$ using (20).
    Update $\boldsymbol{E}$ using Table II.
until converged or $t = t_{\text{iter}}$
Output: $\boldsymbol{X} = \hat{\boldsymbol{X}} - \boldsymbol{E}$, $\boldsymbol{D}$, $\boldsymbol{C}$.
```

[2] Use the chain rule $\frac{\partial\mathcal{L}}{\partial\boldsymbol{Z}} = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\frac{\partial\mathcal{L}}{\partial\mathcal{K}_{ij}}\frac{\partial\mathcal{K}_{ij}}{\partial\boldsymbol{Z}}$, where $\mathcal{K} \in \mathbb{R}^{n_1 \times n_2}$ is the kernel matrix computed from $\boldsymbol{Z}$.

C. Update $\boldsymbol{E}$ by proximal gradient method

Having got $\boldsymbol{C}_t$ and $\boldsymbol{D}_t$, we let

$$\begin{aligned}\mathcal{L}(\boldsymbol{E}) =&\frac{1}{2}\text{Tr}\left(\mathcal{K}(\hat{\boldsymbol{X}}-\boldsymbol{E},\hat{\boldsymbol{X}}-\boldsymbol{E})\right)-\text{Tr}\left(\boldsymbol{C}_t^\top\mathcal{K}(\boldsymbol{D}_t,\hat{\boldsymbol{X}}-\boldsymbol{E})\right)\\ =&\frac{n}{2}-\text{Tr}\left(\boldsymbol{C}_t^\top\mathcal{K}(\boldsymbol{D}_t,\hat{\boldsymbol{X}}-\boldsymbol{E})\right)\end{aligned}$$

Then we need to solve

$$\underset{\boldsymbol{E}}{\text{minimize}}\quad \mathcal{L}(\boldsymbol{E})+\lambda_E\mathcal{R}(\boldsymbol{E}), \tag{23}$$

which, however, has no closed-form solution. Compute the gradient of $\mathcal{L}(\boldsymbol{E})$ as

$$\nabla_{\boldsymbol{E}}\mathcal{L}(\boldsymbol{E})=\tfrac{1}{\sigma^2}\big((\hat{\boldsymbol{X}}-\boldsymbol{E})\bar{\boldsymbol{G}}_E-\boldsymbol{D}_t\boldsymbol{G}_E\big), \tag{24}$$

where $\boldsymbol{G}_E=-\boldsymbol{C}_t\odot\mathcal{K}(\boldsymbol{D}_t,\hat{\boldsymbol{X}}-\boldsymbol{E})$ and $\bar{\boldsymbol{G}}_E=\text{diag}(\boldsymbol{1}_d^\top\boldsymbol{G}_E)$. Suppose the Lipschitz constant of $\nabla_{\boldsymbol{E}}\mathcal{L}(\boldsymbol{E})$ at iteration t is L_E^t. Let $\tau_E^t>L_E^t$ and we have

$$\begin{aligned}\mathcal{L}(\boldsymbol{E})\leq&\mathcal{L}(\boldsymbol{E}_{t-1})+\langle\nabla_{\boldsymbol{E}}\mathcal{L}(\boldsymbol{E}_{t-1}),\boldsymbol{E}-\boldsymbol{E}_{t-1}\rangle\\ &+\frac{\tau_E^t}{2}\|\boldsymbol{E}-\boldsymbol{E}_{t-1}\|_F^2.\end{aligned}$$

Thus we update $\boldsymbol{E}$ by solving

$$\underset{\boldsymbol{E}}{\text{minimize}}\ \frac{\tau_E^t}{2}\|\boldsymbol{E}-\boldsymbol{E}_{t-1}+\nabla_{\boldsymbol{E}}\mathcal{L}(\boldsymbol{E}_{t-1})/\tau_E^t\|_F^2+\lambda_E\mathcal{R}(\boldsymbol{E}). \tag{25}$$

The closed-form solutions of (25) with different $\mathcal{R}(\boldsymbol{E})$ are shown in Table II. In the table, $\Upsilon_u(\cdot)$ is the column-wise soft-thresholding operator [52] defined as

$$\Upsilon_u(\boldsymbol{v})=\begin{cases}\frac{(\|\boldsymbol{v}\|-u)\boldsymbol{v}}{\|\boldsymbol{v}\|}, & \text{if } \|\boldsymbol{v}\|>u;\\ \boldsymbol{0}, & \text{otherwise.}\end{cases}$$

TABLE II: Solution of (25) with respect to different $\mathcal{R}(\boldsymbol{E})$

$\mathcal{R}(\boldsymbol{E})$	$\boldsymbol{E}_t$
$\frac{1}{2}\|\boldsymbol{E}\|_F^2$	$\dfrac{\tau_E^t}{\tau_E^t+\lambda_E}(\boldsymbol{E}_{t-1}-\nabla_{\boldsymbol{E}}\mathcal{L}(\boldsymbol{E}_{t-1})/\tau_E^t)$
$\|\boldsymbol{E}\|_1$	$\Theta_{\lambda_E/\tau_E^t}(\boldsymbol{E}_{t-1}-\nabla_{\boldsymbol{E}}\mathcal{L}(\boldsymbol{E}_{t-1})/\tau_E^t)$
$\|\boldsymbol{E}\|_{2,1}$	$\Upsilon_{\lambda_E/\tau_E^t}(\boldsymbol{E}_{t-1}-\nabla_{\boldsymbol{E}}\mathcal{L}(\boldsymbol{E}_{t-1})/\tau_E^t)$

To determine τ_E^t, we estimate L_E^t as $\hat{L}_E^t=\xi\|\bar{\boldsymbol{G}}_{E_{t-1}}\|_2/\sigma^2=\xi\|\boldsymbol{1}_d^\top\boldsymbol{G}_{E_{t-1}}\|_\infty/\sigma^2$ where ξ is a sufficiently large constant. Equivalently, we set $\tau_E^t=\xi\|\boldsymbol{1}_d^\top\boldsymbol{G}_{E_{t-1}}\|_\infty/\sigma^2$ where ξ is a sufficiently large constant. The following lemma indicates that updating $\boldsymbol{E}$ by Table II is nonexpansive.

Lemma 7. *Let $\boldsymbol{E}_t$ be the solution of* (25)*, where $\tau_E^t=\xi\|\boldsymbol{1}_d^\top\boldsymbol{G}_{E_{t-1}}\|_\infty/\sigma^2$ and ξ is sufficiently large. Then*

$$\begin{aligned}&\mathcal{J}(\boldsymbol{D}_t,\boldsymbol{C}_t,\boldsymbol{E}_t)-\mathcal{J}(\boldsymbol{D}_t,\boldsymbol{C}_t,\boldsymbol{E}_{t-1})\\ \leq&-\frac{\tau_E^t-L_E^t}{2}\|\boldsymbol{E}_t-\boldsymbol{E}_{t-1}\|_F^2\ \leq 0.\end{aligned}$$

Remark 4. *Empirically, we found that Lemma 7 often holds when $\xi=1$. It means $\hat{L}_E^t$ well approximates the Lipschitz constant of $\nabla_{\boldsymbol{E}}\mathcal{L}(\boldsymbol{E})$ at iteration t.*

D. The overall algorithm

The optimization for RNLMF is summarized in Algorithm 1. The hyper-parameter σ controls the smoothness of Gaussian RBF kernel and provides us flexibility to handle nonlinearity of different levels. We set $\sigma=cn^{-2}\sum_{ij}\|\hat{\boldsymbol{x}}_i-\hat{\boldsymbol{x}}_j\|$, where c is a constant such as 0.5, 1, or 3. When the data have strong nonlinearity, we use a smaller σ (smaller c); otherwise, we use a larger σ (larger c).

In Algorithm 1, when $\mathcal{R}(\boldsymbol{C})=\|\boldsymbol{C}\|_*$ or $\|\boldsymbol{C}\|_1$, the convergence with $\eta=0$ is guaranteed by Theorem 1, although a nonzero η (e.g., 0.5) works better in practice. When $\mathcal{R}(\boldsymbol{C})=\frac{1}{2}\|\boldsymbol{C}\|_F^2$, the convergence is similar and the proof is omitted for simplicity. Proving the algorithm converges to a critical point is out of the scope of this paper, though it may be accomplished by following the methods used in [53].

Theorem 1. *Let $\{\boldsymbol{C}_t,\boldsymbol{D}_t,\boldsymbol{E}_t\}$ be the sequence generated by Algorithm 1 with $\eta=0$. Suppose τ_D^t, μ, and ξ are sufficiently large. Then*

$$\begin{aligned}&\lim_{t\to\infty}\mathcal{J}(\boldsymbol{D}_t,\boldsymbol{C}_t,\boldsymbol{E}_t)-\mathcal{J}(\boldsymbol{D}_{t-1},\boldsymbol{C}_{t-1},\boldsymbol{E}_{t-1})=0,\\ &\lim_{t\to\infty}\|\boldsymbol{D}_t-\boldsymbol{D}_{t-1}\|_F=0,\\ &\lim_{t\to\infty}\|\boldsymbol{C}_t-\boldsymbol{C}_{t-1}\|_F=0,\\ &\lim_{t\to\infty}\|\boldsymbol{E}_t-\boldsymbol{E}_{t-1}\|_F=0.\end{aligned}$$

In Section IV-A, when $\mathcal{R}(\boldsymbol{C})=\|\boldsymbol{C}\|_1$ or $\|\boldsymbol{C}\|_*$, the subproblem of $\boldsymbol{C}$ has no closed-form solution, which may slow down convergence. We found that using $\|\boldsymbol{C}\|_F^2$ in the early iterations and then switching to $\|\boldsymbol{C}\|_1$ or $\|\boldsymbol{C}\|_*$ can speed up convergence. Nevertheless, our numerical results in Section VIII showed that $\|\boldsymbol{C}\|_F^2$ outperformed $\|\boldsymbol{C}\|_1$ and $\|\boldsymbol{C}\|_*$.

V. Time and space complexity

In Algorithm 1, we need to store $\hat{\boldsymbol{X}}\in\mathbb{R}^{m\times n}$, $\boldsymbol{E}\in\mathbb{R}^{m\times n}$, $\boldsymbol{D}\in\mathbb{R}^{m\times d}$, $\boldsymbol{C}\in\mathbb{R}^{d\times n}$, $\mathcal{K}(\hat{\boldsymbol{X}}-\boldsymbol{E},\boldsymbol{D})\in\mathbb{R}^{n\times d}$, and $\mathcal{K}(\boldsymbol{D},\boldsymbol{D})\in\mathbb{R}^{d\times d}$. Then the space complexity of RNLMF is as $O(mn+md+dn+d^2)$ or $O(mn+dn)$ equivalently because of $m,d<n$. In each iteration of Algorithm 1, the main computational cost is from the computation of $\mathcal{K}(\hat{\boldsymbol{X}}-\boldsymbol{E},\boldsymbol{D})$, the inverse of a $d\times d$ matrix (or the SVD of a $d\times n$ matrix) in Table I, the inverse of a $d\times d$ matrix in updating $\boldsymbol{D}$, and the related multiplications in (16), (18), and (24). Then the time complexity in each iteration of RNLMF is $O(d^3+d^2n+d^2m+dmn)$.

The time and space complexities of RPCA [4], LRR [10], NLRR [47], SSC [9], RDL (problem (6)), and RNLMF are compared in Table III, where truncated (top-r) SVD is considered in LRR and $\mathcal{R}(\boldsymbol{C})=\|\boldsymbol{C}\|_1$ or $\|\boldsymbol{C}\|_F^2$ are considered for RNLMF. We see that when n is large, SSC and LRR have high time and space complexities. The computational costs of RNLMF and RDL are similar, though in real applications the d in RNLMF should be larger than that in RDL. But RNLMF is able to handle data generated by more complex models.

VI. Out-Of-Sample Extension of RNLMF

It is worth mentioning that in an online fashion, the dictionary $\boldsymbol{D}$ given by Algorithm 1 can be used to denoise a new

TABLE III: Time and space complexities

	Time complexity	Space complexity
RPCA	$O(m^2n)$	$O(mn)$
LRR	$O(mn^2+rn^2)$	$O(mn+n^2)$
NLRR	$O(m^2n+rmn)$	$O(mn+rn)$
SSC	$O(mn^2)$	$O(mn+n^2)$
RDL	$O(d^2n+dmn)$	$O(mn+dn)$
RNLMF	$O(d^2n+dmn)$	$O(mn+dn)$

*Assume $r<m<n$, $r<d<n$, and $d^2 \leq mn$.

