# Learning the nonlinear geometry of high-dimensional data: Models and algorithms

Tong Wu, *Student Member, IEEE,* and Waheed U. Bajwa, *Senior Member, IEEE*

***Abstract*—Modern information processing relies on the axiom that high-dimensional data lie near low-dimensional geometric structures. This paper revisits the problem of data-driven learning of these geometric structures and puts forth two new nonlinear geometric models for data describing "related" objects/phenomena. The first one of these models straddles the two extremes of the subspace model and the union-of-subspaces model, and is termed the *metric-constrained union-of-subspaces* (MC-UoS) model. The second one of these models—suited for data drawn from a mixture of nonlinear manifolds—generalizes the kernel subspace model, and is termed the *metric-constrained kernel union-of-subspaces* (MC-KUoS) model. The main contributions of this paper in this regard include the following. First, it motivates and formalizes the problems of MC-UoS and MC-KUoS learning. Second, it presents algorithms that efficiently learn an MC-UoS or an MC-KUoS underlying data of interest. Third, it extends these algorithms to the case when parts of the data are missing. Last, but not least, it reports the outcomes of a series of numerical experiments involving both synthetic and real data that demonstrate the superiority of the proposed geometric models and learning algorithms over existing approaches in the literature. These experiments also help clarify the connections between this work and the literature on (subspace and kernel $k$-means) clustering.**

***Index Terms*—Data-driven learning, kernel methods, kernel $k$-means, missing data, principal component analysis, subspace clustering, subspace learning, union of subspaces.**

## I. Introduction

We have witnessed an explosion in data generation in the last decade or so. Modern signal processing, machine learning and statistics have been relying on a fundamental maxim of information processing to cope with this data explosion. This maxim states that while real-world data might lie in a high-dimensional Hilbert space, relevant information within them almost always lies near low-dimensional geometric structures embedded in the Hilbert space. Knowledge of these low-dimensional geometric structures not only improves the performance of many processing tasks, but it also helps reduce computational and communication costs, storage requirements, etc.

This work is supported in part by the NSF under grants CCF-1218942 and CCF-1453073, by the Army Research Office under grant W911NF-14-1-0295, and by an Army Research Lab Robotics CTA subaward. Preliminary versions of parts of this work have been presented at the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2014) [1], IEEE Workshop on Statistical Signal Processing (SSP 2014) [2], and IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2015) [3].

The authors are with the Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ 08854, USA (E-mails: {tong.wu.ee, waheed.bajwa}@rutgers.edu).

Information processing literature includes many models for geometry of high-dimensional data, which are then utilized for better performance in numerous applications, such as dimensionality reduction and data compression [4]–[8], denoising [9], [10], classification [11]–[14], and motion segmentation [15], [16]. These geometric models broadly fall into two categories, namely, linear models [4], [12], [17] and nonlinear models [6], [16], [18]–[20]. A further distinction can be made within each of these two categories depending upon whether the models are prespecified [21], [22] or learned from the data themselves [10], [16], [19], [23]–[25]. Our focus in this paper is on the latter case, since data-driven learning of geometric models is known to outperform prespecified geometric models [10], [26].

Linear models, which dictate that data lie near a low-dimensional subspace of the Hilbert space, have been historically preferred within the class of data-driven models due to their simplicity. These models are commonly studied under the rubrics of *principal component analysis* (PCA) [4], [27], Karhunen–Loève transform [28], factor analysis [17], etc. But real-world data in many applications tend to be nonlinear. In order to better capture the geometry of data in such applications, a few nonlinear generalizations of data-driven linear models that remain computationally feasible have been investigated in the last two decades. One of the most popular generalizations is the nonlinear manifold model [6], [8], [29], [30]. The (nonlinear) manifold model can also be considered as the *kernel subspace* model, which dictates that a mapping of the data to a higher- (possibly infinite-) dimensional Hilbert space lies near a low-dimensional subspace [31]. Data-driven learning of geometric models in this case is commonly studied under the moniker of *kernel PCA* (KPCA) [29]. Another one of the most popular generalizations of linear models is the *union-of-subspaces* (UoS) (resp., *union-of-affine-subspaces* (UoAS)) model, which dictates that data lie near a mixture of low-dimensional subspaces (resp., affine subspaces) in the ambient Hilbert space. Data-driven learning of the UoS model is commonly carried out under the rubrics of generalized PCA [32], dictionary learning [19], [33], and subspace clustering [16], [34]–[37]. On the other hand, data-driven learning of the UoAS model is often studied under the umbrella of hybrid linear modeling [38], mixture of factor analyzers [39], etc.

In the literature, encouraging results have been reported for both the UoS and the kernel subspace models in the context of a number of applications [9], [14], [16], [40]. But there remains a lot of room for improvement in both these models. The canonical UoS model, for example, does not impose any constraint on the collection of subspaces underlying data of

interest. On the other hand, one can intuit that subspaces describing "similar" data should have some "relation" on the Grassmann manifold. The lack of any a priori constraint during learning on the subspaces describing "similar" data has the potential to make different methods for UoS learning susceptible to errors due to low *signal-to-noise ratio* (SNR), outliers, missing data, etc. Another limitation of the UoS model is the individual linearity of its constituent subspaces, which limits its usefulness for data drawn from a nonlinear manifold [29]. On the other hand, while the kernel subspace model can handle manifold data, a single kernel subspace requires a large dimension to capture the richness of data drawn from a mixture of nonlinear manifolds.

Our goal in this paper is to improve the state-of-the-art data-driven learning of geometric data models for both complete and missing data describing similar phenomenon. We are in particular interested in learning models for data that are either mildly or highly nonlinear. Here, we are informally using the terms "mildly nonlinear" and "highly nonlinear." Heuristically, nonlinear data that cannot be represented through a mixture of linear components should be deemed "highly nonlinear." Our key objective in this regard is overcoming the aforementioned limitations of the UoS model and the kernel subspace model for mildly nonlinear data and highly nonlinear data, respectively.

### A. Our Contributions and Relation to Other Work

One of our main contributions is introduction of a novel geometric model, termed *metric-constrained union-of-subspaces* (MC-UoS) model, for mildly nonlinear data describing similar phenomenon. Similar to the canonical UoS model, the MC-UoS model also dictates that data lie near a union of low-dimensional subspaces in the ambient space. But the key distinguishing feature of the MC-UoS model is that it also forces its constituent subspaces to be close to each other according to a metric defined on the Grassmann manifold. In this paper, we formulate the MC-UoS learning problem for a particular choice of the metric and derive three novel iterative algorithms for solving this problem. The first one of these algorithms operates on complete data, the second one deals with the case of unknown number and dimension of subspaces, while the third one carries out MC-UoS learning in the presence of missing data.

One of our other main contributions is extension of our MC-UoS model for highly nonlinear data. This model, which can also be considered a generalization of the kernel subspace model, is termed *metric-constrained kernel union-of-subspaces* (MC-KUoS) model. The MC-KUoS model asserts that mapping of data describing similar phenomenon to some higher-dimensional Hilbert space (also known as the *feature space*) lies near a mixture of subspaces in the feature space with the additional constraint that the individual subspaces are also close to each other in the feature space. In this regard, we formulate the MC-KUoS learning problem using the *kernel trick* [18], which avoids explicit mapping of data to the feature space. In addition, we derive two novel iterative algorithms that can carry out MC-KUoS learning in the presence of complete data and missing data.

Our final contribution involves carrying out a series of numerical experiments on both synthetic and real data to justify our heuristics for the two models introduced in this paper. Our main focus in these experiments is learning the geometry of (training) data describing similar phenomenon in the presence of additive, white Gaussian noise and missing entries. In the case of real data, we demonstrate the superiority of the proposed algorithms by focusing on the tasks of denoising of (test) data and clustering of data having either complete or missing entries. (Other applications of our models will be investigated in future works.) Our results confirm the superiority of our models in comparison to a number of state-of-the-art approaches under both the UoS and the kernel subspace models [16], [25], [29], [33], [36], [37], [41].

We conclude this discussion by pointing out that our work is not only related to the traditional literature on geometry learning, but it also has connections to the literature on clustering [16], [36], [37], [41]. Specifically, the mixture components within our two models can be treated as different clusters within the data and the outputs of our algorithms automatically lead us to these clusters. Alternatively, one could approach the MC-UoS/MC-KUoS learning problem by first clustering the data and then learning the individual subspaces in the ambient/feature space. However, numerical experiments confirm that our algorithms perform better than such heuristic approaches.

### B. Notation and Organization

Throughout the paper, we use bold lower-case and bold upper-case letters to represent vectors/sets and matrices, respectively. The $i$-th element of a vector/set $\mathbf{v}$ is denoted by $\mathbf{v}_{(i)}$, while $a_{i,j}$ denotes the $(i,j)$-th element of a matrix $\mathbf{A}$. The $m$-dimensional zero vector is denoted by $\mathbf{0}_m$ and the $m \times m$ identity matrix is denoted by $\mathbf{I}_m$. Given a set $\mathbf{\Omega}$, $[\mathbf{A}]_{\mathbf{\Omega},:}$ (resp., $[\mathbf{v}]_{\mathbf{\Omega}}$) denotes the submatrix of $\mathbf{A}$ (resp., subvector of $\mathbf{v}$) corresponding to the rows of $\mathbf{A}$ (resp., entries of $\mathbf{v}$) indexed by $\mathbf{\Omega}$. Given two sets $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$, $[\mathbf{A}]_{\mathbf{\Omega}_1,\mathbf{\Omega}_2}$ denotes the submatrix of $\mathbf{A}$ corresponding to the rows and columns indexed by $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$, respectively. Finally, $(\cdot)^T$ and $\mathrm{tr}(\cdot)$ denote transpose and trace operations, respectively, while the Frobenius norm of a matrix $\mathbf{A}$ is denoted by $\|\mathbf{A}\|_F$ and the $\ell_2$ norm of a vector $\mathbf{v}$ is represented by $\|\mathbf{v}\|_2$.

The rest of the paper is organized as follows. In Sec. II, we formally define the metric-constrained union-of-subspaces (MC-UoS) model and mathematically formulate the data-driven learning problems studied in this paper. Sec. III presents algorithms for MC-UoS learning in the presence of complete and missing data. Sec. IV gives the details of two algorithms for learning of an MC-UoS in the feature space, corresponding to the cases of complete and missing data. We then present some numerical results in Sec. V, which is followed by concluding remarks in Sec. VI.

## II. Problem Formulation

In this section, we mathematically formulate the two problems of learning the geometry of mildly and highly nonlinear data from training examples. Both of our problems rely on the

notion of a metric-constrained union-of-subspaces (MC-UoS), one in the ambient space and the other in the feature space. We therefore first begin with a mathematical characterization of the MC-UoS model.

Recall that the canonical UoS model asserts data in an $m$-dimensional ambient space can be represented through a union of $L$ low-dimensional subspaces [8], [42]: $\mathcal{M}_L = \bigcup_{\ell=1}^{L} \mathcal{S}_\ell$, where $\mathcal{S}_\ell$ is a subspace of $\mathbb{R}^m$. In here, we make the simplified assumption that all subspaces in $\mathcal{M}_L$ have the same dimension, i.e., $\forall \ell$, $\dim(\mathcal{S}_\ell) = s \ll m$. In this case, each subspace $\mathcal{S}_\ell$ corresponds to a point on the Grassmann manifold $\mathcal{G}_{m,s}$, which denotes the set of all $s$-dimensional subspaces of $\mathbb{R}^m$. While the canonical UoS model allows $\mathcal{S}_\ell$'s to be arbitrary points on $\mathcal{G}_{m,s}$, the basic premise of the MC-UoS model is that subspaces underlying *similar* signals likely form a "cluster" on the Grassmann manifold. In order to formally capture this intuition, we make use of a distance metric on $\mathcal{G}_{m,s}$ and define an MC-UoS according to that metric as follows.

**Definition 1. (Metric-Constrained Union-of-Subspaces.)** A UoS $\mathcal{M}_L = \bigcup_{\ell=1}^{L} \mathcal{S}_\ell$ is said to be constrained with respect to a metric $d_u : \mathcal{G}_{m,s} \times \mathcal{G}_{m,s} \to [0, \infty)$ if $\max_{\ell,p:\ell\neq p} d_u(\mathcal{S}_\ell, \mathcal{S}_p) \leq \epsilon$ for some positive constant $\epsilon$.

The metric we use in this paper to measure distances between subspaces is based on the Hausdorff distance between a vector and a subspace, which was first defined in [43]. Specifically, if $\mathbf{D}_\ell \in \mathbb{R}^{m\times s}$ and $\mathbf{D}_p \in \mathbb{R}^{m\times s}$ denote orthonormal bases of subspaces $\mathcal{S}_\ell$ and $\mathcal{S}_p$, respectively, then

$$\begin{aligned} d_u(\mathcal{S}_\ell, \mathcal{S}_p) &= \sqrt{s - \operatorname{tr}(\mathbf{D}_\ell^T \mathbf{D}_p \mathbf{D}_p^T \mathbf{D}_\ell)} \\ &= \|\mathbf{D}_\ell - P_{\mathcal{S}_p} \mathbf{D}_\ell\|_F, \end{aligned} \tag{1}$$

where $P_{\mathcal{S}_p}$ denotes the projection operator onto the subspace $\mathcal{S}_p$: $P_{\mathcal{S}_p} = \mathbf{D}_p \mathbf{D}_p^T$. It is easy to convince oneself that $d_u(\cdot,\cdot)$ in (1) is invariant to the choice of orthonormal bases of the two subspaces, while it was formally shown to be a metric on $\mathcal{G}_{m,s}$ in [44]. Note that $d_u(\cdot,\cdot)$ in (1) is directly related to the concept of *principal angles* between two subspaces. Given two subspaces $\mathcal{S}_\ell, \mathcal{S}_p$ and their orthonormal bases $\mathbf{D}_\ell, \mathbf{D}_p$, the cosines of the principal angles $\cos(\theta_{\ell,p}^j)$, $j = 1, \ldots, s$, between $\mathcal{S}_\ell$ and $\mathcal{S}_p$ are defined as the ordered singular values of $\mathbf{D}_\ell^T \mathbf{D}_p$ [37]. It therefore follows that $d_u(\mathcal{S}_\ell, \mathcal{S}_p) = \sqrt{s - \sum_{j=1}^{s} \cos^2(\theta_{\ell,p}^j)}$. We conclude our discussion of the MC-UoS model by noting that other definitions of metrics on the Grassmann manifold exist in the literature that are based on different manipulations of $\cos(\theta_{\ell,p}^j)$'s [45]. In this paper, however, we focus only on (1) due to its ease of computation.