data matrix $\hat{\boldsymbol{X}}'$ generated from the same model as $\hat{\boldsymbol{X}}$. The approach is shown in Algorithm 2.

Algorithm 2 Out-of-sample extension of RNLMF

```
Input: $\hat{X}'$, $D$ (given by Algorithm 1), $t_{\text{iter}}$.
Initialize: $E' = 0$, $C' = 0$, $t = 0$.
repeat
    $t \leftarrow t + 1$.
    Update $C'$ using Table I.
    Update $E'$ using Table II.
until converged or $t = t_{\text{iter}}$
Output: $X' = \hat{X}' - E'$, $C'$.
```

VII. Subspace clustering by RNLMF

RNLMF can be regarded as a robust nonlinear feature extraction method, thus the feature matrix $\boldsymbol{C}$ can be used for clustering. One may perform SSC [9] or LRR [10] on $\boldsymbol{C}$, which, however, is not efficient. We propose to compute an affinity matrix by solving the following least squares problem

$$\underset{\boldsymbol{A}}{\text{minimize}}\ \tfrac{1}{2}\|\boldsymbol{C}-\boldsymbol{C}\boldsymbol{A}\|_F^2+\tfrac{\gamma}{2}\|\boldsymbol{A}\|_F^2, \tag{26}$$

where γ is a penalty parameter. The solution is $\boldsymbol{A} = (\boldsymbol{C}^\top\boldsymbol{C}+\gamma\boldsymbol{I})^{-1}\boldsymbol{C}^\top\boldsymbol{C}$. Note that least squares regression is also effective in subspace segmentation [54]. Then let $\boldsymbol{A} \leftarrow |\boldsymbol{A}|$, set $\text{diag}(\boldsymbol{A}) = \boldsymbol{0}$, and keep the largest κ entries of each column of $\boldsymbol{A}$ and discard the other entries. A normalization is performed on each column of $\boldsymbol{A}$: $\boldsymbol{a}_j = \boldsymbol{a}_j/\max(\boldsymbol{a}_j)$, $j = 1, 2, \ldots, n$. To ensure a symmetric affinity matrix, we set $\boldsymbol{A} \leftarrow (\boldsymbol{A}+\boldsymbol{A}^\top)/2$. Finally, spectral clustering is performed on $\boldsymbol{A}$. The procedures are summarized in Algorithm 3. The role of Procedures 3 $\sim$ 5 is similar to the post-processing in SSC and LRR, making the affinity matrix more compact. It is worth mentioning that using the Woodbury identity, line 2 in Algorithm 3 is equivalent to $\boldsymbol{A} = |\gamma^{-1}\boldsymbol{C}^\top(\boldsymbol{I}+\gamma^{-1}\boldsymbol{C}\boldsymbol{C}^\top)^{-1}\boldsymbol{C}|$, which reduced the computational cost.

When the number of data points is very large (e.g. $n > 10^5$), we cannot use Algorithm 3 to cluster the whole dataset because the high space cost of $\boldsymbol{A}$. Recently a few large-scale subspace clustering methods have been proposed [55], [12] and they often take advantage of exemplars or landmark points selection to cluster large-scale datasets but cannot effectively handle sparse noise. Similar ideas may apply to our RNLMF, which however is out of the scope of this paper.

Algorithm 3 Subspace clustering by RNLMF

```
Input: $\hat{X}$, $k$, $\kappa$, $\gamma$.
1: Compute $C$ using Algorithm 1.
2: $A = |(C^\top C + \gamma I)^{-1} C^\top C|$ and set $\text{diag}(A) = 0$.
3: Keep only the largest $\kappa$ entries of each column of $A$.
4: For $j = 1, 2, \ldots, n$, $a_j \leftarrow a_j / \max(a_j)$.
5: $A \leftarrow (A + A^\top)/2$.
6: Perform spectral clustering on $A$ with cluster number $k$.
Output: $k$ clusters of $\hat{X}$.
```

VIII. Experiments on Synthetic Data

We generate synthetic data by

$$\boldsymbol{x} = f(\boldsymbol{z}), f \in \{F^1, F^2, \ldots, F^k\},$$

where each $F^j : \mathbb{R}^3 \to \mathbb{R}^{30}$, $1 \leq j \leq k$ is an order-3 polynomial mapping and $\boldsymbol{z} = [z_1, z_2, z_3]^\top \sim \mathcal{U}(-1, 1)$. The model can be reformulated as

$$\boldsymbol{x} = \boldsymbol{P}\tilde{\boldsymbol{z}}, \boldsymbol{P} \in \{\boldsymbol{\Gamma}^1, \boldsymbol{\Gamma}^2, \ldots, \boldsymbol{\Gamma}^k\},$$

where $\boldsymbol{\Gamma}^j \in \mathbb{R}^{30\times 19} \sim \mathcal{N}(0,1)$ for $1 \leq j \leq k$, and $\tilde{\boldsymbol{z}} \in \mathbb{R}^{19}$ consists of order-1, 2 and 3 polynomial features of $\boldsymbol{z}$. For each fixed $\boldsymbol{\Gamma}^j$, we generate 300 random samples of $\boldsymbol{x}$. Then we obtain a matrix $\boldsymbol{X} \in \mathbb{R}^{30\times 300k}$, which is full-rank when $k \geq 2$. We then add sparse noise to $\boldsymbol{X}$, i.e. $\hat{\boldsymbol{X}} = \boldsymbol{X} + \boldsymbol{E}$, where $\frac{1}{9000k}\sum_{ij}\mathbb{1}(E_{ij} \neq 0) = \rho$ and the nonzero entries of $\boldsymbol{E}$ are drawn from $\mathcal{N}(0, \sigma_e^2)$. The locations of nonzero entries are chosen uniformly at random by sampling without replacement. Denote the standard deviation of the entries of $\boldsymbol{X}$ by σ_x.

The denoising performance is evaluated by the normalized root-mean-square-error:

$$\text{RMSE} := \|\boldsymbol{X} - \check{\boldsymbol{X}}\|_F / \|\boldsymbol{X}\|_F,$$

where $\check{\boldsymbol{X}}$ denotes the recovered matrix. All results we report in this paper are the average of 20 repeated trials. In RNLMF, we set $d = 2mk$, choose λ_C from $[1, 5, 10]/10^3$, and choose λ_E from $[0.3, 0.5, 1, 2]/10^3$; we set $\sigma = n^{-2}\sum_{ij}\|\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_j\|$. Such parameter settings are utilized throughout this paper, unless stated otherwise.

A. Choice of $\mathcal{R}(\boldsymbol{C})$

Figure 1(a) shows the RMSE of RNLMF with different penalty operators of $\boldsymbol{C}$ when $k = 3$ and varying ρ, for which we have sufficiently tuned λ_C and λ_E separately for each regularizer $\mathcal{R}(\boldsymbol{C})$. We see that $\|\boldsymbol{C}\|_F^2$ always outperform $\|\boldsymbol{C}\|_1$ and $\|\boldsymbol{C}\|_*$. In Figure 1(b), where $k = 3$ and $\rho = 0.3$, $\|\boldsymbol{C}\|_F^2$ outperformed $\|\boldsymbol{C}\|_1$ and $\|\boldsymbol{C}\|_*$, in all choices of d (the number of columns of $\boldsymbol{D}$). In addition, RNLMF is relatively not sensitive to d provided that d is large enough (e.g. $d \geq 135$). The advantage of $\|\boldsymbol{C}\|_F^2$ over $\|\boldsymbol{C}\|_1$ and $\|\boldsymbol{C}\|_*$ may result from: (1) $\|\boldsymbol{C}\|_F^2$ leads to a closed-form solution for updating $\boldsymbol{C}$, which enables the optimization of RNLMF to obtain a better stationary point; (b) $\phi(\boldsymbol{X})$ is low-rank such that the denoising problem does not benefit from enforcing $\boldsymbol{C}$ to be sparse; (c) enforcing $\boldsymbol{C}$ to be low-rank reduces the compactness of $\phi(\boldsymbol{D})$. In the remaining of this paper, we only use $\mathcal{R}(\boldsymbol{C}) = \|\boldsymbol{C}\|_F^2$ in RNLMF.

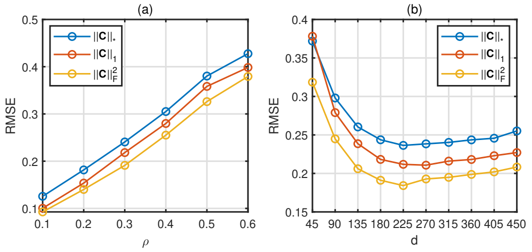

Fig. 1: RNLMF with different $\mathcal{R}(\boldsymbol{C})$: (a) $k=3$, $\sigma_e/\sigma_x=1$, $d=2mk$, and different ρ; (b) $k=3$, $\sigma_e/\sigma_x=1$, $\rho=0.3$, and different d.

B. Influence of hyper-parameters in RNLMF

Figure 2 shows the influence of η in the optimization of RNLMF with $\mathcal{R}(\boldsymbol{C})=\|\boldsymbol{C}\|_F^2$, in the case of $k=3$ and $\rho=0.3$. We see that when η increases, the objective function converges faster. But when η is too large, the algorithm may diverge.

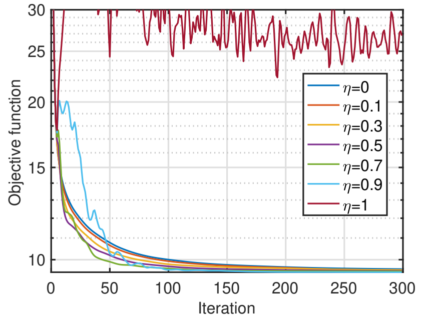

Fig. 2: Influence of η in the optimization of RNLMF

Figure 3(a) shows the sensitivity of our method to the hyper-parameters λ_C and λ_E on synthetic data ($k=3$, ρ=0.3). We see that our RNLMF is not sensitive to λ_C and has low recovery error when $0.1\times10^{-3}\leq\lambda_E\leq0.7\times10^{-3}$. Figure 3(b) shows the influence of the hyper-parameter σ of Gaussian RBF kernel and d in RNLMF. We see that $\sigma=0.5\delta$ and 1.0δ outperformed $\sigma=1.5\delta$ and 2δ. In addition, $d=270$ is the best to $\sigma=0.5\delta$ and 1.0δ, $d=240$ is the best to $\sigma=1.5\delta$, and $d=210$ is the best to $\sigma=2\delta$. It indicates that when σ is small, the optimal d should be large.

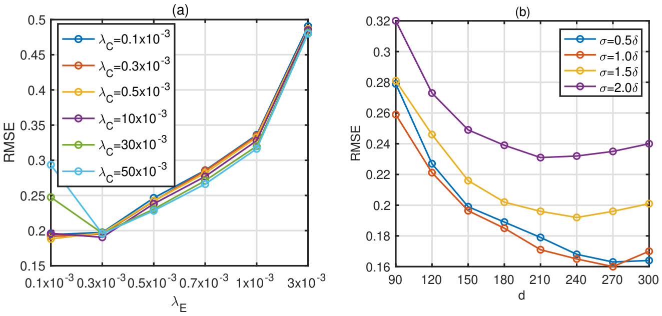

Fig. 3: Influence of λ_C, λ_E, σ, and d in RNLMF ($k=3$, $\rho=0.3$, $\delta=n^{-2}\sum_{ij}\|\hat{\boldsymbol{x}}_i-\hat{\boldsymbol{x}}_j\|$).

C. Comparison with RPCA, LRR, SSC, and RDL in denoising

Since the kernel methods [33], [34], [35], [36], [49] do not provide denoised matrix $\boldsymbol{X}$, we compare RNLMF with RPCA (problem (1), solved by ADMM), RDL (problem (6), solved by PALM [53]), LRR [10], and SSC [9]. First, we consider sparse noise and use $\mathcal{R}(\boldsymbol{E})=\|\boldsymbol{E}\|_1$ for all compared methods. In RPCA, the parameter λ is chosen from $[0.5,0.75,1,1.5,2,2.5,3]/\sqrt{n}$. In RDL, we set the number of dictionary atoms as $0.5mk$ or mk, choose λ_C from $[1,3,5,10]/10^3$, and choose λ_E from $[0.03,0.05,0.07,0.1,0.15,0.2]$. The parameters in LRR and SSC are carefully tuned to provide the best denoising performance as possible. The parameter setting in RNLMF has been stated in the beginning of Section VIII.

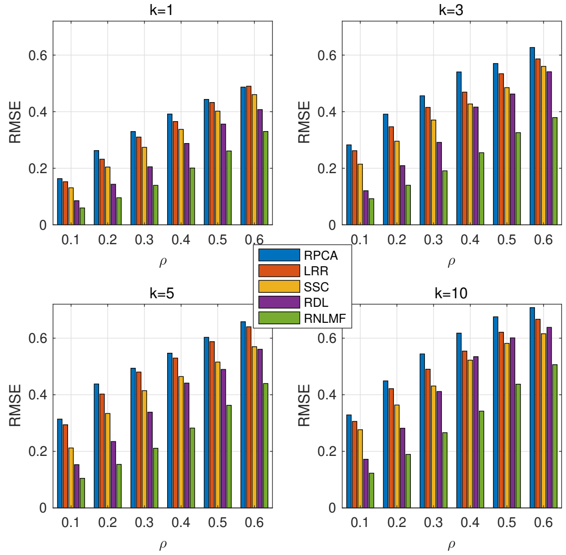

Fig. 4: RMSE on synthetic data ($\frac{\sigma_e}{\sigma_x}=1$, different k and ρ).

Figure 4 shows the recovery errors when $\sigma_e/\sigma_x=1$ and ρ and k vary. When k or ρ increase, the recovery task becomes more difficult. SSC and RDL outperformed RPCA and LRR. The reason is that sparse representation is more effective than low-rank model in handling high-rank matrices. In every case of Figure 4, the RMSE of our RNLMF is much lower than those of other methods. The improvement given by RNLMF is owing to the ability of RNLMF to handle nonlinear data and full-rank matrices, which are challenges for other methods.

We investigate the influence of the noise magnitude on the performance of five methods in the case of $k=3$ and $\rho=0.3$, shown in Figure 5. We see that RNLMF consistently outperforms other methods with different σ_e/σ_x.

To evaluate the ability of all methods to handle column-wise noise, we add independent and identically distributed noise drawn from $\mathcal{N}(0,\sigma_x^2)$ to a fraction (denoted by ρ) of columns of $\boldsymbol{X}$. Therefore, we use $\mathcal{R}(\boldsymbol{E})=\|\boldsymbol{E}\|_{2,1}$ in all compared methods. Shown in Figure 6, the RMSE of RNLMF is the lowest in every case.

IX. Experiments on Image Data

To test the performance of our method on real data, we consider the following four image datasets.

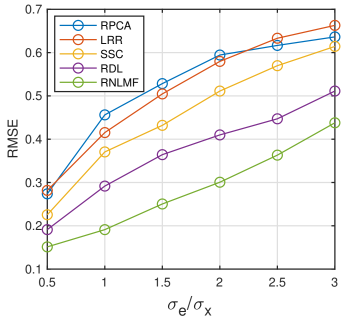

Fig. 5: RMSE on synthetic data ($k = 3$, $\rho = 0.3$).

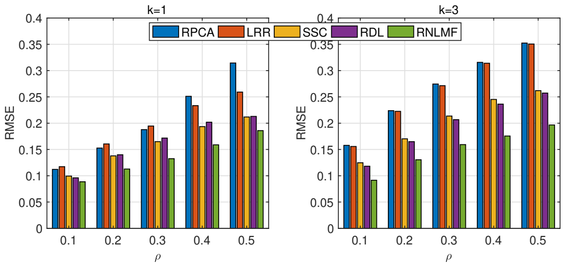

Fig. 6: RMSE on synthetic data with column-wise noise.

- COIL20[56]/COIL100[57], images of 20/100 objects. Each object has 72 images of different poses.
- Extended Yale Face database B (Yale Face for short) [58], face images of 38 subjects. Each subject has about 64 images under various illumination conditions.
- AR Face database (a subset) [59], consisting of the face images of 50 males and 50 females [17]. Each subject has 26 images with different facial expressions, illumination conditions, and occlusions.

We resize the images in AR Face to 33 × 24 and resize the images in the other three databases to 20 × 20. We consider two cases of image corruption. In the first case, we add salt-and-pepper noise of density 0.25 to 30% of the images. In the other case, for each data set, we occlude 30% of the images with a block mask of size $0.25h \times 0.25w$ and position random, where h and w are the height and length of the images. Since the two cases are sparse noise patterns, we use $\mathcal{R}(\boldsymbol{E}) = \|\boldsymbol{E}\|_1$ in RPCA, LRR, SSC, RDL, and RNLMF. For COIL20, Yale Face, AR Face, and COIL100, the d in RDL is set to 196, 196, 256, and 512 respectively, while the d in RNLMF is set to 256, 256, 512, and 768 respectively.

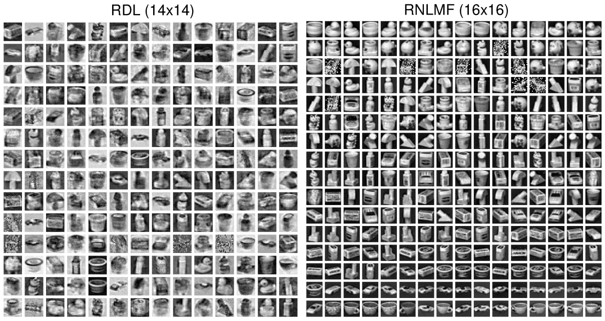

Fig. 7: The dictionaries learned from COIL20.

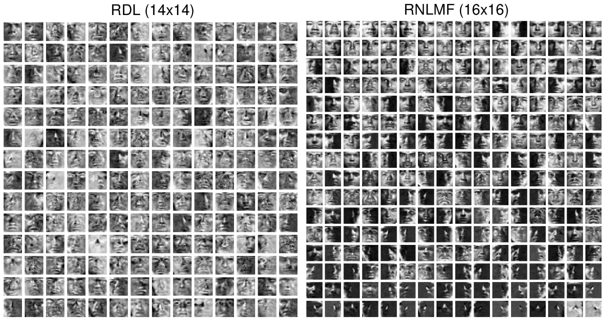

Fig. 8: The dictionaries learned from Yale Face.

Compared to Yale Face and AR Face, the nonlinearity of data structure in COIL20 and COIL100 are much higher because of the different posses of the objects. For each data set, we stack the pixels of each image as a matrix column and then form a matrix $\boldsymbol{X}$ of size $m \times n$, where $m = hw$ and n is the number of images. The metric $R := \|\boldsymbol{X}\|_*/\|\boldsymbol{X}\|_F$ can be utilized to compare the rank of the data matrices of the four data sets. For COIL20, COIL100, Yale Face, and AR Face, the values of R are 5.17, 5.03, 4.65, and 4.33 respectively. Thus for COIL20 and COIL100, we set $\sigma = n^{-2}\sum_{ij}\|\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_j\|$ in RNLMF; for Yale Face and AR Face, we set $\sigma = 3n^{-2}\sum_{ij}\|\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_j\|$ and $5n^{-2}\sum_{ij}\|\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_j\|$ in RNLMF respectively. We expect that the improvement given by RNLMF on COIL20 and COIL100 are higher than those on Yale Face and AR Face.

A. Denoising result

The dictionaries given by RDL and RNLMF on COIL20 and Yale Face are visualized in Figure 7 and Figure 8. We see that the dictionary of RNLMF consists of real images, because Gaussian RBF kernel in RNLMF plays a role of smooth interpolation and RNLMF constructs a set of "landmark" points to represent all data point as accurate as possible.

Figure 9 and Figure 10 show some examples of the original images, noisy images, and recovered images of COIL20 and Yale Face. The denoising performance of RNLMF is better than those of RPCA and RDL. The average RMSE and its standard deviation of 20 repeated trials are reported in Table IV. Note that we actually performed NLRR [47] rather than LRR [10] on COIL100 because the data size is large, though we still use the name LRR for consistency. In the table, RNLMF outperformed other methods significantly in all cases.

B. Clustering result

We check the clustering performance of RNLMF compared with LRR [10] [47], SSC [9], KSSC [45], GMC-LRSSC [48], S_0/ℓ_0-LRSSC [48], and EKSS [60] on the four datasets. Since KSSC, GMC-LRSSC, S_0/ℓ_0-LRSSC, and EKSS cannot handle sparse noise we first process the data by RPCA and then implement the four clustering methods. Moreover, in line with [60], we perform EKSS on the features extracted by PCA rather than the pixel values; otherwise, the clustering error of EKSS is too large. In Algorithm 3, we set $\gamma = 0.01$; we set $\kappa = 5$ on Yale Face and COIL100, and set $\kappa = 15$ on the

TABLE IV: RMSE (%) of denoising on the noisy image data

Data	Noise	RPCA	LRR	SSC	RDL	RNLMF
COIL20	Random	12.28±0.04	13.84±0.09	12.05±0.08	11.04±0.14	**9.31**±0.44
	Occlusion	20.54±0.31	18.34±0.45	15.26±0.38	14.89±0.63	**10.25**±0.59
	Random+Occlusion	21.26±0.23	21.81±0.24	18.84±0.23	18.91±0.39	**12.67**±0.56
COIL100	Random	13.75±0.02	13.28±0.05	12.39±0.06	11.89±0.09	**9.15**±0.03
	Occlusion	20.98±0.05	18.95±0.11	18.09±0.12	17.41±0.15	**13.82**±0.17
	Random&Occlusion	23.93±0.22	21.26±0.18	20.05±0.25	20.76±0.58	**14.86**±0.32
Yale Face	Random	9.25±0.03	12.87±0.03	13.04±0.04	9.71±0.11	**7.71**±0.14
	Occlusion	15.80±0.15	13.41±0.14	13.28±0.11	13.68±0.19	**10.96**±0.14
	Random+Occlusion	17.07±0.14	16.05±0.13	16.66±0.12	16.24±0.28	**12.20**±0.29
AR Face	Random	8.41±0.02	9.26±0.03	9.15±0.02	7.81±0.11	**6.01**±0.02
	Occlusion	13.25±0.04	12.19±0.13	11.57±0.11	11.89±0.16	**10.65**±0.07
	Random+Occlusion	13.49±0.05	13.18±0.07	13.04±0.06	12.71±0.12	**11.75**±0.13

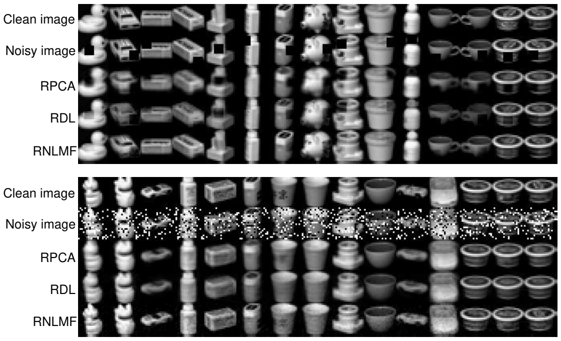

Fig. 9: Denoising COIL20

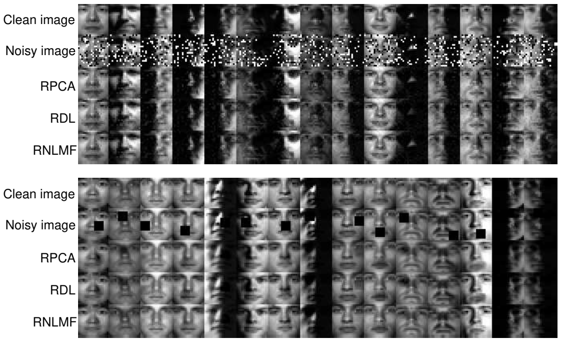

Fig. 10: Denoising Yale Face

TABLE V: Clustering error (%) on the original image data

	LRR	SSC	KSSC	GMC-LRSSC	S_0/ℓ_0-LRSSC	EKSS	RNLMF
COIL20	25.21	14.36	21.94	25.83	18.40	13.47	**13.13**
COIL100	52.28	44.63	44.67	55.04	46.99	28.57	**23.53**
Yale Face	13.26	21.75	20.55	23.20	12.01	14.31	**10.77**
AR Face	19.27	24.61	25.54	18.92	27.15	22.65	**13.88**

COIL20 and AR Face. The hyper-parameters of other methods are carefully tuned to provide their best performances. The clustering errors on the original data are reported in Table V, in which RNLMF has the lowest clustering error in every case. As shown in Table VI, RNLMF outperformed other methods significantly on noisy COIL20, COIL100 and AR Face. The main reason is that RNLMF is more effective than other methods in handling high-rank matrices corrupted by sparse noise. EKSS benefits a lot from the preprocessing of RPCA especially on Yale Face, of which the matrix rank is much lower than those of COIL20 and COIL100.