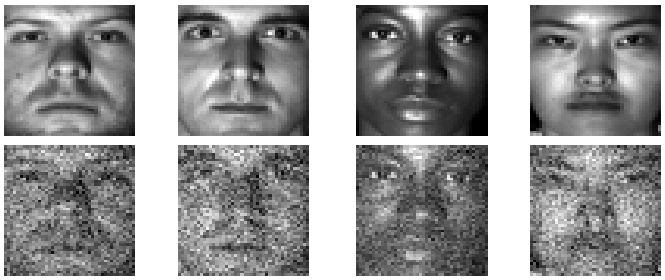
### A. Geometry Learning for Mildly Nonlinear Data

Our first geometry learning problem corresponds to the case of high-dimensional data that lie near an MC-UoS $\mathcal{M}_L$ in the ambient space $\mathbb{R}^m$. We are using the qualifier "mildly nonlinear" for such data since individual components of these data are being modeled in a linear fashion. In terms of a formal characterization, we assume access to a collection of $N$ noisy training samples, $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N] \in \mathbb{R}^{m\times N}$, such that every sample $\mathbf{y}_i$ can be expressed as $\mathbf{y}_i = \mathbf{x}_i + \boldsymbol{\xi}_i$ with $\mathbf{x}_i$ belonging to one of the $\mathcal{S}_\ell$'s in $\mathcal{M}_L$ and $\boldsymbol{\xi}_i \sim \mathcal{N}(\mathbf{0}, (\sigma_{tr}^2/m)\mathbf{I}_m)$ denoting additive noise. We assume without loss of generality throughout this paper that $\|\mathbf{x}_i\|_2^2 = 1$, which results in training SNR of $\|\mathbf{x}_i\|_2^2 / \mathbb{E}[\|\boldsymbol{\xi}_i\|_2^2] = \sigma_{tr}^{-2}$. To begin, we assume both $L$ and $s$ are known a priori. Later, we relax this assumption and extend our work in Sec. III-B to the case when these two parameters are unknown. Our goal is to learn $\mathcal{M}_L$ using the training data $\mathbf{Y}$, which is equivalent to learning a collection of $L$ subspaces that not only approximate the training data, but are also "close" to each other on the Grassmann manifold (cf. Definition 1). Here, we pose this goal of learning an MC-UoS $\mathcal{M}_L$ in terms of the following optimization program:

$$\begin{aligned} \{\mathcal{S}_\ell\}_{\ell=1}^{L} = \underset{\{\mathcal{S}_\ell\}\subset\mathcal{G}_{m,s}}{\arg\min} & \sum_{\substack{\ell,p=1 \\ \ell\neq p}}^{L} d_u^2(\mathcal{S}_\ell, \mathcal{S}_p) \\ & + \lambda \sum_{i=1}^{N} \|\mathbf{y}_i - P_{\mathcal{S}_{l_i}} \mathbf{y}_i\|_2^2, \end{aligned} \tag{2}$$

where $l_i = \arg\min_\ell \|\mathbf{y}_i - P_{\mathcal{S}_\ell}\mathbf{y}_i\|_2^2$ with $P_{\mathcal{S}_\ell}\mathbf{y}_i$ denoting the (orthogonal) projection of $\mathbf{y}_i$ onto the subspace $\mathcal{S}_\ell$. Notice that the first term in (2) forces the learned subspaces to be close to each other, while the second term requires them to simultaneously provide good approximations to the training data. The tuning parameter $\lambda > 0$ in this setup provides a compromise between subspace closeness and approximation error. While a discussion of finding an optimal $\lambda$ is beyond the scope of this paper, cross validation can be used to find ranges of good values of tuning parameters in such problems [46] (also, see Sec. V-A). It is worth pointing out here that (2) can be reformulated for the UoAS model through a simple extension of the metric defined in (1). In addition, note that (2) is mathematically similar to a related problem studied in the clustering literature [47]. In fact, it is straightforward to show that (2) reduces to the clustering problem in [47] for $\mathcal{M}_L$ being a union of zero-dimensional affine subspaces.

*Remark* 1. The MC-UoS model and the learning problem (2) can be further motivated as follows. Consider a set of facial images of individuals under varying illumination conditions in the Extended Yale B dataset [48], as in Figs. 1(a) and 1(b). It is generally agreed that all images of an individual in this case can be regarded as lying near a 9-dimensional subspace [49], which can be computed in a straightforward manner using singular value decomposition (SVD). The subspace distance defined in (1) can be used in this case to identify similar-looking individuals. Given noisy training images of such "similar" individuals, traditional methods for UoS learning such as *sparse subspace clustering* (SSC) [16] that rely only on the approximation error will be prone to errors. Fig. 1 provides a numerical validation of this claim, where it is shown that SSC has good performance on noisy images of different-looking individuals (cf. Fig. 1(b)), but its performance degrades in the case of similar-looking individuals (cf. Fig. 1(a)). The MC-UoS learning problem (2), on the other hand, should be able to handle both cases reliably because of the first term in (2) that penalizes subspaces that do not cluster on the Grassmann manifold. We refer the reader to Sec. V-A for



(a) (b)

Fig. 1. An example illustrating the limitations of existing methods for UoS learning from noisy training data. The top row in this figure shows examples of "clean" facial images of four individuals in the Extended Yale B dataset [48], while the bottom row shows noisy versions of these images, corresponding to $\sigma_{tr}^2 = 0.1$. The "ground truth" distance between the subspaces of the individuals in (a) is $1.7953$, while it is $2.3664$ between the subspaces of the individuals in (b). State-of-the-art UoS learning methods have trouble reliably learning the underlying subspaces whenever the subspaces are close to each other. Indeed, while the distance between the two subspaces learned by the SSC algorithm [16] from noisy images of the individuals in (b) is $2.4103$, it is $2.4537$ for the case of "similar-looking" individuals in (a).

detailed experiments that numerically validate this claim.

In this paper, we study two variants of the MC-UoS learning problem described by (2). In the first variant, all $m$ dimensions of each training sample in $\mathbf{Y}$ are observed and the geometry learning problem is exactly given by (2). In the second variant, it is assumed that some of the $m$ dimensions of each training sample in $\mathbf{Y}$ are unobserved (i.e., missing), which then requires a recharacterization of (2) for the learning problem to be well posed. We defer that recharacterization to Sec. III-C of the paper. In order to quantify the performance of our learning algorithms, we will resort to generation of noisy test data as follows. Given noiseless (synthetic or real) data sample $\mathbf{x}$ with $\|\mathbf{x}\|_2^2 = 1$, noisy test sample $\mathbf{z}$ is given by $\mathbf{z} = \mathbf{x} + \boldsymbol{\xi}$ with the additive noise $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, (\sigma_{te}^2/m)\mathbf{I}_m)$. We will then report the metric of *average approximation error of noisy test data* using the learned subspaces for synthetic and real data. Finally, in the case of synthetic data drawn from an MC-UoS, we will also measure the performance of our algorithms in terms of *average normalized subspace distances* between the learned and the true subspaces. We defer a formal description of both these metrics to Sec. V-A1, which describes in detail the setup of our experiments.

### B. Geometry Learning for Highly Nonlinear Data

Our second geometry learning problem corresponds to the case of high-dimensional data drawn from a mixture of nonlinear manifolds in the ambient space $\mathbb{R}^m$. The basic premise of our model in this case is that when data drawn from a mixture of nonlinear manifolds are mapped through a nonlinear map $\phi : \mathbb{R}^m \to \mathcal{F}$ to a higher-dimensional feature space $\mathcal{F} \subset \mathbb{R}^{\widetilde{m}}$ with $\widetilde{m} \gg m$, then the $\phi$-mapped "images" of these data can be modeled as lying near an MC-UoS $\mathcal{M}_L$ in the feature space. In order to learn this model, we once again assume access to a collection of $N$ training samples, $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N] \in \mathbb{R}^{m \times N}$, with the fundamental difference here being that the *mapped training data* $\phi(\mathbf{Y}) = [\phi(\mathbf{y}_1), \ldots, \phi(\mathbf{y}_N)]$ are now assumed to be drawn from an MC-UoS $\mathcal{M}_L = \bigcup_{\ell=1}^L \mathcal{S}_\ell \subset \mathcal{G}_{\widetilde{m},s} \subset \mathcal{F}$. Here, we also make the simplified assumption that $\operatorname{rank}(\phi(\mathbf{Y})) = N$, which is justified as long as $\widetilde{m} \gg N$ and no two training samples are identical. Our goal in this setting is to learn the (feature space) MC-UoS $\mathcal{M}_L$ using the training data $\mathbf{Y}$, which in theory can still be achieved by solving the following variant of (2):

$$\{\mathcal{S}_\ell\}_{\ell=1}^L = \underset{\{\mathcal{S}_\ell\} \subset \mathcal{G}_{\widetilde{m},s}}{\arg\min} \sum_{\substack{\ell,p=1 \\ \ell \neq p}}^{L} d_u^2(\mathcal{S}_\ell, \mathcal{S}_p) + \lambda \sum_{i=1}^{N} \|\phi(\mathbf{y}_i) - P_{\mathcal{S}_{l_i}} \phi(\mathbf{y}_i)\|_2^2, \quad (3)$$

where $l_i = \arg\min_\ell \|\phi(\mathbf{y}_i) - P_{\mathcal{S}_\ell}\phi(\mathbf{y}_i)\|_2^2$ with $P_{\mathcal{S}_\ell}\phi(\mathbf{y}_i)$ denoting the (orthogonal) projection of $\phi(\mathbf{y}_i)$ onto the $s$-dimensional subspace $\mathcal{S}_\ell$ in $\mathbb{R}^{\widetilde{m}}$.

In practice, however, solving (3) directly is likely to be computationally intractable due to the extremely high dimensionality of the feature space. Instead, we are interested in solving the problem of MC-UoS learning in the feature space using the "kernel trick" [18], which involves transforming (3) into a learning problem that only requires evaluations of inner products in $\mathcal{F}$. Such a transformation can then be followed with the use of a Mercer kernel $\kappa$, which is a positive semidefinite function $\kappa : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ that satisfies $\kappa(\mathbf{y}, \mathbf{y}') = \langle \phi(\mathbf{y}), \phi(\mathbf{y}') \rangle$ for all $\mathbf{y}, \mathbf{y}' \in \mathbb{R}^m$, to develop algorithms that can learn an MC-UoS in the feature space without explicit mapping of the training data to the feature space. We term the learning of an MC-UoS in the feature space using the kernel trick as metric-constrained kernel union-of-subspaces (MC-KUoS) learning. Similar to the case of MC-UoS learning, we consider two scenarios in this paper for MC-KUoS learning. The first one of these scenarios corresponds to the standard setup in which all $m$ dimensions of each training sample in $\mathbf{Y}$ are observed, while the second scenario corresponds to the case of "missing data" in which some dimensions of each training sample in $\mathbf{Y}$ remain unobserved. Finally, we will evaluate the proposed MC-KUoS learning algorithms using $(i)$ the metric of average approximation error of noisy test data, and $(ii)$ their clustering performance on training data having either complete or missing entries. We conclude here by pointing out that MC-KUoS learning invariably also leads us to the problem of finding the "pre-images" of data in the feature space induced by our chosen kernel (e.g., Gaussian or polynomial kernel) [9], [50], which will also be addressed in this paper.

*Remark* 2. It is worth noting here that (3) requires knowledge of the nonlinear map $\phi$. However, since we rely on the kernel trick for our MC-KUoS learning framework, we only need access to an appropriate kernel $\kappa$. It is assumed in this paper that such a kernel is readily available to us. While learning the "best" kernel from training data is an interesting extension of our work, it is beyond the scope of this paper.

## III. MC-UoS Learning for Mildly Nonlinear Data

In this section, we describe our approach to the problem of MC-UoS learning for mildly nonlinear data. We begin our

discussion for the case when all $m$ dimensions of each training sample are available to us.

### A. MC-UoS Learning Using Complete Data

In order to reduce the effects of noisy training data, we begin with a pre-processing step that centers the data matrix $\mathbf{Y}$.[1] This involves defining the mean of the samples in $\mathbf{Y}$ as $\bar{\mathbf{y}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{y}_i$ and then subtracting this mean from $\mathbf{Y}$ to obtain the centered data $\widetilde{\mathbf{Y}} = [\widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_N]$, where $\widetilde{\mathbf{y}}_i = \mathbf{y}_i - \bar{\mathbf{y}}$, $i = 1, \ldots, N$. Next, we focus on simplification of the optimization problem (2). To this end, we first define an $L \times N$ indicator matrix $\mathbf{W}$ that identifies memberships of the $\widetilde{\mathbf{y}}_i$'s in different subspaces, where $w_{\ell,i} = 1$, $\ell = 1, \ldots, L$, $i = 1, \ldots, N$, if and only if $\widetilde{\mathbf{y}}_i$ is "closest" to subspace $\mathcal{S}_\ell$; otherwise, $w_{\ell,i} = 0$. Mathematically,

$$\mathbf{W} = \big[w_{\ell,i} \in \{0,1\} : \forall i = 1, \ldots, N, \sum_{\ell=1}^{L} w_{\ell,i} = 1\big]. \quad (4)$$

Further, notice that $\|\mathbf{y}_i - P_{\mathcal{S}_\ell}\mathbf{y}_i\|_2^2$ in (2) can be rewritten as

$$\|\mathbf{y}_i - P_{\mathcal{S}_\ell}\mathbf{y}_i\|_2^2 = \|\widetilde{\mathbf{y}}_i - P_{\mathcal{S}_\ell}\widetilde{\mathbf{y}}_i\|_2^2 = \|\widetilde{\mathbf{y}}_i\|_2^2 - \|\mathbf{D}_\ell^T\widetilde{\mathbf{y}}_i\|_2^2, \quad (5)$$

where $\mathbf{D}_\ell \in \mathbb{R}^{m\times s}$ denotes an (arbitrary) orthonormal basis of $\mathcal{S}_\ell$. Therefore, defining $\mathbf{D} = [\mathbf{D}_1, \ldots, \mathbf{D}_L]$ to be a collection of orthonormal bases of $\mathcal{S}_\ell$'s, we can rewrite (2) as $(\mathbf{D}, \mathbf{W}) = \arg\min_{\mathbf{D},\mathbf{W}} F_1(\mathbf{D}, \mathbf{W})$ with the objective function $F_1(\mathbf{D}, \mathbf{W})$ given by[2]

$$F_1(\mathbf{D}, \mathbf{W}) = \sum_{\substack{\ell,p=1 \\ \ell\neq p}}^{L} \|\mathbf{D}_\ell - P_{\mathcal{S}_p}\mathbf{D}_\ell\|_F^2 + \lambda\sum_{i=1}^{N}\sum_{\ell=1}^{L} w_{\ell,i}(\|\widetilde{\mathbf{y}}_i\|_2^2 - \|\mathbf{D}_\ell^T\widetilde{\mathbf{y}}_i\|_2^2). \quad (6)$$

Minimizing (6) simultaneously over $\mathbf{D}$ and $\mathbf{W}$ is challenging and is likely to be computationally infeasible. Instead, we adopt an alternate minimization approach [51], [52], which involves iteratively solving (6) by alternating between the following two steps: $(i)$ minimizing $F_1(\mathbf{D}, \mathbf{W})$ over $\mathbf{W}$ for a fixed $\mathbf{D}$, which we term as the *subspace assignment* step; and $(ii)$ minimizing $F_1(\mathbf{D}, \mathbf{W})$ over $\mathbf{D}$ for a fixed $\mathbf{W}$, which we term as the *subspace update* stage. To begin this alternate minimization, we start with an initial $\mathbf{D}$ in which each block $\mathbf{D}_\ell \in \mathbb{R}^{m\times s}$ is a random orthonormal basis. Next, we fix this $\mathbf{D}$ and carry out subspace assignment, which now amounts to solving for each $i = 1, \ldots, N$,

$$l_i = \arg\min_{\ell=1,\ldots,L} \|\widetilde{\mathbf{y}}_i - P_{\mathcal{S}_\ell}\widetilde{\mathbf{y}}_i\|_2^2 = \arg\max_{\ell=1,\ldots,L} \|\mathbf{D}_\ell^T\widetilde{\mathbf{y}}_i\|_2^2, \quad (7)$$

and then setting $w_{l_i,i} = 1$ and $w_{\ell,i} = 0$ $\forall \ell \neq l_i$. In order to move to the subspace update step, we fix the matrix $\mathbf{W}$ and focus on optimizing $F_1(\mathbf{D}, \mathbf{W})$ over $\mathbf{D}$. However, this step requires more attention since minimizing over the entire $\mathbf{D}$ at once will also lead to a large-scale optimization problem.

[1]While such pre-processing is common in many geometry learning algorithms, it is not central to our framework.

[2]Note that the minimization here is being carried out under the assumption of $\mathbf{D}_\ell$'s being orthonormal and $\mathbf{W}$ being described by (4).

**Algorithm 1:** Metric-Constrained Union-of-Subspaces Learning (MiCUSaL)

**Input:** Training data $\mathbf{Y} \in \mathbb{R}^{m\times N}$, number of subspaces $L$, dimension of subspaces $s$, and parameter $\lambda$.
**Initialize:** Random orthonormal bases $\{\mathbf{D}_\ell \in \mathbb{R}^{m\times s}\}_{\ell=1}^{L}$.

```
ȳ ← (1/N) Σ_{i=1}^{N} y_i,  ỹ_i ← y_i − ȳ,  i = 1, ..., N.
while stopping rule do
  for i = 1 to N (Subspace Assignment) do
    l_i ← arg max_ℓ ‖D_ℓ^T ỹ_i‖_2.
    w_{l_i,i} ← 1 and ∀ℓ ≠ l_i,  w_{ℓ,i} ← 0.
  end for
  for ℓ = 1 to L (Subspace Update) do
    c_ℓ ← {i ∈ {1, ..., N} : w_{ℓ,i} = 1}.
    Ỹ_ℓ ← [ỹ_i : i ∈ c_ℓ].
    A_ℓ ← Σ_{p≠ℓ} D_p D_p^T + (λ/2) Ỹ_ℓ Ỹ_ℓ^T.
    Eigen decomposition of A_ℓ = U_ℓ Σ_ℓ U_ℓ^T.
    D_ℓ ← Columns of U_ℓ corresponding to the
    s-largest diagonal elements in Σ_ℓ.
  end for
end while
```

**Output:** Orthonormal bases $\{\mathbf{D}_\ell \in \mathbb{R}^{m\times s}\}_{\ell=1}^{L}$.

We address this problem by once again resorting to block-coordinate descent (BCD) [51] and updating only one $\mathbf{D}_\ell$ at a time while keeping the other $\mathbf{D}_p$'s ($p \neq \ell$) fixed in (6). In this regard, suppose we are in the process of updating $\mathbf{D}_\ell$ for a fixed $\ell$ during the subspace update step. Define $\mathbf{c}_\ell = \{i \in \{1, \ldots, N\} : w_{\ell,i} = 1\}$ to be the set containing the indices of all $\widetilde{\mathbf{y}}_i$'s that are assigned to $\mathcal{S}_\ell$ (equivalently, $\mathbf{D}_\ell$) and let $\widetilde{\mathbf{Y}}_\ell = [\widetilde{\mathbf{y}}_i : i \in \mathbf{c}_\ell]$ be the corresponding $m \times |\mathbf{c}_\ell|$ matrix. Then it can be shown after some manipulations of (6) that updating $\mathbf{D}_\ell$ is equivalent to solving the following problem:

$$\begin{aligned}\mathbf{D}_\ell &= \arg\min_{\mathbf{Q}\in\mathcal{V}_{m,s}} \sum_{p\neq\ell} \|\mathbf{Q} - P_{\mathcal{S}_p}\mathbf{Q}\|_F^2 + \frac{\lambda}{2}(\|\widetilde{\mathbf{Y}}_\ell\|_F^2 - \|\mathbf{Q}^T\widetilde{\mathbf{Y}}_\ell\|_F^2) \\ &= \arg\max_{\mathbf{Q}\in\mathcal{V}_{m,s}} \operatorname{tr}\Big(\mathbf{Q}^T(\sum_{p\neq\ell}\mathbf{D}_p\mathbf{D}_p^T + \frac{\lambda}{2}\widetilde{\mathbf{Y}}_\ell\widetilde{\mathbf{Y}}_\ell^T)\mathbf{Q}\Big), \quad (8)\end{aligned}$$

where $\mathcal{V}_{m,s}$ denotes the Stiefel manifold, defined as the collection of all $m \times s$ orthonormal matrices. Note that (8) has an intuitive interpretation. When $\lambda = 0$, (8) reduces to the problem of finding a subspace that is closest to the remaining $L-1$ subspaces in our collection. When $\lambda = \infty$, (8) reduces to the PCA problem, in which case the learning problem (2) reduces to the subspace clustering problem studied in [41]. By selecting an appropriate $\lambda \in (0, \infty)$ in (8), we straddle the two extremes of *subspace closeness* and *data approximation*. In order to solve (8), we define an $m \times m$ symmetric matrix $\mathbf{A}_\ell = \sum_{p\neq\ell}\mathbf{D}_p\mathbf{D}_p^T + \frac{\lambda}{2}\widetilde{\mathbf{Y}}_\ell\widetilde{\mathbf{Y}}_\ell^T$. It then follows from [53] that (8) has a closed-form solution that involves eigen decomposition of $\mathbf{A}_\ell$. Specifically, $\mathbf{D}_\ell = \arg\max \operatorname{tr}(\mathbf{D}_\ell^T\mathbf{A}_\ell\mathbf{D}_\ell)$ is given by the first $s$ eigenvectors of $\mathbf{A}_\ell$ associated with its $s$-largest eigenvalues.

This completes our description of the subspace update step. We can now combine the subspace assignment and subspace update steps to fully describe our algorithm for MC-UoS learn-

ing. This algorithm, which we term *metric-constrained union-of-subspaces learning* (MiCUSaL), is given by Algorithm 1. In terms of the complexity of this algorithm in each iteration, notice that the subspace assignment step requires $\mathcal{O}(mLsN)$ operations. In addition, the total number of operations needed to compute the $\mathbf{A}_\ell$'s in each iteration is $\mathcal{O}(m^2(Ls+N))$. Finally, each iteration also requires $L$ eigen decompositions of $m \times m$ matrices, each one of which has $\mathcal{O}(m^3)$ complexity. Therefore, the computational complexity of MiCUSaL in each iteration is given by $\mathcal{O}(m^3L + m^2N + m^2Ls + mLsN)$. We conclude this discussion by pointing out that we cannot guarantee convergence of MiCUSaL to a global optimal solution. However, since the objective function $F_1$ in (6) is bounded below by zero and MiCUSaL ensures that $F_1$ does not increase after each iteration, it follows that MiCUSaL iterates do indeed converge (possibly to one of the local optimal solutions). This local convergence, of course, will be a function of the initialization of MiCUSaL. In this paper, we advocate the use of random subspaces for initialization, while we study the impact of such random initialization in Sec. V-A.

### B. Practical Considerations

The MiCUSaL algorithm described in Sec. III-A requires knowledge of the number of subspaces $L$ and the dimension of subspaces $s$. In practice, however, one cannot assume knowledge of these parameters a priori. Instead, one must estimate both the number and the dimension of subspaces from the training data themselves. In this section, we describe a generalization of the MiCUSaL algorithm that achieves this objective. Our algorithm, which we term *adaptive MC-UoS learning* (aMiCUSaL), requires only knowledge of loose upper bounds on $L$ and $s$, which we denote by $L_{max}$ and $s_{max}$, respectively.

The aMiCUSaL algorithm initializes with a collection of random orthonormal bases $\mathbf{D} = [\mathbf{D}_1, \ldots, \mathbf{D}_{L_{max}}]$, where each basis $\mathbf{D}_\ell$ is a point on the Stiefel manifold $\mathcal{V}_{m,s_{max}}$. Similar to the case of MiCUSaL, it then carries out the subspace assignment and subspace update steps in an iterative fashion. Unlike MiCUSaL, however, we also greedily remove redundant subspaces from our collection of subspaces $\{\mathcal{S}_\ell\}_{\ell=1}^{L_{max}}$ after each subspace assignment step. This involves removal of $\mathbf{D}_\ell$ from $\mathbf{D}$ if no signals in our training data get assigned to the subspace $\mathcal{S}_\ell$. This step of *greedy subspace pruning* ensures that only "active" subspaces survive before the subspace update step.

Once the aMiCUSaL algorithm finishes iterating between subspace assignment, subspace pruning, and subspace update, we move onto the step of *greedy subspace merging*, which involves merging of pairs of subspaces that are so close to each other that even a single subspace of the same dimension can be used to well approximate the data represented by the two subspaces individually.[3] In this step, we greedily merge pairs of closest subspaces as long as their normalized subspace distance is below a predefined threshold $\epsilon_{min} \in [0, 1)$. Mathematically, the subspace merging step involves first finding the pair of subspaces $(\mathcal{S}_{\ell^*}, \mathcal{S}_{p^*})$ that satisfies

$$(\ell^*, p^*) = \arg\min_{\ell \neq p} d_u(\mathcal{S}_\ell, \mathcal{S}_p) \text{ s.t. } \frac{d_u(\mathcal{S}_{\ell^*}, \mathcal{S}_{p^*})}{\sqrt{s_{max}}} \leq \epsilon_{min}. \quad (9)$$

We then merge $\mathcal{S}_{\ell^*}$ and $\mathcal{S}_{p^*}$ by setting $\mathbf{c}_{\ell^*} = \mathbf{c}_{\ell^*} \cup \mathbf{c}_{p^*}$ and $\widetilde{\mathbf{Y}}_{\ell^*} = [\widetilde{\mathbf{y}}_i : i \in \mathbf{c}_{\ell^*}]$, where $\mathbf{c}_{\ell^*}, \mathbf{c}_{p^*}$ are as defined in Algorithm 1. By defining an $m \times m$ symmetric matrix $\mathbf{A}_{\ell^*} = \sum_{\ell \neq \ell^*, p^*} \mathbf{D}_\ell \mathbf{D}_\ell^T + \frac{\lambda}{2} \widetilde{\mathbf{Y}}_{\ell^*} \widetilde{\mathbf{Y}}_{\ell^*}^T$, $\mathbf{D}_{\ell^*}$ is then set equal to the first $s_{max}$ eigenvectors of $\mathbf{A}_{\ell^*}$ associated with its $s_{max}$-largest eigenvalues. Finally, we remove $\mathbf{D}_{p^*}$ from $\mathbf{D}$. This process of finding the closest pair of subspaces and merging them is repeated until the normalized subspace distance between every pair of subspaces becomes greater than $\epsilon_{min}$. We assume without loss of generality that $\widehat{L}$ subspaces are left after this greedy subspace merging, where each $\mathcal{S}_\ell$ $(\ell = 1, \ldots, \widehat{L})$ is a subspace in $\mathbb{R}^m$ of dimension $s_{max}$.