TABLE VI: Clustering error (%) on the noisy image data

	LRR	SSC	KSSC	GMC-LRSSC	S_0/ℓ_0-LRSSC	EKSS	RNLMF
COIL20	34.79	24.65	40.42	31.11	28.68	21.11	**14.03**
COIL100	65.54	52.17	61.56	65.97	49.58	54.17	**34.18**
Yale Face	55.82	35.21	64.25	33.43	33.14	**18.97**	21.13
AR Face	63.38	39.88	61.50	36.65	38.92	28.96	**23.27**

Fig. 11: A few examples of CMU motion capture data.

X. Experiments on Motion Capture Data

Besides image data sets that consist of the images of multiple objects or subjects, many other data sets in computer vision as well as other areas can also form high-rank matrices. For example, in CMU motion capture database (http://mocap.cs.cmu.edu/), many subsets consist of the time-series trajectories of multiple human motions such as walking, jumping, stretching, and climbing; the dimension of the signal (the number of sensors) is 62, much smaller than the number of samples; the formed matrices are often high-rank because different human motion corresponds to different data latent structure. Figure 11 shows a few examples of the data.

In this paper, we consider the Trial 09 of subject #01 and the Trial 06 of subject #56. The sizes of the corresponding data matrices are 62×4242 and 62×6784, respectively. We add Gaussian noises to 10% or 30% of the entries of the two matrices, where the variance of the noise is the same as that of the data. In RPCA, the parameter λ is set as $1/\sqrt{n}$ or $1.5/\sqrt{n}$;

TABLE VII: RMSE (%) and MAE (%) on motion capture data

	subject	ρ	RPCA	LRR	SSC	RDL	RNLMF
RMSE	#01	0.1	11.19±0.10	9.89±0.08	10.36±0.11	12.75±0.98	**9.78**±0.45
		0.3	21.98±0.13	17.54±0.10	18.66±0.12	23.51±1.22	**11.82**±1.13
	#56	0.1	8.67±0.11	9.06± 0.08	9.95±0.07	9.19±0.50	**6.11**±0.33
		0.3	22.93±0.69	17.73±0.07	18.50±0.07	19.03±0.99	**11.27**±1.17
MAE	#01	0.1	7.10±0.04	7.38±0.08	8.52±0.06	7.33±0.72	**4.90**±0.40
		0.3	21.85±0.20	22.31±0.14	23.75±0.09	23.61±1.76	**10.11**±0.79
	#56	0.1	5.68±0.33	6.57±0.05	8.23±0.05	5.68±0.28	**4.11**±0.28
		0.3	22.23±0.40	18.67±0.13	22.48±0.09	17.89±0.48	**11.04**±0.71

TABLE VIII: RMSE (%) and MAE (%) on motion capture data (out-of-sample-extension)

			Training data			Testing data		
	subject	ρ	RPCA	RDL	RNLMF	RPCA	RDL	RNLMF
RMSE	#01	0.1	11.34±0.39	12.20±1.19	**9.21**±1.02	13.78±0.33	11.98±0.51	**7.51**±0.87
		0.3	21.96±0.17	23.06±1.14	**11.45**±0.73	20.35±0.34	22.83±0.96	**10.86**±1.09
	#56	0.1	8.65±0.11	9.07±0.83	**6.04**±0.52	10.24±0.19	8.92±1.04	**5.57**±0.28
		0.3	22.51±1.14	19.87±1.19	**10.89**±0.69	17.37±0.50	18.94±1.16	**10.58**±0.62
MAE	#01	0.1	7.13±0.10	7.37±0.58	**4.83**±0.68	9.65±0.13	7.96±0.41	**4.93**±0.60
		0.3	22.03±0.32	22.04±1.33	**10.11**±0.32	23.51±0.28	22.52±1.03	**10.33**±0.28
	#56	0.1	5.70±0.08	5.81±0.42	**4.07**±0.25	7.86±0.90	5.93±0.36	**4.05**±0.22
		0.3	22.04±0.69	18.06±0.62	**10.78**±0.53	19.07±0.77	16.94±0.55	**10.69**±0.50

the Lagrange penalty parameter is set as λ. In RDL, we set $d = 31$, $\lambda_C = 0.07$ or 0.08, and $\lambda_E = 2$ or 3. In RNLMF, we set $d = 62$, $\sigma = 0.5n^{-2}\sum_{ij}\|\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_j\|$, $\lambda_C = 0.01$, and choose λ_E from $\{2\times 10^{-5}, 4\times 10^{-5}, 5\times 10^{-5}\}$.

Since the variances of the 62 signals are not at the same level, we also consider the normalized mean-absolute-error

$$\text{MAE} := \|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_1/\|\boldsymbol{X}\|_1.$$

Table VII shows the average RMSE and MAE of 20 repeated trials and the standard deviation. RNLMF outperformed other methods significantly especially when $\rho = 0.3$.

We randomly split the data into two subsets of equal size. We perform RPCA, RDL, and RNLMF on one subset (training data) and then use the trained model to denoise the other subset (testing data). Let $\hat{\boldsymbol{X}}'$ be the noisy testing data. For RPCA, we consider the following problem

$$\underset{\boldsymbol{V},\boldsymbol{E}'}{\text{minimize}}\ \frac{1}{2}\|\hat{\boldsymbol{X}}' - \boldsymbol{U}\boldsymbol{V} - \boldsymbol{E}'\|_F^2 + \lambda_V\|\boldsymbol{V}\|_F^2 + \lambda_E\|\boldsymbol{E}'\|_1, \quad (27)$$

where $\boldsymbol{U} \in \mathbb{R}^{m\times r}$ consists of the first r left singular vectors of $\boldsymbol{X}$ obtained by solving (1) and r is set to be 10 or 20 in this study. For RDL, we consider

$$\underset{\boldsymbol{C}',\boldsymbol{E}'}{\text{minimize}}\ \frac{1}{2}\|\hat{\boldsymbol{X}}' - \boldsymbol{D}\boldsymbol{C}' - \boldsymbol{E}'\|_F^2 + \lambda_C\|\boldsymbol{C}'\|_1 + \lambda_E\|\boldsymbol{E}'\|_1, \quad (28)$$

where $\boldsymbol{D}$ is obtained by solving (6). Notice that the out-of-sample extensions of LRR and SSC use the whole data matrix $\boldsymbol{X}$ as a dictionary and hence are not efficient, compared to those of RPCA, RDL, and RNLMF. Moreover, the performance of LRR and SSC are similar to those of RPCA and RDL. Therefore, for simplicity, the out-of-sample extensions of LRR and SSC will not be considered in this study.

The results of 20 repeated trials are reported in Table VIII. We see that the recovery performance on training data and testing data are similar. In addition, RNLMF is more effective than RPCA and RDL in denoising new data. The results in Table VIII are also similar to those of RNLMF in Table VII. We conclude that the dictionary matrix $\boldsymbol{D}$ given by RNLMF can be used to denoise new data efficiently and the denosing accuracy is comparable to that on the training data.

XI. Conclusion

We have proposed a new method called RNLMF to recover high-rank matrices from sparse noise. We analyzed the underlying meaning of the factorization loss and the regularization terms in the objective function. RNLMF can be used in robust dicionary learning, denoising, and clustering and is also scalable to large-scale data. Comparative studies on synthetic data and real data verified the superiority of RNLMF. One interesting finding is that in RNLMF, $\mathcal{R} = \|\boldsymbol{C}\|_F^2$ yields higher recover accuracy, compared to $\mathcal{R} = \|\boldsymbol{C}\|_*$ and $\|\boldsymbol{C}\|_1$. The reason has been analyzed in Section VIII. It is possible that a sparse $\boldsymbol{C}$ given by RNLMF is more useful in sparse coding based classification.

Acknowledgment

Jicong Fan gratefully acknowledges the support from the funding NO.T00120210002 of Shenzhen Research Institute of Big Data. Chengrun Yang and Madeline Udell gratefully acknowledge the support from NSF Awards IIS-1943131 and CCF-1740822, the ONR Young Investigator Program, DARPA Award FA8750-17-2-0101, the Simons Institute, and Capital One.

Appendix

A. Proof for Lemma 1

Proof. Let $\bar{\boldsymbol{x}} = [\boldsymbol{x}; \sqrt{c}]$.

$$\begin{aligned}
&\mathcal{K}_\sigma(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathcal{K}_\sigma(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_j) \\
&= \exp\left(-\frac{1}{2\sigma^2}\left(\|\bar{\boldsymbol{x}}_i\|^2 + \|\bar{\boldsymbol{x}}_j\|^2 - 2\langle \bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_j\rangle\right)\right) \\
&= \exp\left(-\frac{\|\bar{\boldsymbol{x}}_i\|^2 + \|\bar{\boldsymbol{x}}_j\|^2}{2\sigma^2}\right)\exp\left(\frac{1}{\sigma^2}\langle \bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_j\rangle\right) \\
&= \exp\left(-\frac{\|\bar{\boldsymbol{x}}_i\|^2 + \|\bar{\boldsymbol{x}}_j\|^2}{2\sigma^2}\right)\sum_{u=0}^{\infty}\frac{\langle \bar{\boldsymbol{x}}_i, \bar{\boldsymbol{x}}_j\rangle^u}{\sigma^{2u}u!} \\
&= \exp\left(-\frac{\|\bar{\boldsymbol{x}}_i\|^2 + \|\bar{\boldsymbol{x}}_j\|^2}{2\sigma^2}\right)\sum_{u=0}^{\infty}\frac{(\boldsymbol{x}_i^\top \boldsymbol{x}_j + c)^u}{\sigma^{2u}u!} \\
&= \exp\left(-\frac{\|\boldsymbol{x}_i\|^2 + \|\boldsymbol{x}_j\|^2 + 2c}{2\sigma^2}\right)\sum_{u=0}^{\infty}\frac{\mathcal{K}_{c,u}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sigma^{2u}u!}.