After subspace merging, we move onto the step of estimation of the dimension, $s$, of the subspaces. To this end, we first estimate the dimension of each subspace $\mathcal{S}_\ell$, denoted by $s_\ell$, and then $s$ is selected as the maximum of these $s_\ell$'s. There have been many efforts in the literature to estimate the dimension of a subspace; see, e.g., [54]–[57] for an incomplete list. In this paper, we focus on the method given in [54], which formulates the maximum likelihood estimator (MLE) of $s_\ell$. This is because: $(i)$ the noise level is unknown in our problem, and $(ii)$ the MLE in [54] has a simple form. However, the MLE of [54] is sensitive to noise. We therefore first apply a "smoothing" process before using that estimator. This involves first updating $\mathbf{W}$ (i.e., $\mathbf{c}_\ell$'s) using the updated $\mathbf{D}$ and "denoising" our data by projecting $\widetilde{\mathbf{Y}}_\ell$ onto $\mathcal{S}_\ell$, given by $\widehat{\mathbf{Y}}_\ell = \mathbf{D}_\ell \mathbf{D}_\ell^T \widetilde{\mathbf{Y}}_\ell$, and then using $\widehat{\mathbf{Y}}_\ell$ to estimate $s_\ell$. For a given column $\widehat{\mathbf{y}}$ in $\widehat{\mathbf{Y}}_\ell$ and a fixed number of nearest neighbors $k_0$, the unbiased MLE of $s_\ell$ with respect to $\widehat{\mathbf{y}}$ is given by [54]

$$\widehat{s}_\ell^{k_0}(\widehat{\mathbf{y}}) = \Big[\frac{1}{k_0 - 2} \sum_{a=1}^{k_0 - 1} \log \frac{\Gamma_{k_0}(\widehat{\mathbf{y}})}{\Gamma_a(\widehat{\mathbf{y}})}\Big]^{-1}, \quad (10)$$

where $\Gamma_a(\widehat{\mathbf{y}})$ is the $\ell_2$ distance between $\widehat{\mathbf{y}}$ and its $a$-th nearest neighbor in $\widehat{\mathbf{Y}}_\ell$. An estimate of $s_\ell$ can now be written as the average of all estimates with respect to every signal in $\widehat{\mathbf{Y}}_\ell$, i.e., $\widehat{s}_\ell^{k_0} = \frac{1}{|\mathbf{c}_\ell|} \sum_{i \in \mathbf{c}_\ell} \widehat{s}_\ell^{k_0}(\widehat{\mathbf{y}}_i)$. In fact, as suggested in [54], we estimate $s_\ell$ by averaging over a range of $k_0 = k_1, \ldots, k_2$, i.e.,

$$\widehat{s}_\ell = \frac{1}{k_2 - k_1 + 1} \sum_{k_0 = k_1}^{k_2} \widehat{s}_\ell^{k_0}. \quad (11)$$

Once we get an estimate $s = \max_\ell \widehat{s}_\ell$, we trim each orthonormal basis by keeping only the first $s$ columns of each (ordered) orthonormal basis $\mathbf{D}_\ell$ in our collection, which is denoted by $\widehat{\mathbf{D}}_\ell$.[4] Given the bases $\{\widehat{\mathbf{D}}_\ell \in \mathbb{R}^{m \times s}\}_{\ell=1}^{\widehat{L}}$, we finally run MiCUSaL again that is initialized using these $\widehat{\mathbf{D}}_\ell$'s until it converges. Combining all the steps mentioned above,

[3] Note that the step of subspace merging is needed due to our lack of knowledge of the true number of subspaces in the underlying model. In particular, the assumption here is that the merging threshold $\epsilon_{min}$ in Algorithm 2 satisfies $\epsilon_{min} \ll \frac{\epsilon}{\sqrt{s}}$.

[4] Recall that the columns of $\mathbf{D}_\ell$ correspond to the eigenvectors of $\mathbf{A}_\ell$. Here, we are assuming that the order of these eigenvectors within $\mathbf{D}_\ell$ corresponds to the nonincreasing order of the eigenvalues of $\mathbf{A}_\ell$. Therefore, $\widehat{\mathbf{D}}_\ell$ comprises the eigenvectors of $\mathbf{A}_\ell$ associated with its $s$-largest eigenvalues.

**Algorithm 2:** Adaptive MC-UoS Learning (aMiCUSaL)

**Input:** Training data $\mathbf{Y} \in \mathbb{R}^{m\times N}$, loose upper bounds $L_{max}$ and $s_{max}$, and parameters $\lambda$, $k_1$, $k_2$, $\epsilon_{min}$.
**Initialize:** Random orthonormal bases $\{\mathbf{D}_\ell \in \mathbb{R}^{m\times s_{max}}\}_{\ell=1}^{L_{max}}$, and set $\widehat{L} \leftarrow L_{max}$.
$\bar{\mathbf{y}} \leftarrow \frac{1}{N}\sum_{i=1}^N \mathbf{y}_i$, $\widetilde{\mathbf{y}}_i \leftarrow \mathbf{y}_i - \bar{\mathbf{y}}$, $i = 1,\ldots,N$.
**while** stopping rule **do**
  Fix $\mathbf{D}$ and update $\mathbf{W}$ using (7). Also, set $\boldsymbol{\mathcal{T}} \leftarrow \emptyset$ and $L_1 \leftarrow 0$.
  **for** $\ell = 1$ to $\widehat{L}$ (*Subspace Pruning*) **do**
    $\mathbf{c}_\ell \leftarrow \{i \in \{1,\ldots,N\} : w_{\ell,i} = 1\}$.
    If $|\mathbf{c}_\ell| \neq 0$ then $\mathbf{c}_{L_1+1} \leftarrow \mathbf{c}_\ell$, $\widetilde{\mathbf{Y}}_{L_1+1} \leftarrow [\widetilde{\mathbf{y}}_i : i \in \mathbf{c}_\ell]$, $L_1 \leftarrow L_1 + 1$ and $\boldsymbol{\mathcal{T}} \leftarrow \boldsymbol{\mathcal{T}} \cup \{\ell\}$.
  **end for**
  $\mathbf{D} \leftarrow [\mathbf{D}_{\boldsymbol{\mathcal{T}}_{(1)}}, \ldots, \mathbf{D}_{\boldsymbol{\mathcal{T}}_{(L_1)}}]$ and $\widehat{L} \leftarrow L_1$.
  Update each $\mathbf{D}_\ell$ ($\ell = 1,\ldots,\widehat{L}$) in $\mathbf{D}$ using (8).
**end while**
$(\ell^*, p^*) = \arg\min_{\ell\neq p, \ell, p=1,\ldots,\widehat{L}} d_u(\mathcal{S}_\ell, \mathcal{S}_p)$.
**while** $\frac{d_u(\mathcal{S}_{\ell^*}, \mathcal{S}_{p^*})}{\sqrt{s_{max}}} \leq \epsilon_{min}$ (*Subspace Merging*) **do**
  Merge $\mathcal{S}_{\ell^*}$ and $\mathcal{S}_{p^*}$, and update $\mathbf{Y}_{\ell^*}$ and $\mathbf{D}_{\ell^*}$.
  $\mathbf{D} \leftarrow [\mathbf{D}_1, \ldots, \mathbf{D}_{\ell^*}, \ldots, \mathbf{D}_{p^*-1}, \mathbf{D}_{p^*+1}, \ldots, \mathbf{D}_{\widehat{L}}]$ and $\widehat{L} \leftarrow \widehat{L} - 1$.
  $(\ell^*, p^*) = \arg\min_{\ell\neq p, \ell, p=1,\ldots,\widehat{L}} d_u(\mathcal{S}_\ell, \mathcal{S}_p)$.
**end while**
Fix $\mathbf{D}$, update $\mathbf{W} \in \mathbb{R}^{\widehat{L}\times N}$ using (7), and update $\{\mathbf{c}_\ell\}_{\ell=1}^{\widehat{L}}$.
**for** $\ell = 1$ to $\widehat{L}$ **do**
  $\widetilde{\mathbf{Y}}_\ell \leftarrow [\widetilde{\mathbf{y}}_i : i \in \mathbf{c}_\ell]$ and $\widehat{\mathbf{Y}}_\ell \leftarrow \mathbf{D}_\ell \mathbf{D}_\ell^T \widetilde{\mathbf{Y}}_\ell$.
  Calculate $\widehat{s}_\ell$ using (10) and (11).
**end for**
$s \leftarrow \max\{\widehat{s}_1, \ldots, \widehat{s}_{\widehat{L}}\}$.
$\widehat{\mathbf{D}}_\ell \leftarrow$ First $s$ columns of $\mathbf{D}_\ell$, $\ell = 1,\ldots,\widehat{L}$.
Initialize Algorithm 1 with $\{\widehat{\mathbf{D}}_\ell\}_{\ell=1}^{\widehat{L}}$ and update $\{\widehat{\mathbf{D}}_\ell\}_{\ell=1}^{\widehat{L}}$ using Algorithm 1.
**Output:** Orthonormal bases $\{\widehat{\mathbf{D}}_\ell \in \mathbb{R}^{m\times s}\}_{\ell=1}^{\widehat{L}}$.

we can now formally describe *adaptive MC-UoS learning* (aMiCUSaL) in Algorithm 2.

### C. MC-UoS Learning Using Missing Data

In this section, we study MC-UoS learning for the case of training data having missing entries. To be specific, for each $\mathbf{y}_i$ in $\mathbf{Y}$, we assume to only observe its entries at locations given by the set $\boldsymbol{\Omega}_i \subset \{1,\ldots,m\}$ with $|\boldsymbol{\Omega}_i| > s$, which is denoted by $[\mathbf{y}_i]_{\boldsymbol{\Omega}_i} \in \mathbb{R}^{|\boldsymbol{\Omega}_i|}$. Since we do not have access to the complete data, it is impossible to compute the quantities $\|\mathbf{y}_i - P_{\mathcal{S}_\ell}\mathbf{y}_i\|_2^2$ in (2) explicitly. But, it is shown in [58] that $\|[\mathbf{y}_i]_{\boldsymbol{\Omega}_i} - P_{\mathcal{S}_{\ell\boldsymbol{\Omega}_i}}[\mathbf{y}_i]_{\boldsymbol{\Omega}_i}\|_2^2$ for uniformly random $\boldsymbol{\Omega}_i$ is very close to $\frac{|\boldsymbol{\Omega}_i|}{m}\|\mathbf{y}_i - P_{\mathcal{S}_\ell}\mathbf{y}_i\|_2^2$ with very high probability as long as $|\boldsymbol{\Omega}_i|$ is slightly greater than $s$. Here, $P_{\mathcal{S}_{\ell\boldsymbol{\Omega}_i}}$ is defined as $P_{\mathcal{S}_{\ell\boldsymbol{\Omega}_i}} = [\mathbf{D}_\ell]_{\boldsymbol{\Omega}_i,:}([\mathbf{D}_\ell]_{\boldsymbol{\Omega}_i,:})^\dagger$ with $([\mathbf{D}_\ell]_{\boldsymbol{\Omega}_i,:})^\dagger = \left([\mathbf{D}_\ell]_{\boldsymbol{\Omega}_i,:}^T[\mathbf{D}_\ell]_{\boldsymbol{\Omega}_i,:}\right)^{-1}[\mathbf{D}_\ell]_{\boldsymbol{\Omega}_i,:}^T$. Motivated by this, we replace $\|\mathbf{y}_i - P_{\mathcal{S}_\ell}\mathbf{y}_i\|_2^2$ by $\frac{m}{|\boldsymbol{\Omega}_i|}\|[\mathbf{y}_i]_{\boldsymbol{\Omega}_i} - P_{\mathcal{S}_{\ell\boldsymbol{\Omega}_i}}[\mathbf{y}_i]_{\boldsymbol{\Omega}_i}\|_2^2$ in (2) and reformulate the MC-UoS learning problem as $(\mathbf{D}, \mathbf{W}) = \arg\min_{\mathbf{D},\mathbf{W}} F_2(\mathbf{D}, \mathbf{W})$ with the objective function $F_2(\mathbf{D}, \mathbf{W})$ given by

$$F_2(\mathbf{D}, \mathbf{W}) = \sum_{\substack{\ell,p=1\\ \ell\neq p}}^{L} \|\mathbf{D}_\ell - P_{\mathcal{S}_p}\mathbf{D}_\ell\|_F^2 + \lambda \sum_{i=1}^{N}\sum_{\ell=1}^{L} w_{\ell,i}\frac{m}{|\boldsymbol{\Omega}_i|}\left\|[\mathbf{y}_i]_{\boldsymbol{\Omega}_i} - P_{\mathcal{S}_{\ell\boldsymbol{\Omega}_i}}[\mathbf{y}_i]_{\boldsymbol{\Omega}_i}\right\|_2^2. \quad (12)$$

As in Sec. III-A, we propose to solve this problem by making use of alternating minimization that comprises subspace assignment and subspace update steps. To this end, we again initialize $\mathbf{D}$ such that each block $\mathbf{D}_\ell \in \mathbb{R}^{m\times s}$ is a random orthonormal basis. Next, when $\mathbf{D}$ is fixed, subspace assignment corresponds to solving for each $i = 1,\ldots,N$,

$$l_i = \arg\min_{\ell=1,\ldots,L} \|[\mathbf{y}_i]_{\boldsymbol{\Omega}_i} - P_{\mathcal{S}_{\ell\boldsymbol{\Omega}_i}}[\mathbf{y}_i]_{\boldsymbol{\Omega}_i}\|_2^2, \quad (13)$$

and then setting $w_{l_i,i} = 1$ and $w_{\ell,i} = 0$ $\forall \ell \neq l_i$. When $\mathbf{W}$ is fixed, we carry out subspace update using BCD again, in which case $\min_{\mathbf{D}} F_2(\mathbf{D}, \mathbf{W})$ for a fixed $\mathbf{W}$ can be decoupled into $L$ distinct problems of the form $\mathbf{D}_\ell = \arg\min_{\mathbf{D}_\ell \in \mathcal{V}_{m,s}} f_2(\mathbf{D}_\ell)$, $\ell = 1,\ldots,L$, with

$$f_2(\mathbf{D}_\ell) = -\mathrm{tr}(\mathbf{D}_\ell^T\mathbf{A}_\ell\mathbf{D}_\ell) + \frac{\lambda}{2}\sum_{i\in\mathbf{c}_\ell}\frac{m}{|\boldsymbol{\Omega}_i|}\left\|[\mathbf{y}_i]_{\boldsymbol{\Omega}_i} - P_{\mathcal{S}_{\ell\boldsymbol{\Omega}_i}}[\mathbf{y}_i]_{\boldsymbol{\Omega}_i}\right\|_2^2.$$

Here, $\mathbf{c}_\ell$ is as defined in Sec. III-A and $\mathbf{A}_\ell = \sum_{p\neq\ell}\mathbf{D}_p\mathbf{D}_p^T$. It is also easy to verify that $f_2(\mathbf{D}_\ell)$ is invariant to the choice of the orthonormal basis of $\mathcal{S}_\ell$; hence, we can treat $\min_{\mathbf{D}_\ell\in\mathcal{V}_{m,s}} f_2(\mathbf{D}_\ell)$ as an optimization problem on the Grassmann manifold [59]. Note that we can rewrite $f_2(\mathbf{D}_\ell)$ as $f_2(\mathbf{D}_\ell) = \sum_{q=0}^{|\mathbf{c}_\ell|} f_2^{(q)}(\mathbf{D}_\ell)$, where $f_2^{(0)}(\mathbf{D}_\ell) = -\mathrm{tr}(\mathbf{D}_\ell^T\mathbf{A}_\ell\mathbf{D}_\ell)$ and $f_2^{(q)}(\mathbf{D}_\ell) = \frac{\lambda m}{2|\boldsymbol{\Omega}_{\mathbf{c}_{\ell(q)}}|}\|[\mathbf{y}_{\mathbf{c}_{\ell(q)}}]_{\boldsymbol{\Omega}_{\mathbf{c}_{\ell(q)}}} - P_{\mathcal{S}_{\ell\boldsymbol{\Omega}_{\mathbf{c}_{\ell(q)}}}}[\mathbf{y}_{\mathbf{c}_{\ell(q)}}]_{\boldsymbol{\Omega}_{\mathbf{c}_{\ell(q)}}}\|_2^2$ for $q = 1,\ldots,|\mathbf{c}_\ell|$. In here, $\mathbf{c}_{\ell(q)}$ denotes the $q$-th element in $\mathbf{c}_\ell$. In order to minimize $f_2(\mathbf{D}_\ell)$, we employ incremental gradient descent procedure on Grassmann manifold [60], which performs the update with respect to a single component of $f_2(\mathbf{D}_\ell)$ in each step. To be specific, we first compute the gradient of a single cost function $f_2^{(q)}(\mathbf{D}_\ell)$ in $f_2(\mathbf{D}_\ell)$, and move along a short geodesic curve in the gradient direction. For instance, the gradient of $f_2^{(0)}(\mathbf{D}_\ell)$ is

$$\nabla f_2^{(0)} = (\mathbf{I}_m - \mathbf{D}_\ell\mathbf{D}_\ell^T)\frac{df_2^{(0)}}{d\mathbf{D}_\ell} = -2(\mathbf{I}_m - \mathbf{D}_\ell\mathbf{D}_\ell^T)\mathbf{A}_\ell\mathbf{D}_\ell.$$

Then the geodesic equation emanating from $\mathbf{D}_\ell$ in the direction $-\nabla f_2^{(0)}$ with a step length $\eta$ is given by [59]

$$\mathbf{D}_\ell(\eta) = \mathbf{D}_\ell\mathbf{V}_\ell\cos(\boldsymbol{\Sigma}_\ell\eta)\mathbf{V}_\ell^T + \mathbf{U}_\ell\sin(\boldsymbol{\Sigma}_\ell\eta)\mathbf{V}_\ell^T, \quad (14)$$

where $\mathbf{U}_\ell\boldsymbol{\Sigma}_\ell\mathbf{V}_\ell^T$ is the SVD decomposition of $-\nabla f_2^{(0)}$. The update of $\mathbf{D}_\ell$ with respect to $f_2^{(q)}(\mathbf{D}_\ell)$ ($q = 1,\ldots,|\mathbf{c}_\ell|$) can be performed as in the GROUSE algorithm [61] *but* with a step size $\frac{\lambda m}{2|\boldsymbol{\Omega}_{\mathbf{c}_{\ell(q)}}|}\eta$. In order for $f_2$ to converge, we also reduce the step size after each iteration [61]. We complete this discussion by presenting our learning algorithm for missing data in

**Algorithm 3:** Robust MC-UoS learning (rMiCUSaL)

**Input:** Training data $\{[\mathbf{y}_i]_{\mathbf{\Omega}_i}\}_{i=1}^N$, number of subspaces $L$, dimension of subspaces $s$, and parameters $\lambda$ and $\eta$.
**Initialize:** Random orthonormal bases $\{\mathbf{D}_\ell \in \mathbb{R}^{m\times s}\}_{\ell=1}^L$.

**while** stopping rule **do**
  **for** $i = 1$ to $N$ (*Subspace Assignment*) **do**
    $l_i \leftarrow \arg\min_\ell \|[\mathbf{y}_i]_{\mathbf{\Omega}_i} - P_{\mathcal{S}_\ell \mathbf{\Omega}_i}[\mathbf{y}_i]_{\mathbf{\Omega}_i}\|_2^2$.
    $w_{l_i,i} \leftarrow 1$ and $\forall \ell \neq l_i,\ w_{\ell,i} \leftarrow 0$.
  **end for**
  **for** $\ell = 1$ to $L$ (*Subspace Update*) **do**
    $\mathbf{c}_\ell \leftarrow \{i \in \{1,\dots,N\} : w_{\ell,i} = 1\},\ t \leftarrow 0$.
    **while** stopping rule **do**
      $t \leftarrow t+1,\ \eta_t \leftarrow \frac{\eta}{t}$.
      $\mathbf{A}_\ell \leftarrow \sum_{p\neq\ell} \mathbf{D}_p\mathbf{D}_p^T,\ \mathbf{\Delta}_\ell \leftarrow 2(\mathbf{I}_m - \mathbf{D}_\ell\mathbf{D}_\ell^T)\mathbf{A}_\ell\mathbf{D}_\ell$.
      $\mathbf{D}_\ell \leftarrow \mathbf{D}_\ell\mathbf{V}_\ell\cos(\mathbf{\Sigma}_\ell\eta_t)\mathbf{V}_\ell^T + \mathbf{U}_\ell\sin(\mathbf{\Sigma}_\ell\eta_t)\mathbf{V}_\ell^T$, where $\mathbf{U}_\ell\mathbf{\Sigma}_\ell\mathbf{V}_\ell^T$ is the compact SVD of $\mathbf{\Delta}_\ell$.
      **for** $q = 1$ to $|\mathbf{c}_\ell|$ **do**
        $\boldsymbol{\theta} \leftarrow ([\mathbf{D}_\ell]_{\mathbf{\Omega}_{\mathbf{c}_\ell(q)},:})^\dagger [\mathbf{y}_{\mathbf{c}_\ell(q)}]_{\mathbf{\Omega}_{\mathbf{c}_\ell(q)}},\ \boldsymbol{\omega} \leftarrow \mathbf{D}_\ell\boldsymbol{\theta}$.
        $\mathbf{r} \leftarrow \mathbf{0}_m,\ [\mathbf{r}]_{\mathbf{\Omega}_{\mathbf{c}_\ell(q)}} \leftarrow [\mathbf{y}_{\mathbf{c}_\ell(q)}]_{\mathbf{\Omega}_{\mathbf{c}_\ell(q)}} - [\boldsymbol{\omega}]_{\mathbf{\Omega}_{\mathbf{c}_\ell(q)}}$.
        $\mathbf{D}_\ell \leftarrow \mathbf{D}_\ell + \Big((\cos(\mu\frac{\lambda m}{|\mathbf{\Omega}_{\mathbf{c}_\ell(q)}|}\eta_t) - 1)\frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|_2} + \sin(\mu\frac{\lambda m}{|\mathbf{\Omega}_{\mathbf{c}_\ell(q)}|}\eta_t)\frac{\mathbf{r}}{\|\mathbf{r}\|_2}\Big)\frac{\boldsymbol{\theta}^T}{\|\boldsymbol{\theta}\|_2}$, where $\mu = \|\mathbf{r}\|_2\|\boldsymbol{\omega}\|_2$.
      **end for**
    **end while**
  **end for**
**end while**

**Output:** Orthonormal bases $\{\mathbf{D}_\ell \in \mathbb{R}^{m\times s}\}_{\ell=1}^L$.

Algorithm 3, termed *robust MC-UoS learning* (rMiCUSaL). By defining $T = \max_i |\mathbf{\Omega}_i|$, the subspace assignment step of rMiCUSaL requires $\mathcal{O}(TLs^2N)$ flops in each iteration [61]. Computing the $\mathbf{A}_\ell$'s in each iteration requires $\mathcal{O}(m^2Ls)$ operations. Next, the cost of updating each $\mathbf{D}_\ell$ with respect to $f_2^{(0)}(\mathbf{D}_\ell)$ is $\mathcal{O}(m^3)$, while it is $\mathcal{O}(ms + Ts^2)$ with respect to $f_2^{(q)}(\mathbf{D}_\ell)$ for $q \neq 0$ [61]. It therefore follows that the computational complexity of rMiCUSaL in each iteration is $\mathcal{O}(m^3L + m^2Ls + msN + TLs^2N)$. We refer the reader to Sec. V-A for exact running time of rMiCUSaL on training data.

## IV. MC-KUoS Learning for Highly Nonlinear Data

In this section, we present algorithms to solve the problem of MC-KUoS learning from $\mathbf{Y} \in \mathbb{R}^{m\times N}$ for highly nonlinear data. We first generalize the MiCUSaL algorithm using the kernel trick [18] to learn an MC-KUoS from complete data. To deal with the case of "missing data," we propose "kernel function value estimators" to solve (3). Finally, we discuss the problem of finding the "pre-images" of data in the feature space based on the MC-KUoS model in Sec. IV-C.

### A. MC-KUoS Learning Using Complete Data

To begin our discussion, we define the kernel matrix on the training data $\mathbf{Y}$ to be $\mathbf{G} = \phi(\mathbf{Y})^T\phi(\mathbf{Y}) \in \mathbb{R}^{N\times N}$, with its individual entries $g_{i,j} = \kappa(\mathbf{y}_i, \mathbf{y}_j)$ for a pre-specified kernel $\kappa : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$. Under the assumption that $\operatorname{rank}(\phi(\mathbf{Y})) = N$, the matrix $\mathbf{G}$ is positive definite. Similar to Algorithm 1, we begin with centering the $\phi$-mapped data in the feature space $\mathcal{F}$ as a pre-processing stage.[5] We denote the mean of the $\phi$-mapped "images" of $\mathbf{Y}$ by $\overline{\boldsymbol{\phi}} = \frac{1}{N}\sum_{i=1}^N \phi(\mathbf{y}_i)$ and write the $N$ centered "mapped training data" as $\widetilde{\phi}(\mathbf{Y}) = [\widetilde{\phi}(\mathbf{y}_1), \dots, \widetilde{\phi}(\mathbf{y}_N)]$, where $\widetilde{\phi}(\mathbf{y}_i) = \phi(\mathbf{y}_i) - \overline{\boldsymbol{\phi}}$, $i = 1,\dots,N$. The centered kernel matrix $\widetilde{\mathbf{G}} = \widetilde{\phi}(\mathbf{Y})^T\widetilde{\phi}(\mathbf{Y})$ can be calculated from $\mathbf{G}$ by [18]

$$\widetilde{\mathbf{G}} = \mathbf{G} - \mathbf{H}_N\mathbf{G} - \mathbf{G}\mathbf{H}_N + \mathbf{H}_N\mathbf{G}\mathbf{H}_N, \tag{15}$$

where $\mathbf{H}_N$ is an $N \times N$ matrix with all elements $\frac{1}{N}$. Then for any $\mathbf{y}, \mathbf{y}' \in \mathbb{R}^m$, we have [50]

$$\begin{aligned}\widetilde{\kappa}(\mathbf{y},\mathbf{y}') &= \widetilde{\phi}(\mathbf{y})^T\widetilde{\phi}(\mathbf{y}') \\ &= \kappa(\mathbf{y},\mathbf{y}') - \frac{1}{N}\mathbf{1}_N^T\boldsymbol{k}_{\mathbf{y}} - \frac{1}{N}\mathbf{1}_N^T\boldsymbol{k}_{\mathbf{y}'} + \frac{1}{N^2}\mathbf{1}_N^T\mathbf{G}\mathbf{1}_N,\end{aligned}$$

where $\mathbf{1}_N = [1,1,\dots,1]^T$ is an $N$-dimensional vector and $\boldsymbol{k}_{\mathbf{y}} = [\kappa(\mathbf{y},\mathbf{y}_1),\dots,\kappa(\mathbf{y},\mathbf{y}_N)]^T \in \mathbb{R}^N$. To write the expression in (3) in terms of inner products, we again use $\mathbf{W}$ to denote the membership indicator matrix as in (4), where $w_{\ell,i} = 1$, $\ell = 1,\dots,L$, $i = 1,\dots,N$, if and only if $\widetilde{\phi}(\mathbf{y}_i)$ is assigned to subspace $\mathcal{S}_\ell$. Let $\mathbf{D} = [\mathbf{D}_1,\dots,\mathbf{D}_L]$, where $\mathbf{D}_\ell$ is an (arbitrary) orthonormal basis of $\mathcal{S}_\ell$. Then for any $i = 1,\dots,N$, we have the following

$$\begin{aligned}\|\phi(\mathbf{y}_i) - P_{\mathcal{S}_\ell}\phi(\mathbf{y}_i)\|_2^2 &= \|\widetilde{\phi}(\mathbf{y}_i) - P_{\mathcal{S}_\ell}\widetilde{\phi}(\mathbf{y}_i)\|_2^2 \\ &= \|\widetilde{\phi}(\mathbf{y}_i)\|_2^2 - \|\mathbf{D}_\ell^T\widetilde{\phi}(\mathbf{y}_i)\|_2^2.\end{aligned} \tag{16}$$