\end{aligned}$$

□

B. Proof for Corollary 1

Proof. Since $d \geq \operatorname{rank}(\phi_{c,q}(\boldsymbol{X}))$, there exist $\boldsymbol{D} \in \mathbb{R}^{m\times d}$ and $\bar{\boldsymbol{C}} \in \mathbb{R}^{d\times n}$ such that

$$\phi_{c,q}(\boldsymbol{X}) = \phi_{c,q}(\boldsymbol{D})\bar{\boldsymbol{C}}.$$

Let $\boldsymbol{C} = \boldsymbol{S}_D^{-1}\bar{\boldsymbol{C}}\boldsymbol{S}_X$, then

$$\phi_{c,q}(\boldsymbol{X})\boldsymbol{S}_X = \phi_{c,q}(\boldsymbol{D})\boldsymbol{S}_D\boldsymbol{C}.$$

As $\phi_{c,q}$ contains all features of $\phi_{c,u}$ with $u \leq q$, i.e. $\phi_{c,q} = [\phi_{c,u}, \ldots]^\top$, we have

$$\phi_{c,u}(\boldsymbol{X})\boldsymbol{S}_X = \phi_{c,u}(\boldsymbol{D})\boldsymbol{S}_D\boldsymbol{C}, \quad \forall\ 0 \leq u \leq q.$$

Combing these equalities with Lemma 2, we finish the proof. □

C. Proof for Lemma 2

Proof. Define $w_u = 1/(\sigma^u\sqrt{u!})$. We have

$$\begin{aligned}
&\frac{1}{2}\|\phi_\sigma(\boldsymbol{X}) - \phi_\sigma(\boldsymbol{D})\boldsymbol{C}\|_F^2 \\
=&\sum_{u=0}^{\infty}\frac{w_u^2}{2}\|\phi_{c,u}(\boldsymbol{X})\boldsymbol{S}_X - \phi_{c,u}(\boldsymbol{D})\boldsymbol{S}_D\boldsymbol{C}\|_F^2 \\
=&\sum_{u=0}^{q}\frac{w_u^2}{2}\|\phi_{c,u}(\boldsymbol{X})\boldsymbol{S}_X - \phi_{c,u}(\boldsymbol{D})\boldsymbol{S}_D\boldsymbol{C}\|_F^2 \\
&+ R_1 + R_2 + R_3,
\end{aligned} \tag{29}$$

where $R_1 = \sum_{u=q+1}^{\infty}\frac{w_u^2}{2}\operatorname{Tr}\big(\mathcal{K}_{c,u}(\boldsymbol{X}, \boldsymbol{X})\boldsymbol{S}_X\boldsymbol{S}_X\big)$, $R_2 = -\sum_{u=q+1}^{\infty} w_u^2\operatorname{Tr}\big(\boldsymbol{C}^\top\boldsymbol{S}_D^\top\mathcal{K}_{c,u}(\boldsymbol{D}, \boldsymbol{X})\boldsymbol{S}_X\big)$, and $R_3 = \sum_{u=q+1}^{\infty}\frac{w_u^2}{2}(\operatorname{Tr}\big(\boldsymbol{C}^\top\boldsymbol{S}_D^\top\mathcal{K}_{c,u}(\boldsymbol{D}, \boldsymbol{D})\boldsymbol{S}_D\boldsymbol{C}\big)$. Suppose $\sigma^2 > \kappa_2 + c$. We have

$$\begin{aligned}
|R_1| &= \sum_{i=1}^{n} s_i^2\sum_{u=q+1}^{\infty}\frac{w_u^2}{2}(\|\boldsymbol{x}_i\|^2 + c)^u \\
&\leq \sum_{i=1}^{n}\frac{s_i^2}{2(q!)}\Big(\frac{\|\boldsymbol{x}_i\|^2 + c}{\sigma^2}\Big)^q \\
&\leq \frac{0.5n\exp(-\frac{c}{\sigma^2})}{q!}\Big(\frac{\max_i\|\boldsymbol{x}_i\|^2 + c}{\sigma^2}\Big)^q.
\end{aligned} \tag{30}$$

$$\begin{aligned}
|R_2| \leq& \sum_{u=q+1}^{\infty} w_u^2\|\boldsymbol{C}\|_F\|\boldsymbol{S}_D\|_2\|\boldsymbol{S}_X\|_2\|\mathcal{K}_{c,u}(\boldsymbol{D}, \boldsymbol{X})\|_F \\
&\leq \sqrt{dn}\exp(-\tfrac{c}{\sigma^2})\|\boldsymbol{C}\|_F\sum_{u=q+1}^{\infty} w_u^2(\max_{ij}|\boldsymbol{x}_i^\top\boldsymbol{d}_j + c|)^u \\
&\leq \frac{\sqrt{dn}\exp(-\frac{c}{\sigma^2})\|\boldsymbol{C}\|_F}{q!}\big(\frac{\max_{ij}\|\boldsymbol{x}_i\|\|\boldsymbol{d}_j\| + c}{\sigma^2}\big)^q.
\end{aligned} \tag{31}$$

$$\begin{aligned}
|R_3| \leq& \sum_{u=q+1}^{\infty}\frac{w_u^2}{2}\|\boldsymbol{C}\|_2\|\boldsymbol{C}\|_F\|\boldsymbol{S}_D\|_2^2\|\mathcal{K}_{c,u}(\boldsymbol{D}, \boldsymbol{D})\|_F \\
&\leq 0.5d\exp(-\tfrac{c}{\sigma^2})\|\boldsymbol{C}\|_2\|\boldsymbol{C}\|_F\sum_{u=q+1}^{\infty} w_u^2(\max_{ij}|\boldsymbol{d}_i^\top\boldsymbol{d}_j + c|)^u \\
&\leq \frac{0.5d\exp(-\frac{c}{\sigma^2})\|\boldsymbol{C}\|_2\|\boldsymbol{C}\|_F}{q!}\big(\frac{\max_i\|\boldsymbol{d}_i\|^2 + c}{\sigma^2}\big)^q.
\end{aligned} \tag{32}$$

Let $\kappa_1 = \max\{0.5n, \sqrt{dn}\|\boldsymbol{C}\|_F, 0.5d\|\boldsymbol{C}\|_2\|\boldsymbol{C}\|_F\}$ and $\kappa_2 = \max\{\max_i\|\boldsymbol{x}_i\|^2, \max_j\|\boldsymbol{d}_j\|^2\}$. We obtain

$$|R_1| + |R_2| + |R_3| \leq \frac{3\kappa_1\exp(-\frac{c}{\sigma^2})}{q!}\big(\frac{\kappa_2 + c}{\sigma^2}\big)^q. \tag{33}$$

□

D. Proof for Lemma 3

Proof. It is known that

$$\min_{\boldsymbol{BC}=\phi(\boldsymbol{X})}\frac{1}{2}\|\boldsymbol{B}\|_F^2 + \frac{1}{2}\|\boldsymbol{C}\|_F^2 = \|\phi(\boldsymbol{X})\|_*. \tag{34}$$

Considering one more constraint $\boldsymbol{B} = \phi(\boldsymbol{D})$, we must have

$$\begin{aligned}
&\min_{\boldsymbol{BC}=\phi(\boldsymbol{X}),\boldsymbol{B}=\phi(\boldsymbol{D})}\frac{1}{2}\|\phi(\boldsymbol{D})\|_F^2 + \frac{1}{2}\|\boldsymbol{C}\|_F^2 \\
\geq& \min_{\boldsymbol{BC}=\phi(\boldsymbol{X})}\frac{1}{2}\|\boldsymbol{B}\|_F^2 + \frac{1}{2}\|\boldsymbol{C}\|_F^2.
\end{aligned} \tag{35}$$

Combining (34) and (35) finishes the proof. □

E. Proof for Lemma 4

Proof. For all $c > 0$, we have

$$\|\phi(\boldsymbol{D})\|_F^2 + c\|\boldsymbol{C}\|_F^2 \geq 2\sqrt{c}\|\phi(\boldsymbol{D})\boldsymbol{C}\|_*.$$

Choosing $c = \|\phi(\boldsymbol{D})\|_F^2/\|\boldsymbol{C}\|_F^2$, we have

$$\|\boldsymbol{C}\|_F\|\phi(\boldsymbol{D})\|_F \geq \|\phi(\boldsymbol{D})\boldsymbol{C}\|_*.$$

Recalling $\|\phi(\boldsymbol{D})\|_F^2 \equiv d$, we arrive at

$$\|\boldsymbol{C}\|_F \geq \|\phi(\boldsymbol{D})\boldsymbol{C}\|_*/\sqrt{d}.$$

□

F. Proof for Lemma 5

Proof. Note that $\nabla_{\boldsymbol{C}}\mathcal{L}(\boldsymbol{C}) = -\mathcal{K}(\boldsymbol{D}_{t-1}, \hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1}) + \mathcal{K}(\boldsymbol{D}_{t-1}, \boldsymbol{D}_{t-1})\boldsymbol{C}$ is L_C^t-Lipschitz continuous, where $L_C^t = \|\mathcal{K}(\boldsymbol{D}_{t-1}, \boldsymbol{D}_{t-1})\|_2$. Thus

$$\begin{aligned}\mathcal{L}(\boldsymbol{C}_t) \leq \mathcal{L}(\boldsymbol{C}_{t-1}) + \langle \boldsymbol{C}_t - \boldsymbol{C}_{t-1}, \nabla_{\boldsymbol{C}}\mathcal{L}(\boldsymbol{C}_{t-1})\rangle \\ + \frac{L_C^t}{2}\|\boldsymbol{C}_t - \boldsymbol{C}_{t-1}\|_F^2.\end{aligned} \tag{36}$$

According to the definition of the proximal map Θ_u (or Ψ_u) [52], we have

$$\begin{aligned}\boldsymbol{C}_t \in \min_{\boldsymbol{C}} \langle \boldsymbol{C} - \boldsymbol{C}_{t-1}, \nabla_{\boldsymbol{C}}\mathcal{L}(\boldsymbol{C}_{t-1})\rangle \\ + \frac{\tau_C^t}{2}\|\boldsymbol{C} - \boldsymbol{C}_{t-1}\|_F^2 + \lambda_C\mathcal{R}(\boldsymbol{C}).\end{aligned} \tag{37}$$

where $\mathcal{R}(\boldsymbol{C}) = \|\boldsymbol{C}\|_1$ (or $\|\boldsymbol{C}\|_*$). By taking $\boldsymbol{C} = \boldsymbol{C}_{t-1}$, it follows from (37) that

$$\begin{aligned}&\langle \boldsymbol{C}_t - \boldsymbol{C}_{t-1}, \nabla_{\boldsymbol{C}}\mathcal{L}(\boldsymbol{C}_{t-1})\rangle + \lambda_C\mathcal{R}(\boldsymbol{C}_t) \\ \leq&\lambda_C\mathcal{R}(\boldsymbol{C}_{t-1}) - \frac{\tau_C^t}{2}\|\boldsymbol{C}_t - \boldsymbol{C}_{t-1}\|_F^2.\end{aligned} \tag{38}$$

Combining (36) and (38), we have

$$\begin{aligned}\mathcal{L}(\boldsymbol{C}_t) + \lambda_C\mathcal{R}(\boldsymbol{C}_t) \leq&\mathcal{L}(\boldsymbol{C}_{t-1}) + \lambda_C\mathcal{R}(\boldsymbol{C}_{t-1}) \\ &- \frac{\tau_C^t - L_C^t}{2}\|\boldsymbol{C}_t - \boldsymbol{C}_{t-1}\|_F^2.\end{aligned}$$

This finished the proof. □

G. Proof for Lemma 6

Proof. Recall that $\mathcal{L}(\boldsymbol{D}) = -\text{Tr}\left(\boldsymbol{C}_t^\top\mathcal{K}(\boldsymbol{D}, \hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1})\right) + \frac{1}{2}\text{Tr}\left(\boldsymbol{C}_t^\top\mathcal{K}(\boldsymbol{D}, \boldsymbol{D})\boldsymbol{C}_t\right)$ and the gradient is

$$\begin{aligned}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}) =&\tfrac{1}{\sigma^2}(\hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1})\boldsymbol{W}_D - \tfrac{1}{\sigma^2}\boldsymbol{D}\bar{\boldsymbol{W}}_D \\ &+ \tfrac{2}{\sigma^2}\boldsymbol{D}\boldsymbol{Q}_D - \tfrac{2}{\sigma^2}\boldsymbol{D}\bar{\boldsymbol{Q}}_D,\end{aligned}$$

where $\boldsymbol{W}_D = -\boldsymbol{C}_t^\top \odot \mathcal{K}(\hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1}, \boldsymbol{D})$, $\boldsymbol{Q}_D = (0.5\boldsymbol{C}_t\boldsymbol{C}_t^\top) \odot \mathcal{K}(\boldsymbol{D}, \boldsymbol{D})$, $\bar{\boldsymbol{W}}_D = \text{diag}(\boldsymbol{1}_n^\top\boldsymbol{W}_D)$, and $\bar{\boldsymbol{Q}}_D = \text{diag}(\boldsymbol{1}_d^\top\boldsymbol{Q}_D)$. Let $s_{ij} = \exp(-\frac{\|[\hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1}]_{:i}\|^2 + \|[\boldsymbol{D}]_{:j}\|^2}{2\sigma^2})$ and suppose σ is large enough. Then according to Lemma 1, we have

$$\boldsymbol{W}_D \approx -\boldsymbol{C}_t^\top \odot \boldsymbol{S} \odot (1 + \tfrac{(\hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1})^\top\boldsymbol{D}}{\sigma^2}),$$

where we have omitted the higher ($u > 2$) order terms of the polynomial approximate of Gaussian RBF kernel. Similarly, we have

$$\boldsymbol{Q}_D \approx (0.5\boldsymbol{C}_t\boldsymbol{C}_t^\top) \odot \boldsymbol{S}_D \odot (1 + \tfrac{\boldsymbol{D}^\top\boldsymbol{D}}{\sigma^2}),$$

where $[\boldsymbol{S}_D]_{ij} = \exp(-\frac{\|[\boldsymbol{D}]_{:i}\|^2 + \|[\boldsymbol{D}]_{:j}\|^2}{2\sigma^2})$.