Therefore, (3) can be written as $(\mathbf{D},\mathbf{W}) = \arg\min_{\mathbf{D},\mathbf{W}} F_3(\mathbf{D},\mathbf{W})$ with the objective function $F_3(\mathbf{D},\mathbf{W})$ given by

$$\begin{aligned}F_3(\mathbf{D},\mathbf{W}) = &\sum_{\substack{\ell,p=1\\ \ell\neq p}}^L \|\mathbf{D}_\ell - P_{\mathcal{S}_p}\mathbf{D}_\ell\|_F^2 \\ &+ \lambda\sum_{i=1}^N\sum_{\ell=1}^L w_{\ell,i}(\|\widetilde{\phi}(\mathbf{y}_i)\|_2^2 - \|\mathbf{D}_\ell^T\widetilde{\phi}(\mathbf{y}_i)\|_2^2).\end{aligned} \tag{17}$$

Before discussing our algorithm to minimize (17) using the kernel trick, we further simplify the terms in (17). We again define $\mathbf{c}_\ell = \{i \in \{1,\dots,N\} : w_{\ell,i} = 1\}$ to be the set containing the indices of all $\widetilde{\phi}(\mathbf{y}_i)$'s that are assigned to $\mathcal{S}_\ell$, and let $\mathbf{Y}_\ell = [\mathbf{y}_i : i \in \mathbf{c}_\ell]$ be the corresponding $m \times |\mathbf{c}_\ell|$ matrix. Then the centered data that are assigned to subspace $\mathcal{S}_\ell$ can be denoted by $\widetilde{\phi}(\mathbf{Y}_\ell) = [\widetilde{\phi}(\mathbf{y}_i) : i \in \mathbf{c}_\ell]$. Since $\mathcal{S}_\ell$ is spanned by the columns of $\widetilde{\phi}(\mathbf{Y}_\ell)$, we can write $\mathbf{D}_\ell = \widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{E}_\ell$, where $\mathbf{E}_\ell \in \mathbb{R}^{N_\ell\times s}$ is some basis representation matrix with $N_\ell = |\mathbf{c}_\ell|$ such that $\mathbf{D}_\ell$ is an orthonormal matrix. Also, it is easy to verify that $\mathbf{E}_\ell$ has to satisfy $\mathbf{E}_\ell^T[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}\mathbf{E}_\ell = \mathbf{I}_s$, where $[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell} = \widetilde{\phi}(\mathbf{Y}_\ell)^T\widetilde{\phi}(\mathbf{Y}_\ell)$ denotes the centered kernel matrix for subspace $\mathcal{S}_\ell$. Now instead of using $\mathbf{D}_\ell$ explicitly for computations, it suffices to use $\mathbf{c}_\ell$ and $\mathbf{E}_\ell$ for MC-KUoS learning and all the computations involving $\mathbf{D}_\ell$ can be carried

[5] This step is only for the purpose of derivation of our algorithm. In particular, explicit centering of data in the feature space is never required in the following.

**Algorithm 4:** Initialization for $\mathcal{S}_\ell$'s in $\mathcal{F}$ (GKIOP)

**Input:** Centered kernel matrix $\widetilde{\mathbf{G}} \in \mathbb{R}^{N\times N}$, number of subspaces $L$, and dimension of subspaces $s$.
**Initialize:** $\boldsymbol{\mathcal{I}}_N \leftarrow \{1,\dots,N\}$ and $\{\mathbf{c}_\ell \leftarrow \emptyset\}_{\ell=1}^L$.
1: **for** $\ell = 1$ to $L$ **do**
2: Choose an arbitrary element in $\boldsymbol{\mathcal{I}}_N$, include that element in $\mathbf{c}_\ell$, and set $\boldsymbol{\mathcal{I}}_N \leftarrow \boldsymbol{\mathcal{I}}_N \setminus \mathbf{c}_\ell$.
3: **for** $q = 2$ to $s$ **do**
4: $i^* \leftarrow \arg\max_{i\in\boldsymbol{\mathcal{I}}_N} \sum_{j\in\mathbf{c}_\ell} \widetilde{g}_{i,j}$.
5: Set $\mathbf{c}_\ell \leftarrow \mathbf{c}_\ell \cup \{i^*\}$ and $\boldsymbol{\mathcal{I}}_N \leftarrow \boldsymbol{\mathcal{I}}_N \setminus \{i^*\}$.
6: **end for**
7: Eigen decomposition of $[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell} = \mathbf{U}_\ell\boldsymbol{\Sigma}_\ell\mathbf{U}_\ell^T$.
8: $\mathbf{E}_\ell \leftarrow \mathbf{U}_\ell\boldsymbol{\Sigma}_\ell^{-\frac{1}{2}}$.
9: **end for**
**Output:** Initial $\{\mathbf{c}_\ell\}_{\ell=1}^L$ and $\{\mathbf{E}_\ell \in \mathbb{R}^{s\times s}\}_{\ell=1}^L$.

out using $\mathbf{c}_\ell$, $\mathbf{E}_\ell$ and the kernel trick. Specifically, notice that for any $i = 1,\dots,N$, we can write

$$\|\widetilde{\phi}(\mathbf{y}_i)\|_2^2 - \|\mathbf{D}_\ell^T\widetilde{\phi}(\mathbf{y}_i)\|_2^2 = \widetilde{\kappa}(\mathbf{y}_i,\mathbf{y}_i) - \|\mathbf{E}_\ell^T\widetilde{\phi}(\mathbf{Y}_\ell)^T\widetilde{\phi}(\mathbf{y}_i)\|_2^2,$$

where $\widetilde{\kappa}(\mathbf{y}_i,\mathbf{y}_i) = \kappa(\mathbf{y}_i,\mathbf{y}_i) - \frac{2}{N}\mathbf{1}_N^T\boldsymbol{k}_{\mathbf{y}_i} + \frac{1}{N^2}\mathbf{1}_N^T\mathbf{G}\mathbf{1}_N$. To compute $\widetilde{\phi}(\mathbf{Y}_\ell)^T\widetilde{\phi}(\mathbf{y}_i)$, we define $\phi(\mathbf{Y}_\ell) = [\phi(\mathbf{y}_i) : i \in \mathbf{c}_\ell]$ and let $\boldsymbol{\psi}_\ell(\mathbf{y}_i) = [\kappa(\mathbf{y}_{\mathbf{c}_\ell(1)},\mathbf{y}_i),\dots,\kappa(\mathbf{y}_{\mathbf{c}_\ell(N_\ell)},\mathbf{y}_i)]^T \in \mathbb{R}^{N_\ell}$ be a vector with elements given by the inner products between $\phi(\mathbf{y}_i)$ and columns of $\phi(\mathbf{Y}_\ell)$. Then $\widetilde{\boldsymbol{\psi}}_\ell(\mathbf{y}_i) = \widetilde{\phi}(\mathbf{Y}_\ell)^T\widetilde{\phi}(\mathbf{y}_i) = \boldsymbol{\psi}_\ell(\mathbf{y}_i) - \frac{1}{N}\mathbf{1}_{N_\ell}\mathbf{1}_N^T\boldsymbol{k}_{\mathbf{y}_i} - \frac{1}{N}[\mathbf{G}]_{\mathbf{c}_\ell,:}\mathbf{1}_N + \frac{1}{N^2}\mathbf{1}_{N_\ell}\mathbf{1}_N^T\mathbf{G}\mathbf{1}_N$. Therefore, we can write $\|\phi(\mathbf{y}_i) - P_{\mathcal{S}_\ell}\phi(\mathbf{y}_i)\|_2^2 = \widetilde{\kappa}(\mathbf{y}_i,\mathbf{y}_i) - \|\mathbf{E}_\ell^T\widetilde{\boldsymbol{\psi}}_\ell(\mathbf{y}_i)\|_2^2$. Also, we have

$$\begin{aligned} &d_u^2(\mathcal{S}_\ell,\mathcal{S}_p) \\ &= \|\mathbf{D}_\ell - P_{\mathcal{S}_p}\mathbf{D}_\ell\|_F^2 = s - \mathrm{tr}(\mathbf{D}_\ell^T\mathbf{D}_p\mathbf{D}_p^T\mathbf{D}_\ell) \\ &= s - \mathrm{tr}\Big[(\widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{E}_\ell)^T\widetilde{\phi}(\mathbf{Y}_p)\mathbf{E}_p(\widetilde{\phi}(\mathbf{Y}_p)\mathbf{E}_p)^T\widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{E}_\ell\Big] \\ &= s - \mathrm{tr}\Big[\mathbf{E}_\ell^T[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_p}\mathbf{E}_p\mathbf{E}_p^T[\widetilde{\mathbf{G}}]_{\mathbf{c}_p,\mathbf{c}_\ell}\mathbf{E}_\ell\Big], \end{aligned} \tag{18}$$

where $[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_p} = \widetilde{\phi}(\mathbf{Y}_\ell)^T\widetilde{\phi}(\mathbf{Y}_p)$ denotes the centered inter-subspace kernel matrix between $\mathcal{S}_\ell$ and $\mathcal{S}_p$.

Now we are ready to describe our algorithm in detail. Similar to MiCUSaL, we minimize (17) by alternating between $(i)$ minimizing $F_3(\mathbf{D},\mathbf{W})$ over $\mathbf{W}$ for a fixed $\mathbf{D}$ (the *kernel subspace assignment* step) and $(ii)$ minimizing $F_3(\mathbf{D},\mathbf{W})$ over $\mathbf{D}$ for a fixed $\mathbf{W}$ (the *kernel subspace update* step). To begin this alternate optimization strategy, we start by initializing the orthonormal basis of each subspace. As discussed earlier, the orthonormal basis $\mathbf{D}_\ell$ of $\mathcal{S}_\ell$ can be represented as $\mathbf{D}_\ell = \widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{E}_\ell$ and we can compute $\mathbf{E}_\ell$ explicitly by using $[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}$. Therefore the initialization of $\mathbf{D}_\ell$ is equivalent to initializing $\mathbf{c}_\ell$. Note that any $s$ linear independent vectors define an $s$-dimensional subspace. Therefore, to initialize $\mathbf{c}_\ell$, we need to choose $s$ samples in the training set such that the $\phi$-mapped "images" of these training samples are linearly independent in the feature space. Our selection of $s$ samples is based on the intuition that the inner products between samples that lie in the same subspace in the feature space will be typically large [36]. Our initialization procedure then involves greedily selecting a new sample $\mathbf{y}_{i^*}$ from the training data in each step such that the sum of the inner products between $\widetilde{\phi}(\mathbf{y}_{i^*})$ and the data points already in $\widetilde{\phi}(\mathbf{Y}_\ell)$ is the largest, and then setting $\mathbf{Y}_\ell = \mathbf{Y}_\ell \cup \mathbf{y}_{i^*}$. We list our initialization method in Algorithm 4, referred to as *greedy kernel initial-orthogonalization procedure* (GKIOP). Based on the assumption that all the $\phi(\mathbf{y}_i)$'s are linearly independent, it is guaranteed that $\widetilde{\phi}(\mathbf{Y}_\ell)$ can define an $s$-dimensional subspace $\mathcal{S}_\ell$. Note that $\bigcap_{\ell=1}^L \mathbf{c}_\ell = \emptyset$ and we compute $\mathbf{E}_\ell$ by $\mathbf{E}_\ell = \mathbf{U}_\ell\boldsymbol{\Sigma}_\ell^{-\frac{1}{2}}$, where $[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell} = \mathbf{U}_\ell\boldsymbol{\Sigma}_\ell\mathbf{U}_\ell^T$. Since $\mathbf{D}_\ell = \widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{E}_\ell$, it is easy to convince oneself that $\mathbf{D}_\ell^T\mathbf{D}_\ell = \mathbf{I}_s$ in this case.