Let's check the sensitivity of $\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D})$ to the perturbation on $\boldsymbol{D}$. First, consider the first term in $\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D})$ and let $[\hat{\boldsymbol{S}}]_{ij} = \exp(-\frac{\|[\hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1}]_{:i}\|^2 + \|[\hat{\boldsymbol{D}}]_{:j}\|^2}{2\sigma^2})$. We have

$$\begin{aligned}&\|\tfrac{1}{\sigma^2}(\hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1})(\boldsymbol{C}_t^\top \odot (\boldsymbol{S} \odot (1 + \tfrac{(\hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1})^\top\boldsymbol{D}}{\sigma^2}) \\ &\quad - \hat{\boldsymbol{S}} \odot (1 + \tfrac{(\hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1})^\top\hat{\boldsymbol{D}}}{\sigma^2})))\|_F \\ \leq&\tfrac{1}{\sigma^2}\|\hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1}\|\|\boldsymbol{C}_t\|_\infty \max\{2\|\boldsymbol{S}\|_\infty, 2\|\hat{\boldsymbol{S}}\|_\infty\} \\ &\quad \times \|\sigma^{-2}(\hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1})(\boldsymbol{D} - \hat{\boldsymbol{D}})\|_F \\ \leq&\tfrac{2\|\boldsymbol{C}_t\|_\infty}{\sigma^4}\|\hat{\boldsymbol{X}} - \boldsymbol{E}_{t-1}\|^2\|\boldsymbol{D} - \hat{\boldsymbol{D}}\|_F,\end{aligned} \tag{39}$$

where $\|\boldsymbol{S}\|_\infty$,$\|\hat{\boldsymbol{S}}\|_\infty \leq 1$. We see that when σ is large and $\|\boldsymbol{C}\|_\infty$ is small, the first term in $\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D})$ is not sensitive to the changes of $\boldsymbol{D}$.

Now consider the third term of $\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D})$. Let $\boldsymbol{Q}_{\hat{D}}$ be the perturbed copy of $\boldsymbol{Q}_D$ computed from $\hat{\boldsymbol{D}}$. We have

$$\begin{aligned}&\tfrac{2}{\sigma^2}\|\boldsymbol{D}\boldsymbol{Q}_D - \hat{\boldsymbol{D}}\boldsymbol{Q}_{\hat{D}}\|_F \\ \approx&\tfrac{2}{\sigma^2}\|(\boldsymbol{D} - \hat{\boldsymbol{D}})\boldsymbol{Q}_D - \hat{\boldsymbol{D}}((0.5\boldsymbol{C}_t\boldsymbol{C}_t^\top) \odot \boldsymbol{S}_D \odot (1 + \tfrac{\boldsymbol{D}^\top\boldsymbol{D}}{\sigma^2}) \\ &\quad - (0.5\boldsymbol{C}_t\boldsymbol{C}_t^\top) \odot \hat{\boldsymbol{S}}_D \odot (1 + \tfrac{\hat{\boldsymbol{D}}^\top\hat{\boldsymbol{D}}}{\sigma^2}))\|_F \\ \leq&\tfrac{2}{\sigma^2}\|\boldsymbol{Q}_D\|\|\boldsymbol{D} - \hat{\boldsymbol{D}}\|_F + \tfrac{2}{\sigma^4}\|\hat{\boldsymbol{D}}\|\|\boldsymbol{C}_t\boldsymbol{C}_t^\top\|_\infty\|\boldsymbol{D}^\top\boldsymbol{D} - \hat{\boldsymbol{D}}^\top\hat{\boldsymbol{D}}\|_F \\ \leq&\tfrac{2}{\sigma^2}\|\boldsymbol{Q}_D\|\|\boldsymbol{D} - \hat{\boldsymbol{D}}\|_F \\ &\quad + \tfrac{2}{\sigma^4}(\|\hat{\boldsymbol{D}}\|^2 + \|\boldsymbol{D}\|\|\hat{\boldsymbol{D}}\|)\|\boldsymbol{C}_t\boldsymbol{C}_t^\top\|_\infty\|\boldsymbol{D} - \hat{\boldsymbol{D}}\|_F.\end{aligned} \tag{40}$$

In (40), when σ is large enough, the second term can be smaller than the first term and $\boldsymbol{Q}_D \approx 0.5\boldsymbol{C}_t\boldsymbol{C}_t^\top$. It means the $\boldsymbol{D}$ in $\boldsymbol{Q}_D$ makes small contribution to $\boldsymbol{D}\boldsymbol{Q}_D$. Therefore, the contribution of $\boldsymbol{D}$ in $\boldsymbol{Q}_D$ to the Hessian of $\mathcal{L}(\boldsymbol{D})$ can be neglected. The conclusion also applies to $\bar{\boldsymbol{Q}}_D$ and $\bar{\boldsymbol{W}}_D$. In addition, comparing (39) with (40), we can neglect the contribution of the first term of $\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D})$ to the Hessian of $\mathcal{L}(\boldsymbol{D})$.

Now let's consider second order approximation of $\mathcal{L}(D)$ around $\boldsymbol{D}_{t-1}$:

$$\begin{aligned}\mathcal{L}(\boldsymbol{D}) \approx&\mathcal{L}(\boldsymbol{D}_{t-1}) + \langle \nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1}), \boldsymbol{D} - \boldsymbol{D}_{t-1}\rangle \\ &+ \tfrac{1}{2}\text{vec}(\boldsymbol{D} - \boldsymbol{D_{t-1}})^\top\boldsymbol{\mathcal{H}}_{t-1}\text{vec}(\boldsymbol{D} - \boldsymbol{D_{t-1}})^\top,\end{aligned} \tag{41}$$

where $\boldsymbol{\mathcal{H}}_{t-1}$ is the Hessian at iteration $t-1$. It is difficult to compute $\boldsymbol{\mathcal{H}}_{t-1}$. However, our previous analysis indicates that we can treat $\boldsymbol{W}_D$, $\bar{\boldsymbol{W}}_D$, $\boldsymbol{Q}_D$, and $\bar{\boldsymbol{Q}}_D$ as constants independent of $\boldsymbol{D}$ at iteration t. Thus the estimated Hessian is

$$\hat{\boldsymbol{\mathcal{H}}}_{t-1} = \begin{bmatrix}\boldsymbol{H}_{t-1} & \dots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{H}_{t-1}\end{bmatrix} \in \mathbb{R}^{md \times md}$$

where $\boldsymbol{H}_{t-1} = \frac{1}{\sigma^2}(-\bar{\boldsymbol{W}}_{D_{t-1}} + 2\boldsymbol{Q}_{D_{t-1}} - 2\bar{\boldsymbol{Q}}_{D_{t-1}}) \in \mathbb{R}^{d \times d}$.

Let $\mu \geq 0$ be sufficiently large such that $\boldsymbol{H}_{t-1} + \mu\boldsymbol{I}$ is positive definite. Let $\tau_D^t \geq 1$ be sufficiently large such that

$$\begin{aligned}\mathcal{L}(\boldsymbol{D}) \leq&\mathcal{L}(\boldsymbol{D}_{t-1}) + \langle \nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1}), \boldsymbol{D} - \boldsymbol{D}_{t-1}\rangle \\ &+ \tfrac{\tau_D^t}{2}\text{Tr}\left((\boldsymbol{D} - \boldsymbol{D_{t-1}})(\boldsymbol{H}_{t-1} + \mu\boldsymbol{I})(\boldsymbol{D} - \boldsymbol{D_{t-1}})^\top\right).\end{aligned} \tag{42}$$

We then minimize the right side of (42) by letting the derivative be zero and get

$$\boldsymbol{D}_t = \boldsymbol{D}_{t-1} - \frac{1}{\tau_D^t}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})(\boldsymbol{H}_{t-1} + \mu\boldsymbol{I})^{-1}. \tag{43}$$

Invoking (43) into (42) yields

$$\begin{aligned}&\mathcal{L}(\boldsymbol{D}_t) \leq \mathcal{L}(\boldsymbol{D}_{t-1}) \\ &- \frac{1}{2\tau_D^t}\text{Tr}\left(\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})(\boldsymbol{H}_{t-1} + \mu\boldsymbol{I})^{-1}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})^\top\right).\end{aligned}$$

Since $\boldsymbol{H}_{t-1} + \mu\boldsymbol{I}$ is positive definite, we have

$$\mathcal{L}(\boldsymbol{D}_t) - \mathcal{L}(\boldsymbol{D}_{t-1}) \leq 0.$$

This finished the proof. □

H. Proof for Corollary 3

Proof. Recall that $\boldsymbol{D}_t = \boldsymbol{D}_{t-1} - \boldsymbol{\Delta}_t$, where $\boldsymbol{\Delta}_t = \eta\boldsymbol{\Delta}_{t-1} + \frac{1}{\tau_D^t}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})(\boldsymbol{H}_{t-1}+\mu\boldsymbol{I})^{-1}$ and $0 < \eta < 1$. Follow the same analysis on $\boldsymbol{H}$ in the proof of Lemma 6. In (42), letting $\boldsymbol{D} = \boldsymbol{D}_t$, we have

$$
\begin{aligned}
&\mathcal{L}(\boldsymbol{D}_t)\\
\leq&\mathcal{L}(\boldsymbol{D}_{t-1}) - \langle\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1}), \eta\boldsymbol{\Delta}_{t-1}\rangle\\
&-\langle\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1}), \tfrac{1}{\tau_D^t}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})(\boldsymbol{H}_{t-1}+\mu\boldsymbol{I})^{-1}\rangle\\
&+\tfrac{\tau_D^t}{2}\mathrm{Tr}\left(\eta^2\boldsymbol{\Delta}_{t-1}(\boldsymbol{H}_{t-1}+\mu\boldsymbol{I})\boldsymbol{\Delta}_{t-1}^\top\right)\\
&+\tfrac{\tau_D^t}{2}\mathrm{Tr}\left(\eta\boldsymbol{\Delta}_{t-1}(\boldsymbol{H}_{t-1}+\mu\boldsymbol{I})(\tfrac{1}{\tau_D^t}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})(\boldsymbol{H}_{t-1}+\mu\boldsymbol{I})^{-1})^\top\right)\\
&+\tfrac{\tau_D^t}{2}\mathrm{Tr}\left((\tfrac{1}{\tau_D^t}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})(\boldsymbol{H}_{t-1}+\mu\boldsymbol{I})^{-1})(\boldsymbol{H}_{t-1}+\mu\boldsymbol{I})\eta\boldsymbol{\Delta}_{t-1}^\top\right)\\
&+\tfrac{1}{2\tau_D^t}\mathrm{Tr}\left(\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})(\boldsymbol{H}_{t-1}+\mu\boldsymbol{I})^{-1}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})^\top\right)\\
=&\mathcal{L}(\boldsymbol{D}_{t-1}) - \tfrac{1}{2\tau_D^t}\mathrm{Tr}\left(\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})(\boldsymbol{H}_{t-1}+\mu\boldsymbol{I})^{-1}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})^\top\right)\\
&-\langle\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1}), \eta\boldsymbol{\Delta}_{t-1}\rangle\\
&+\tfrac{\eta^2\tau_D^t}{2}\mathrm{Tr}\left(\boldsymbol{\Delta}_{t-1}(\boldsymbol{H}_{t-1}+\mu\boldsymbol{I})\boldsymbol{\Delta}_{t-1}^\top\right)\\
&+\eta\mathrm{Tr}\left(\boldsymbol{\Delta}_{t-1}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})^\top\right)\\
=&\mathcal{L}(\boldsymbol{D}_{t-1}) - \tfrac{1}{2\tau_D^t}\mathrm{Tr}\left(\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})(\boldsymbol{H}_{t-1}+\mu\boldsymbol{I})^{-1}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})^\top\right)\\
&+\tfrac{\eta^2\tau_D^t}{2}\mathrm{Tr}\left(\boldsymbol{\Delta}_{t-1}(\boldsymbol{H}_{t-1}+\mu\boldsymbol{I})\boldsymbol{\Delta}_{t-1}^\top\right)
\end{aligned}
$$

□

I. Proof for Lemma 7

Proof. Note that

$$\nabla_{\boldsymbol{E}}\mathcal{L}(\boldsymbol{E}) = \tfrac{1}{\sigma^2}\left((\hat{\boldsymbol{X}}-\boldsymbol{E})\bar{\boldsymbol{G}}_E - \boldsymbol{D}_t\boldsymbol{G}_E\right),$$

where $\boldsymbol{G}_E = -\boldsymbol{C}_t \odot \mathcal{K}(\boldsymbol{D}_t, \hat{\boldsymbol{X}}-\boldsymbol{E})$ and $\bar{\boldsymbol{G}}_E = \mathrm{diag}(\boldsymbol{1}_d^\top\boldsymbol{G}_E)$. According to Lemma 1 (let $c = 0$ and assume σ is large enough), we have

$$[\boldsymbol{G}_E]_{ij} \approx -[\boldsymbol{C}_t]_{ij}s_{ij}(1+\tfrac{[\boldsymbol{D}_t]_{:i}^\top(\boldsymbol{x}_j-\boldsymbol{e}_j)}{\sigma^2}) \approx -[\boldsymbol{C}_t]_{ij},$$

where $s_{ij} = \exp(-\frac{\|[\boldsymbol{D}_t]_{:i}\|^2+\|\boldsymbol{x}_j-\boldsymbol{e}_j\|^2}{2\sigma^2})$. Let $\hat{\boldsymbol{E}}$ be an perturbed copy of $\boldsymbol{E}$. It follows that

$$
\begin{aligned}
&\|\boldsymbol{G}_E - \boldsymbol{G}_{\hat{E}}\|_F\\
\approx&\|\boldsymbol{C}_t\odot\boldsymbol{S}\odot(1+\tfrac{\boldsymbol{D}_t^\top(\boldsymbol{X}-\boldsymbol{E})}{\sigma^2}) - \boldsymbol{C}_t\odot\hat{\boldsymbol{S}}\odot(1+\tfrac{\boldsymbol{D}_t^\top(\boldsymbol{X}-\hat{\boldsymbol{E}})}{\sigma^2})\|_F\\
\leq&2\sigma^{-2}\|\boldsymbol{C}\|_\infty\max\{\|\boldsymbol{S}\|_\infty, \|\hat{\boldsymbol{S}}\|_\infty\}\|\boldsymbol{D}_t^\top(\boldsymbol{E}-\hat{\boldsymbol{E}})\|_F\\
\leq&2\sigma^{-2}\|\boldsymbol{C}\|_\infty\|\boldsymbol{D}_t\|\|\boldsymbol{E}-\hat{\boldsymbol{E}}\|_F.