We now move onto the kernel subspace assignment step. When $\mathbf{D}$ (equivalently, $\mathbf{c}_\ell$'s and $\mathbf{E}_\ell$'s) is fixed, kernel subspace assignment corresponds to first solving $\forall i = 1,\dots,N$,

$$\begin{aligned} l_i &= \arg\min_{\ell=1,\dots,L} \|\widetilde{\phi}(\mathbf{y}_i) - P_{\mathcal{S}_\ell}\widetilde{\phi}(\mathbf{y}_i)\|_2^2 \\ &= \arg\min_{\ell=1,\dots,L} \widetilde{\kappa}(\mathbf{y}_i,\mathbf{y}_i) - \|\mathbf{E}_\ell^T\widetilde{\boldsymbol{\psi}}_\ell(\mathbf{y}_i)\|_2^2, \end{aligned} \tag{19}$$

and then setting $w_{l_i,i} = 1$ and $w_{\ell,i} = 0\ \forall \ell \neq l_i$. Next, for the kernel subspace update stage, since $\mathbf{W}$ is fixed, all the $\mathbf{c}_\ell$'s and $\mathbf{Y}_\ell$'s are fixed. By writing $\mathbf{D}_\ell = \widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{E}_\ell$, minimization of (17) for a fixed $\mathbf{W}$ can be written as a function of $\mathbf{E}_\ell$'s as follows:

$$\begin{aligned} \min_{\{\mathbf{E}_\ell\}} f_3(\mathbf{E}_1,\dots,\mathbf{E}_L) = &\sum_{\substack{\ell,p=1\\ \ell\neq p}}^L \|\widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{E}_\ell - P_{\mathcal{S}_p}(\widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{E}_\ell)\|_F^2 \\ &+ \lambda\sum_{\ell=1}^L \big(\|\widetilde{\phi}(\mathbf{Y}_\ell)\|_F^2 - \|\mathbf{E}_\ell^T\widetilde{\phi}(\mathbf{Y}_\ell)^T\widetilde{\phi}(\mathbf{Y}_\ell)\|_F^2\big) \\ \text{s.t.}\quad &\mathbf{E}_\ell^T[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}\mathbf{E}_\ell = \mathbf{I}_s,\ \ell = 1,2,\dots,L. \end{aligned} \tag{20}$$

Instead of updating all the $\mathbf{E}_\ell$'s simultaneously, which is again a difficult optimization problem, we use BCD to minimize $f_3$ and update $\mathbf{E}_\ell$'s sequentially. Unlike MC-UoS learning, however, we have to be careful here since the number of samples in $\mathbf{Y}$ that belong to $\mathbf{Y}_\ell$ (i.e., $N_\ell$) may change after each subspace assignment step. In particular, we first need to initialize all the $\mathbf{E}_\ell$'s such that $\mathbf{E}_\ell \in \mathbb{R}^{N_\ell\times s}$ and $\mathbf{E}_\ell^T[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}\mathbf{E}_\ell = \mathbf{I}_s$. To do so, we again apply eigen decomposition of $[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell} = \mathbf{U}_\ell\boldsymbol{\Sigma}_\ell\mathbf{U}_\ell^T$ with the diagonal entries of $\boldsymbol{\Sigma}_\ell$ in nonincreasing order. Then we define $\boldsymbol{\mathcal{I}}_s = \{1,\dots,s\}$ and $\mathbf{E}_\ell = [\mathbf{U}_\ell]_{:,\boldsymbol{\mathcal{I}}_s}[\boldsymbol{\Sigma}_\ell]_{\boldsymbol{\mathcal{I}}_s,\boldsymbol{\mathcal{I}}_s}^{-\frac{1}{2}}$. After this bases initialization step, we are ready to update $\mathbf{E}_\ell$'s sequentially and after some manipulations, each BCD subproblem of (20) can be expressed as

$$\begin{aligned} \mathbf{E}_\ell &= \arg\min_{\mathbf{Q}:\mathbf{Q}^T[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}\mathbf{Q}=\mathbf{I}_s} \sum_{p\neq\ell} \|\widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{Q} - P_{\mathcal{S}_p}(\widetilde{\phi}(\mathbf{Y}_\ell)\mathbf{Q})\|_F^2 \\ &\qquad + \frac{\lambda}{2}(\|\widetilde{\phi}(\mathbf{Y}_\ell)\|_F^2 - \|\mathbf{Q}^T\widetilde{\phi}(\mathbf{Y}_\ell)^T\widetilde{\phi}(\mathbf{Y}_\ell)\|_F^2) \\ &= \arg\max_{\mathbf{Q}:\mathbf{Q}^T[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}\mathbf{Q}=\mathbf{I}_s} \mathrm{tr}(\mathbf{Q}^T\mathbf{A}_\ell\mathbf{Q}), \end{aligned} \tag{21}$$

where $\mathbf{A}_\ell = \sum_{p\neq\ell}[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_p}\mathbf{E}_p\mathbf{E}_p^T[\widetilde{\mathbf{G}}]_{\mathbf{c}_p,\mathbf{c}_\ell} + \frac{\lambda}{2}[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}^2$ is a symmetric matrix of dimension $N_\ell \times N_\ell$. Note that (21) has a similar intuitive interpretation as (8). When $\lambda = 0$, (21) reduces to the problem of finding a subspace that is closest to

**Algorithm 5:** Metric-Constrained Kernel Union-of-Subspaces Learning (MC-KUSaL)

**Input:** Training data $\mathbf{Y} \in \mathbb{R}^{m\times N}$, number and dimension of subspaces $L$ and $s$, kernel function $\kappa$ and parameter $\lambda$.

Compute kernel matrix $\mathbf{G}$: $g_{i,j} \leftarrow \kappa(\mathbf{y}_i, \mathbf{y}_j)$.
$\widetilde{\mathbf{G}} \leftarrow \mathbf{G} - \mathbf{H}_N\mathbf{G} - \mathbf{G}\mathbf{H}_N + \mathbf{H}_N\mathbf{G}\mathbf{H}_N$.
Initialize $\{\mathbf{c}_\ell\}_{\ell=1}^L$ and $\{\mathbf{E}_\ell\}_{\ell=1}^L$ using GKIOP (Algorithm 4).
**while** stopping rule **do**
  **for** $i = 1$ to $N$ (*Kernel Subspace Assignment*) **do**
    $l_i \leftarrow \arg\min_\ell \widetilde{\kappa}(\mathbf{y}_i, \mathbf{y}_i) - \|\mathbf{E}_\ell^T \widetilde{\boldsymbol{\psi}}_\ell(\mathbf{y}_i)\|_2^2$.
    $w_{l_i,i} \leftarrow 1$ and $\forall \ell \neq l_i,\ w_{\ell,i} \leftarrow 0$.
  **end for**
  **for** $\ell = 1$ to $L$ (*Kernel Bases Initialization*) **do**
    $\mathbf{c}_\ell \leftarrow \{i \in \{1,\dots,N\} : w_{\ell,i} = 1\}$ and $N_\ell \leftarrow |\mathbf{c}_\ell|$.
    Eigen decomposition of $[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell} = \mathbf{U}_\ell\boldsymbol{\Sigma}_\ell\mathbf{U}_\ell^T$, with the diagonal elements of $\boldsymbol{\Sigma}_\ell$ in nonincreasing order.
    $\mathbf{E}_\ell \leftarrow [\mathbf{U}_\ell]_{:,\boldsymbol{\mathcal{I}}_s}[\boldsymbol{\Sigma}_\ell]^{-\frac{1}{2}}_{\boldsymbol{\mathcal{I}}_s,\boldsymbol{\mathcal{I}}_s}$.
  **end for**
  **while** stopping rule **do**
    **for** $\ell = 1$ to $L$ (*Kernel Subspace Update*) **do**
      $\mathbf{A}_\ell \leftarrow \sum_{p\neq\ell}[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_p}\mathbf{E}_p\mathbf{E}_p^T[\widetilde{\mathbf{G}}]_{\mathbf{c}_p,\mathbf{c}_\ell} + \frac{\lambda}{2}[\widetilde{\mathbf{G}}]^2_{\mathbf{c}_\ell,\mathbf{c}_\ell}$.
      $\mathbf{E}_\ell \leftarrow$ Eigenvectors corresponding to the $s$-largest eigenvalues for the generalized problem $\mathbf{A}_\ell\mathbf{b} = \zeta[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}\mathbf{b}$ such that $\mathbf{E}_\ell^T[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}\mathbf{E}_\ell = \mathbf{I}_s$.
    **end for**
  **end while**
**end while**

**Output:** $\{N_\ell \in \mathbb{N}\}_{\ell=1}^L$, $\{\mathbf{c}_\ell\}_{\ell=1}^L$ and $\{\mathbf{E}_\ell \in \mathbb{R}^{N_\ell\times s}\}_{\ell=1}^L$.

the remaining $L-1$ subspaces in the feature space. When $\lambda = \infty$, (21) reduces to the kernel PCA problem [29]. Since $[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}$ is a positive definite matrix, it again follows from [53] that the trace of $\mathbf{E}_\ell^T\mathbf{A}_\ell\mathbf{E}_\ell$ is maximized when $\mathbf{E}_\ell = [\mathbf{b}_1,\dots,\mathbf{b}_s]$ is the set of eigenvectors associated with the $s$-largest eigenvalues for the generalized problem $\mathbf{A}_\ell\mathbf{b} = \zeta[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}\mathbf{b}$, with $\mathbf{E}_\ell^T[\widetilde{\mathbf{G}}]_{\mathbf{c}_\ell,\mathbf{c}_\ell}\mathbf{E}_\ell = \mathbf{I}_s$. The whole algorithm can be detailed in Algorithm 5, which we refer to as *metric-constrained kernel union-of-subspaces learning* (MC-KUSaL). An important thing to notice here is that the complexity of MC-KUSaL does not scale with the dimensionality of the feature space $\mathcal{F}$ owing to our use of the kernel trick.

### *B. MC-KUoS Learning Using Missing Data*

In this section, we focus on MC-KUoS learning for the case of training data having missing entries in the input space. Our setup is similar to the one in Sec. III-C. That is, for $i = 1,\dots,N$, we observe $\mathbf{y}_i$ only at locations $\boldsymbol{\Omega}_i \subset \{1,\dots,m\}$. In the following, the resulting observed vector of $\mathbf{y}_i$ is denoted by $[\mathbf{y}_i]_{\boldsymbol{\Omega}_i} \in \mathbb{R}^{|\boldsymbol{\Omega}_i|}$. Also, we assume that the observed indices of each signal, $\boldsymbol{\Omega}_i$, are drawn uniformly at random with replacement from $\{1,\dots,m\}$. Note that the results derived in here can also be translated to the case of sampling $\boldsymbol{\Omega}_i$ without replacement (we refer the reader to [62, Lemma 1] as an example). Given the missing data aspect of this setup and the kernel trick, it is clear that we cannot apply the method in Sec. III-C for MC-KUoS learning. However, as described in Sec. IV-A, the solution to the MC-KUoS learning problem using complete data only requires computations of the inner products in $\mathcal{F}$. In this regard, we propose an estimate of the kernel function value $\kappa(\mathbf{y}_i, \mathbf{y}_j)$ using incomplete data $[\mathbf{y}_i]_{\boldsymbol{\Omega}_i}$ and $[\mathbf{y}_j]_{\boldsymbol{\Omega}_j}$. Mathematically, our goal is to find a proxy function $h(\cdot,\cdot)$ such that $h([\mathbf{y}_i]_{\boldsymbol{\Omega}_i}, [\mathbf{y}_j]_{\boldsymbol{\Omega}_j}) \approx \kappa(\mathbf{y}_i, \mathbf{y}_j)$. To derive this proxy function, we start by considering the relationship between $[\mathbf{y}_i]_{\boldsymbol{\Omega}_i}, [\mathbf{y}_j]_{\boldsymbol{\Omega}_j}$ and $\mathbf{y}_i, \mathbf{y}_j$ in the context of different kernel functions.

We first consider isotropic kernels of the form $\kappa(\mathbf{y}_i, \mathbf{y}_j) = k(\|\mathbf{y}_i - \mathbf{y}_j\|_2^2)$ for our analysis. To begin, we define $\mathbf{z}_{ij}^- = \mathbf{y}_i - \mathbf{y}_j$ and $\boldsymbol{\Omega}_{ij} = \boldsymbol{\Omega}_i \cap \boldsymbol{\Omega}_j$, resulting in $[\mathbf{z}_{ij}^-]_{\boldsymbol{\Omega}_{ij}} = [\mathbf{y}_i]_{\boldsymbol{\Omega}_{ij}} - [\mathbf{y}_j]_{\boldsymbol{\Omega}_{ij}} \in \mathbb{R}^{|\boldsymbol{\Omega}_{ij}|}$. For any vector $\mathbf{z}_{ij}^-$, the authors in [58] define the coherence of a subspace spanned by a vector $\mathbf{z}_{ij}^-$ to be $\mu(\mathbf{z}_{ij}^-) = \frac{m\|\mathbf{z}_{ij}^-\|_\infty^2}{\|\mathbf{z}_{ij}^-\|_2^2}$ and show that $\|[\mathbf{z}_{ij}^-]_{\boldsymbol{\Omega}_{ij}}\|_2^2$ is close to $\frac{|\boldsymbol{\Omega}_{ij}|}{m}\|\mathbf{z}_{ij}^-\|_2^2$ with high probability. Leveraging this result, we can give the following corollary that is essentially due to [58, Lemma 1] by plugging in the definition of $\mathbf{z}_{ij}^-$ and $[\mathbf{z}_{ij}^-]_{\boldsymbol{\Omega}_{ij}}$.