\end{aligned}
$$

In addition, $\|\bar{\boldsymbol{G}}_E - \bar{\boldsymbol{G}}_{\hat{E}}\|_F \leq \sqrt{d}\|\boldsymbol{G}_E - \boldsymbol{G}_{\hat{E}}\|_F$.

Then we have

$$
\begin{aligned}
&\|\nabla_{\boldsymbol{E}}\mathcal{L}(\boldsymbol{E}) - \nabla_{\hat{\boldsymbol{E}}}\mathcal{L}(\hat{\boldsymbol{E}})\|_F\\
\leq&\tfrac{1}{\sigma^2}\|(\hat{\boldsymbol{X}}-\boldsymbol{E})\bar{\boldsymbol{G}}_E - \boldsymbol{D}_t\boldsymbol{G}_E - ((\hat{\boldsymbol{X}}-\boldsymbol{E})\bar{\boldsymbol{G}}_{\hat{E}} - \boldsymbol{D}_t\boldsymbol{G}_{\hat{E}})\|_F\\
&+\|(\hat{\boldsymbol{X}}-\boldsymbol{E})\bar{\boldsymbol{G}}_{\hat{E}} - \boldsymbol{D}_t\boldsymbol{G}_{\hat{E}} - ((\hat{\boldsymbol{X}}-\hat{\boldsymbol{E}})\bar{\boldsymbol{G}}_{\hat{E}} - \boldsymbol{D}_t\boldsymbol{G}_{\hat{E}})\|_F\\
\leq&\tfrac{1}{\sigma^2}\|\hat{\boldsymbol{X}}-\boldsymbol{E}\|\|\bar{\boldsymbol{G}}_E - \bar{\boldsymbol{G}}_{\hat{E}}\|_F + \tfrac{1}{\sigma^2}\|\boldsymbol{D}_t\|\|\boldsymbol{G}_E - \boldsymbol{G}_{\hat{E}}\|_F\\
&+\tfrac{1}{\sigma^2}\|\bar{\boldsymbol{G}}_{\hat{E}}\|\|\boldsymbol{E}-\hat{\boldsymbol{E}}\|_F\\
\lessapprox&(\tfrac{2\|\boldsymbol{C}\|_\infty}{\sigma^4}(\sqrt{d}\|\hat{\boldsymbol{X}}-\boldsymbol{E}\|\|\boldsymbol{D}_t\| + \|\boldsymbol{D}_t\|^2) + \tfrac{1}{\sigma^2}\|\bar{\boldsymbol{G}}_{\hat{E}}\|)\|\boldsymbol{E}-\hat{\boldsymbol{E}}\|_F\\
\leq&\tfrac{\xi}{\sigma^2}\|\bar{\boldsymbol{G}}_{\hat{E}}\|\|\boldsymbol{E}-\hat{\boldsymbol{E}}\|_F,
\end{aligned}
\tag{44}
$$

where $\xi \geq 1$ is a large enough constant. Since σ is sufficiently large and $\|\boldsymbol{C}\|_\infty$ is encouraged to be small in its update, ξ will not be too large.

We may estimate the Lipschitz constant of $\nabla_{\boldsymbol{E}}\mathcal{L}(\boldsymbol{E})$ as

$$\hat{L}_E^t = \tfrac{2\|\boldsymbol{C}\|_\infty}{\sigma^4}(\sqrt{d}\|\hat{\boldsymbol{X}}-\boldsymbol{E}\|\|\boldsymbol{D}_t\| + \|\boldsymbol{D}_t\|^2) + \tfrac{1}{\sigma^2}\|\bar{\boldsymbol{G}}_{\hat{E}}\|,$$

which however will increase the computational cost because of the spectral norms of $\hat{\boldsymbol{X}}-\boldsymbol{E}$ and $\boldsymbol{D}_t$. For simplicity, we use

$$\hat{L}_E^t = \xi\|\bar{\boldsymbol{G}}_{E_{t-1}}\|/\sigma^2 = \xi\|\boldsymbol{1}_d^\top\boldsymbol{G}_{E_{t-1}}\|_\infty/\sigma^2,$$

where $\xi \geq 1$ should be large enough. Then letting $\tau_E^t > \hat{L}_E^t$, we have

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{E}) \leq \mathcal{L}(\boldsymbol{E}_{t-1}) &+ \langle\boldsymbol{E}-\boldsymbol{E}_{t-1}, \nabla_{\boldsymbol{E}}\mathcal{L}(\boldsymbol{E}_{t-1})\rangle\\
&+\frac{\tau_E^t}{2}\|\boldsymbol{E}-\boldsymbol{E}_{t-1}\|_F^2,
\end{aligned}
\tag{45}
$$

provided that ξ is sufficiently large. According to Table II and the definition of the proximal map [52], we have

$$
\begin{aligned}
\boldsymbol{E}_t \in \min_{\boldsymbol{E}}\ &\langle\boldsymbol{E}-\boldsymbol{E}_{t-1}, \nabla_{\boldsymbol{E}}\mathcal{L}(\boldsymbol{E}_{t-1})\rangle\\
&+\frac{\tau_E^t}{2}\|\boldsymbol{E}-\boldsymbol{E}_{t-1}\|_F^2 + \lambda_E\mathcal{R}(\boldsymbol{E}).
\end{aligned}
\tag{46}
$$

where $\mathcal{R}(\boldsymbol{E}) = \|\boldsymbol{E}\|_F^2$, $\|\boldsymbol{E}\|_1$, or $\|\boldsymbol{E}\|_{2,1}$. By taking $\boldsymbol{E} = \boldsymbol{E}_{t-1}$, it follows from (46) that

$$
\begin{aligned}
&\langle\boldsymbol{E}_t-\boldsymbol{E}_{t-1}, \nabla_{\boldsymbol{E}}\mathcal{L}(\boldsymbol{E}_{t-1})\rangle + \lambda_E\mathcal{R}(\boldsymbol{E}_t)\\
\leq&\lambda_E\mathcal{R}(\boldsymbol{E}_{t-1}) - \frac{\tau_E^t}{2}\|\boldsymbol{E}_t-\boldsymbol{E}_{t-1}\|_F^2.
\end{aligned}
\tag{47}
$$

In addition, the Lipschitz continuity of $\nabla_{\boldsymbol{E}}\mathcal{L}(\boldsymbol{E})$ around $\boldsymbol{E}_{t-1}$ indicates that

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{E}_t) \leq \mathcal{L}(\boldsymbol{E}_{t-1}) &+ \langle\boldsymbol{E}_t-\boldsymbol{E}_{t-1}, \nabla_{\boldsymbol{E}}\mathcal{L}(\boldsymbol{E}_{t-1})\rangle\\
&+\frac{L_E^t}{2}\|\boldsymbol{E}_t-\boldsymbol{E}_{t-1}\|_F^2,
\end{aligned}
\tag{48}
$$

Combining (48) and (47), we have

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{E}_t) + \lambda_E\mathcal{R}(\boldsymbol{E}_t) \leq&\mathcal{L}(\boldsymbol{E}_{t-1}) + \lambda_E\mathcal{R}(\boldsymbol{E}_{t-1})\\
&-\frac{\tau_E^t - L_E^t}{2}\|\boldsymbol{E}_t-\boldsymbol{E}_{t-1}\|_F^2.
\end{aligned}
$$

Let $\tau_E^t = \xi\|\boldsymbol{1}_d^\top\boldsymbol{G}_{E_{t-1}}\|_\infty/\sigma^2$ and ξ be sufficiently large, then $\tau_E^t - L_E^t \geq 0$. This finished the proof. □

J. Proof for Theorem 1

Proof. Combining Lemmas 5, 6, and 7, we have

$$\mathcal{J}(\boldsymbol{D}_t, \boldsymbol{C}_t, \boldsymbol{E}_t) \leq \mathcal{J}(\boldsymbol{D}_{t-1}, \boldsymbol{C}_{t-1}, \boldsymbol{E}_{t-1}) - \Delta_J^t, \tag{49}$$

where $\Delta_J^t = \Delta_{J_C}^t + \Delta_{J_D}^t + \Delta_{J_E}^t$ and

$$
\begin{aligned}
\Delta_{J_C}^t =&\frac{\tau_C^t - L_C^t}{2}\|\boldsymbol{C}_t - \boldsymbol{C}_{t-1}\|_F^2,\\
\Delta_{J_D}^t =&\frac{1}{2\tau_D^t}\mathrm{Tr}\left(\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})(\boldsymbol{H}_{t-1}+\mu\boldsymbol{I})^{-1}\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})^\top\right),\\
\Delta_{J_E}^t =&\frac{\tau_E^t - L_E^t}{2}\|\boldsymbol{E}_t - \boldsymbol{E}_{t-1}\|_F^2.
\end{aligned}
$$

As $\tau_C^t > L_C^t$, $\boldsymbol{H}_{t-1}+\mu\boldsymbol{I}$ is positive definite, and $\tau_E^t > L_E^t$, we have $\Delta_{J_C}^t \geq 0$, $\Delta_{J_D}^t \geq 0$, and $\Delta_{J_E}^t \geq 0$. It follows that

$$\mathcal{J}(\boldsymbol{D}_t, \boldsymbol{C}_t, \boldsymbol{E}_t) \leq \mathcal{J}(\boldsymbol{D}_{t-1}, \boldsymbol{C}_{t-1}, \boldsymbol{E}_{t-1}).$$

Since $\mathcal{J}(\boldsymbol{D},\boldsymbol{C},\boldsymbol{E})$ is bounded below, we have

$$\lim_{t\to\infty}\mathcal{J}(\boldsymbol{D}_t,\boldsymbol{C}_t,\boldsymbol{E}_t)-\mathcal{J}(\boldsymbol{D}_{t-1},\boldsymbol{C}_{t-1},\boldsymbol{E}_{t-1})=0.$$

Summing (49) from 0 to ∞, we have

$$\mathcal{J}(\boldsymbol{D}_0,\boldsymbol{C}_0,\boldsymbol{E}_0)-\mathcal{J}(\boldsymbol{D}_\infty,\boldsymbol{C}_\infty,\boldsymbol{E}_\infty)\geq\sum_{t=0}^{\infty}\Delta_J^t.$$

Hence

$$\sum_{t=0}^{\infty}\Delta_{J_C}^t+\Delta_{J_D}^t+\Delta_{J_E}^t<\infty.$$

As $\Delta_{J_C}^t,\Delta_{J_D}^t,\Delta_{J_E}^t\geq 0$ for all t, we conclude that

$$\lim_{t\to\infty}\|\nabla_{\boldsymbol{D}}\mathcal{L}(\boldsymbol{D}_{t-1})\|_F=0,$$
$$\lim_{t\to\infty}\|\boldsymbol{C}_t-\boldsymbol{C}_{t-1}\|_F=0,$$
$$\lim_{t\to\infty}\|\boldsymbol{E}_t-\boldsymbol{E}_{t-1}\|_F=0.$$

Combining the first equality above with (43) yields

$$\lim_{t\to\infty}\|\boldsymbol{D}_t-\boldsymbol{D}_{t-1}\|_F=0.$$

This finished the proof. □

References

[1] M. Udell and A. Townsend, "Why are big data matrices approximately low rank?" *SIAM Journal on Mathematics of Data Science*, vol. 1, no. 1, pp. 144–160, 2019.

[2] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[3] I. Jolliffe, *Principal Component Analysis*, ser. Springer Series in Statistics. Springer, 2002.

[4] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *J. ACM*, vol. 58, no. 3, pp. 1–37, 2011.

[5] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, pp. 717–772, 2009.

[6] J. Fan and T. W. Chow, "Matrix completion by least-square, low-rank, and sparse self-representations," *Pattern Recognition*, vol. 71, pp. 290 – 305, 2017.

[7] J. Fan, L. Ding, Y. Chen, and M. Udell, "Factor group-sparse regularization for efficient low-rank matrix recovery," in *Advances in Neural Information Processing Systems*, 2019, pp. 5104–5114.

[8] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, April 2006.

[9] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[10] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 171–184, 2013.

[11] J. Fan, Z. Tian, M. Zhao, and T. W. Chow, "Accelerated low-rank representation for subspace clustering and semi-supervised classification on large-scale data," *Neural Networks*, vol. 100, pp. 39 – 48, 2018.

[12] C. You, C. Li, D. P. Robinson, and R. Vidal, "Scalable exemplar-based subspace clustering on class-imbalanced data," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 67–83.

[13] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 689–696.

[14] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach, "Supervised dictionary learning," in *Advances in neural information processing systems*, 2009, pp. 1033–1040.

[15] Z. Zhang, J. Ren, W. Jiang, Z. Zhang, R. Hong, S. Yan, and M. Wang, "Joint subspace recovery and enhanced locality driven robust flexible discriminative dictionary learning," *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[16] Z. Zhang, W. Jiang, Z. Zhang, S. Li, G. Liu, and J. Qin, "Scalable block-diagonal locality-constrained projective dictionary learning," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 4376–4382.

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

[18] L. Zhang, M. Yang, and X. Feng, "Sparse representation or collaborative representation: Which helps face recognition?" in *Computer vision (ICCV), 2011 IEEE international conference on*. IEEE, 2011, pp. 471–478.

[19] C. Lu, J. Shi, and J. Jia, "Online robust dictionary learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 415–422.

[20] W. Jiang, F. Nie, and H. Huang, "Robust dictionary learning with capped ℓ_1-norm," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[21] Z. Li and J. Tang, "Weakly supervised deep matrix factorization for social image understanding," *IEEE Transactions on Image Processing*, vol. 26, no. 1, pp. 276–288, 2016.

[22] Z. Li, J. Tang, and X. He, "Robust structured nonnegative matrix factorization for image representation," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 5, pp. 1947–1960, 2017.

[23] V. M. Patel, N. Hien Van, and R. Vidal, "Latent space sparse and low-rank subspace clustering," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 9, no. 4, pp. 691–701, 2015.

[24] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, "Kernel pca and de-noising in feature spaces," in *Advances in Neural Information Processing Systems 11*. MIT Press, 1999, pp. 536–542.

[25] J.-Y. Kwok and I.-H. Tsang, "The pre-image problem in kernel methods," *IEEE transactions on neural networks*, vol. 15, no. 6, pp. 1517–1525, 2004.

[26] S.-Y. Huang, Y.-R. Yeh, and S. Eguchi, "Robust kernel principal component analysis," *Neural computation*, vol. 21, no. 11, pp. 3179–3213, 2009.

[27] M. H. Nguyen and F. Torre, "Robust kernel principal component analysis," in *Advances in Neural Information Processing Systems*, 2009, pp. 1185–1192.

[28] J. Fan and T. W. S. Chow, "Exactly robust kernel principal component analysis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 3, pp. 749–761, 2020.

[29] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[30] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with bm3d?" in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 2392–2399.

[31] X. Mao, C. Shen, and Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Advances in neural information processing systems*, 2016, pp. 2802–2810.

[32] R. Gao and K. Grauman, "On-demand learning for deep image restoration," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1086–1095.

[33] H. Van Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Design of non-linear kernel dictionaries for object recognition," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 5123–5135, 2013.

[34] M. Harandi and M. Salzmann, "Riemannian coding and dictionary learning: Kernels to the rescue," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 3926–3935.

[35] H. Liu, J. Qin, H. Cheng, and F. Sun, "Robust kernel dictionary learning using a whole sequence convergent algorithm," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[36] Y. Quan, C. Bao, and H. Ji, "Equiangular kernel dictionary learning with applications to dynamic texture analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 308–316.

[37] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[38] K. Skretting and K. Engan, "Recursive least squares dictionary learning algorithm," *IEEE Transactions on Signal Processing*, vol. 58, no. 4, pp. 2121–2130, 2010.

[39] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 4, pp. 791–804, 2011.

[40] Z. Jiang, Z. Lin, and L. S. Davis, "Label consistent K-SVD: Learning a discriminative dictionary for recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2651–2664, Nov 2013.

[41] Z. Chen and Y. Wu, "Robust dictionary learning by error source decomposition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2216–2223.

[42] H. Wang, F. Nie, W. Cai, and H. Huang, "Semi-supervised robust dictionary learning via efficient l-norms minimization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1145–1152.

[43] S. P. Awate and N. N. Koushik, "Robust dictionary learning on the hilbert sphere in kernel feature space," in *Machine Learning and Knowledge Discovery in Databases*, P. Frasconi, N. Landwehr, G. Manco, and J. Vreeken, Eds. Cham: Springer International Publishing, 2016, pp. 731–748.

[44] T. Zhou, F. Liu, H. Bhaskar, and J. Yang, "Robust visual tracking via online discriminative and low-rank dictionary learning," *IEEE Transactions on Cybernetics*, vol. 48, no. 9, pp. 2643–2655, Sep. 2018.

[45] V. M. Patel and R. Vidal, "Kernel sparse subspace clustering," in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct 2014, pp. 2849–2853.

[46] Y.-X. Wang, H. Xu, and C. Leng, "Provable subspace clustering: When lrr meets ssc," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 64–72.

[47] J. Shen and P. Li, "Learning structured low-rank representation via matrix factorization," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 51. Cadiz, Spain: PMLR, 09–11 May 2016, pp. 500–509.

[48] M. Brbić and I. Kopriva, "ℓ_0 -motivated low-rank sparse subspace clustering," *IEEE Transactions on Cybernetics*, vol. 50, no. 4, pp. 1711–1725, April 2020.

[49] J. Fan and M. Udell, "Online high rank matrix completion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8690–8698.

[50] J. Fan, Y. Zhang, and M. Udell, "Polynomial matrix completion for missing data imputation and transductive learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3842–3849.

[51] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[52] N. Parikh, S. Boyd *et al.*, "Proximal algorithms," *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[53] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Mathematical Programming*, vol. 146, no. 1-2, pp. 459–494, 2014.

[54] C.-Y. Lu, H. Min, Z.-Q. Zhao, L. Zhu, D.-S. Huang, and S. Yan, "Robust and efficient subspace segmentation via least squares regression," in *European conference on computer vision*. Springer, 2012, pp. 347–360.

[55] D. Cai and X. Chen, "Large scale spectral clustering via landmark-based sparse representation," *IEEE transactions on cybernetics*, vol. 45, no. 8, pp. 1669–1680, 2014.

[56] S. A. Nene, S. K. Nayar, H. Murase *et al.*, "Columbia object image library (coil-20)," 1996.

[57] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia object image library (coil-100)," 1996.

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

[59] A. M. Martínez and A. C. Kak, "PCA versus LDA," *IEEE transactions on pattern analysis and machine intelligence*, vol. 23, no. 2, pp. 228–233, 2001.

[60] J. Lipor, D. Hong, Y. S. Tan, and L. Balzano, "Subspace clustering using ensembles of k-subspaces," *arXiv preprint arXiv:1709.04744*, 2017.

Jicong Fan received his B.E and M.E degrees in Automation and Control Science & Engineering, from Beijing University of Chemical Technology, Beijing, P.R., China, in 2010 and 2013, respectively. From 2013 to 2015, he was a research assistant at the University of Hong Kong. He received his Ph.D. degree in Electronic Engineering, from City University of Hong Kong, Hong Kong S.A.R. in 2018. From 2018.01 to 2018.06, he was a visiting scholar at the Department of Electrical and Computer Engineering, University of Wisconsin-Madison, USA. From 2018.10 to 2020.07, he was a Postdoc Associate at the School of Operations Research and Information Engineering, Cornell University, Ithaca, USA. Currently, he is a Research Assistant Professor at the School of Data Science, The Chinese University of Hong Kong (Shenzhen) and Shenzhen Research Institute of Big Data, Shenzhen, China. His research interests include statistical process control, signal processing, computer vision, optimization, and machine learning.

Chengrun Yang received his BS degree in Physics from Fudan University, Shanghai, China in 2016. Currently, he is a PhD student at the School of Electrical and Computer Engineering, Cornell University. His research interests include the application of low dimensional structures and active learning in resource-constrained learning problems.

Madeleine Udell Madeleine Udell is Assistant Professor of Operations Research and Information Engineering and Richard and Sybil Smith Sesquicentennial Fellow at Cornell University. She studies optimization and machine learning for large scale data analysis and control, with applications in marketing, demographic modeling, medical informatics, engineering system design, and automated machine learning. Her work has been recognized by an NSF CAREER award, an Office of Naval Research (ONR) Young Investigator Award, and an INFORMS Optimization Society Best Student Paper Award (as advisor). Madeleine completed her PhD at Stanford University in Computational & Mathematical Engineering in 2015 under the supervision of Stephen Boyd, and a one year postdoctoral fellowship at Caltech in the Center for the Mathematics of Information hosted by Professor Joel Tropp.