**Corollary 1.** *Let* $\delta > 0$, $\boldsymbol{\Omega}_{ij} = \boldsymbol{\Omega}_i \cap \boldsymbol{\Omega}_j$ *and* $\alpha = \sqrt{\frac{2\mu(\mathbf{y}_i-\mathbf{y}_j)^2}{|\boldsymbol{\Omega}_{ij}|}\log(\frac{1}{\delta})}$. *Then with probability at least* $1-2\delta$,

$$(1-\alpha)\|\mathbf{y}_i - \mathbf{y}_j\|_2^2 \leq \frac{m}{|\boldsymbol{\Omega}_{ij}|}\|[\mathbf{z}_{ij}^-]_{\boldsymbol{\Omega}_{ij}}\|_2^2 \leq (1+\alpha)\|\mathbf{y}_i - \mathbf{y}_j\|_2^2.$$

Using this simple relationship in Corollary 1, we can replace the distance term $\|\mathbf{y}_i - \mathbf{y}_j\|_2^2$ in any isotropic kernel function by $\frac{m}{|\boldsymbol{\Omega}_{ij}|}\|[\mathbf{y}_i]_{\boldsymbol{\Omega}_{ij}} - [\mathbf{y}_j]_{\boldsymbol{\Omega}_{ij}}\|_2^2$ and provide an estimate of its true value $\kappa(\mathbf{y}_i, \mathbf{y}_j)$ using entries of $\mathbf{y}_i$ and $\mathbf{y}_j$ that correspond to $\boldsymbol{\Omega}_{ij}$ only. For example, for the Gaussian kernel $\kappa(\mathbf{y}_i, \mathbf{y}_j) = \exp(-\frac{\|\mathbf{y}_i-\mathbf{y}_j\|_2^2}{c})$ with $c > 0$, we can replace $\kappa(\mathbf{y}_i, \mathbf{y}_j)$ with $h([\mathbf{y}_i]_{\boldsymbol{\Omega}_i}, [\mathbf{y}_j]_{\boldsymbol{\Omega}_j}) = \exp(-\frac{m\|[\mathbf{y}_i]_{\boldsymbol{\Omega}_{ij}} - [\mathbf{y}_j]_{\boldsymbol{\Omega}_{ij}}\|_2^2}{|\boldsymbol{\Omega}_{ij}|c})$ in our algorithms. In this case, the following result provides bounds for estimation of the Gaussian kernel value.

**Theorem 1.** *Let* $\delta > 0$, $\boldsymbol{\Omega}_{ij} = \boldsymbol{\Omega}_i \cap \boldsymbol{\Omega}_j$ *and* $\alpha = \sqrt{\frac{2\mu(\mathbf{y}_i-\mathbf{y}_j)^2}{|\boldsymbol{\Omega}_{ij}|}\log(\frac{1}{\delta})}$. *Then for a Gaussian kernel* $\kappa(\mathbf{y}_i, \mathbf{y}_j)$, *with probability at least* $1-2\delta$, *we have*

$$h([\mathbf{y}_i]_{\boldsymbol{\Omega}_i}, [\mathbf{y}_j]_{\boldsymbol{\Omega}_j})^{\frac{1}{1-\alpha}} \leq \kappa(\mathbf{y}_i, \mathbf{y}_j) \leq h([\mathbf{y}_i]_{\boldsymbol{\Omega}_i}, [\mathbf{y}_j]_{\boldsymbol{\Omega}_j})^{\frac{1}{1+\alpha}}.$$

We skip the proof of this theorem since it is elementary. We also note here that $h([\mathbf{y}_i]_{\boldsymbol{\Omega}_i}, [\mathbf{y}_i]_{\boldsymbol{\Omega}_i}) = \kappa(\mathbf{y}_i, \mathbf{y}_i) = 1$ as a special case for Gaussian kernels.

Next, we consider dot product kernels of the form $\kappa(\mathbf{y}_i, \mathbf{y}_j) = k(\langle \mathbf{y}_i, \mathbf{y}_j\rangle)$, where we again need to estimate $\langle \mathbf{y}_i, \mathbf{y}_j\rangle$ using entries of $\mathbf{y}_i$ and $\mathbf{y}_j$ corresponding to $\boldsymbol{\Omega}_{ij}$ only. In order to find an estimator of $\langle \mathbf{y}_i, \mathbf{y}_j\rangle$, we define $\mathbf{z}_{ij}^* = \mathbf{y}_i \circ \mathbf{y}_j \in \mathbb{R}^m$ to be the coordinate-wise product of $\mathbf{y}_i$ and $\mathbf{y}_j$. This means that $\langle \mathbf{y}_i, \mathbf{y}_j\rangle$ and $\langle [\mathbf{y}_i]_{\boldsymbol{\Omega}_{ij}}, [\mathbf{y}_j]_{\boldsymbol{\Omega}_{ij}}\rangle$ equal the sum of all the entries of $\mathbf{z}_{ij}^*$ and $[\mathbf{z}_{ij}^*]_{\boldsymbol{\Omega}_{ij}} \in \mathbb{R}^{|\boldsymbol{\Omega}_{ij}|}$, respectively. We now have the following lemma that describes deviation of the estimated inner product between $\mathbf{y}_i$ and $\mathbf{y}_j$.

**Lemma 1.** *Let* $\delta > 0$, $\boldsymbol{\Omega}_{ij} = \boldsymbol{\Omega}_i \cap \boldsymbol{\Omega}_j$ *and* $\beta =$

$\sqrt{\frac{2m^2\|\mathbf{y}_i\circ\mathbf{y}_j\|_\infty^2}{|\mathbf{\Omega}_{ij}|}\log(\frac{1}{\delta})}$. *Then with probability at least* $1-2\delta$,

$$\langle \mathbf{y}_i, \mathbf{y}_j\rangle - \beta \leq \frac{m}{|\mathbf{\Omega}_{ij}|}\langle [\mathbf{y}_i]_{\mathbf{\Omega}_{ij}}, [\mathbf{y}_j]_{\mathbf{\Omega}_{ij}}\rangle \leq \langle \mathbf{y}_i, \mathbf{y}_j\rangle + \beta. \quad (22)$$

*Proof:* See Appendix A. ∎

The above lemma establishes that $\langle [\mathbf{y}_i]_{\mathbf{\Omega}_{ij}}, [\mathbf{y}_j]_{\mathbf{\Omega}_{ij}}\rangle$ is close to $\frac{|\mathbf{\Omega}_{ij}|}{m}\langle \mathbf{y}_i, \mathbf{y}_j\rangle$ with high probability. We once again use this relationship and give an estimate of the corresponding kernel function value. For example, for the polynomial kernel $\kappa(\mathbf{y}_i, \mathbf{y}_j) = (\langle \mathbf{y}_i, \mathbf{y}_j\rangle + c)^d$ with $d > 0$ and $c \geq 0$, we have $h([\mathbf{y}_i]_{\mathbf{\Omega}_i}, [\mathbf{y}_j]_{\mathbf{\Omega}_j}) = (\frac{m}{|\mathbf{\Omega}_{ij}|}\langle [\mathbf{y}_i]_{\mathbf{\Omega}_{ij}}, [\mathbf{y}_j]_{\mathbf{\Omega}_{ij}}\rangle + c)^d$. To analyze the bounds on estimated kernel function value in this case, notice that if (22) holds and $d$ is odd, we will have

$$\begin{aligned}(\langle \mathbf{y}_i, \mathbf{y}_j\rangle - \beta + c)^d &\leq (\frac{m}{|\mathbf{\Omega}_{ij}|}\langle [\mathbf{y}_i]_{\mathbf{\Omega}_{ij}}, [\mathbf{y}_j]_{\mathbf{\Omega}_{ij}}\rangle + c)^d \\ &\leq (\langle \mathbf{y}_i, \mathbf{y}_j\rangle + \beta + c)^d.\end{aligned}$$

But the above inequalities cannot be guaranteed to hold when $d$ is even. Using this, we trivially obtain the theorem below, as a counterpart of Theorem 1, for polynomial kernels.

**Theorem 2.** *Let* $\delta > 0$, $\mathbf{\Omega}_{ij} = \mathbf{\Omega}_i \cap \mathbf{\Omega}_j$ *and* $\beta = \sqrt{\frac{2m^2\|\mathbf{y}_i\circ\mathbf{y}_j\|_\infty^2}{|\mathbf{\Omega}_{ij}|}\log(\frac{1}{\delta})}$. *Then for a polynomial kernel* $\kappa(\mathbf{y}_i, \mathbf{y}_j)$ *with an odd degree* $d$*, with probability at least* $1-2\delta$*, we have*

$$\begin{aligned}&(h([\mathbf{y}_i]_{\mathbf{\Omega}_i}, [\mathbf{y}_j]_{\mathbf{\Omega}_j})^{\frac{1}{d}} - \beta)^d \leq \kappa(\mathbf{y}_i, \mathbf{y}_j) \\ \textit{and}\quad &\kappa(\mathbf{y}_i, \mathbf{y}_j) \leq (h([\mathbf{y}_i]_{\mathbf{\Omega}_i}, [\mathbf{y}_j]_{\mathbf{\Omega}_j})^{\frac{1}{d}} + \beta)^d.\end{aligned}$$

Based on the discussion above, we can estimate the kernel function value $\kappa(\mathbf{y}_i, \mathbf{y}_j)$ using the associated proxy function $h(\cdot,\cdot)$ that utilizes entries of $\mathbf{y}_i$ and $\mathbf{y}_j$ belonging to $\mathbf{\Omega}_{ij}$ only. Thus, we can compute the estimated kernel matrix $\mathbf{G} \in \mathbb{R}^{N\times N}$ as $g_{i,j} = h([\mathbf{y}_i]_{\mathbf{\Omega}_i}, [\mathbf{y}_j]_{\mathbf{\Omega}_j})$ in the case of missing data. But the positive definiteness of $\mathbf{G}$ is not guaranteed in this case. We therefore first need to find a positive definite matrix $\widehat{\mathbf{G}} \approx \mathbf{G}$ before we can carry on with MC-KUoS learning in this setting. To deal with this issue, we begin with eigen decomposition of $\mathbf{G} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where $\mathbf{\Lambda} = \mathrm{diag}\{\lambda_{\mathbf{G}}^{(1)}, \ldots, \lambda_{\mathbf{G}}^{(N)}\}$ contains eigenvalues of $\mathbf{G}$. The resulting approximated kernel matrix $\widehat{\mathbf{G}}$ that is "closest" to $\mathbf{G}$ can then be calculated by $\widehat{\mathbf{G}} = \mathbf{U}\widehat{\mathbf{\Lambda}}\mathbf{U}^T$, where $\widehat{\mathbf{\Lambda}} = \mathrm{diag}\{\lambda_{\widehat{\mathbf{G}}}^{(1)}, \ldots, \lambda_{\widehat{\mathbf{G}}}^{(N)}\}$ and each $\lambda_{\widehat{\mathbf{G}}}^{(i)}$, $i = 1, \ldots, N$, is defined as

$$\lambda_{\widehat{\mathbf{G}}}^{(i)} = \begin{cases} \lambda_{\mathbf{G}}^{(i)}, & \lambda_{\mathbf{G}}^{(i)} > 0 \\ \delta_{min}, & \lambda_{\mathbf{G}}^{(i)} = 0 \\ -\lambda_{\mathbf{G}}^{(i)}, & \lambda_{\mathbf{G}}^{(i)} < 0. \end{cases}$$

Here, $\delta_{min} > 0$ is a predefined (small) parameter. Using the above procedure, one can obtain a positive definite matrix $\widehat{\mathbf{G}}$ such that $\widehat{g}_{i,j} \approx \kappa(\mathbf{y}_i, \mathbf{y}_j)$ and use it for MC-UoS learning in the feature space. Effectively, MC-KUoS learning in the presence of missing data also relies on Algorithm 5, with the difference being that we use $\widehat{g}_{i,j}$, obtained from $h([\mathbf{y}_i]_{\mathbf{\Omega}_i}, [\mathbf{y}_j]_{\mathbf{\Omega}_j})$, in lieu of $\kappa(\mathbf{y}_i, \mathbf{y}_j)$ in the overall learning process, which includes both kernel subspace assignment and kernel subspace update stages. We dub this approach *robust MC-KUoS learning* (rMC-KUSaL). We conclude this section by noting that we can also robustify classical kernel PCA by using $\widehat{\mathbf{G}}$ as a means of performing kernel PCA with missing data, which we call rKPCA in our experiments.

### C. Pre-Image Reconstruction

Thus far in this section, we have discussed MC-UoS learning in the kernel space with complete and missing data using the kernel trick. Now suppose we are given a new noisy (test) sample $\mathbf{z} = \mathbf{x} + \boldsymbol{\xi} \in \mathbb{R}^m$, where $\boldsymbol{\xi}$ is a noise term and $\widetilde{\phi}(\mathbf{x}) = \phi(\mathbf{x}) - \overline{\boldsymbol{\phi}}$ belongs to one of the subspaces in $\mathcal{M}_L$ (i.e., $\widetilde{\phi}(\mathbf{x}) \in \mathcal{S}_\tau, \tau \in \{1, \ldots, L\}$). In most information processing tasks, one needs to first find a representation of this sample $\mathbf{z}$ in terms of the learned MC-KUoS, which is akin to "denoising" $\mathbf{z}$. The "denoised sample" in the feature space is the projection of $\phi(\mathbf{z})$ onto $\mathcal{S}_\tau$, which is given by $P_{\mathcal{S}_\tau}\phi(\mathbf{z}) = \mathbf{D}_\tau\mathbf{D}_\tau^T\widetilde{\phi}(\mathbf{z}) + \overline{\boldsymbol{\phi}}$ with $\widetilde{\phi}(\mathbf{z}) = \phi(\mathbf{z}) - \overline{\boldsymbol{\phi}}$. However, in order to visualize the "denoised" sample in the ambient space, we often need to project $P_{\mathcal{S}_\tau}\phi(\mathbf{z})$ onto the input space in many applications [9], [63], which is termed *pre-image reconstruction*. In this section, we consider the problem of pre-image reconstruction based on the MC-KUoS model.

Mathematically, the problem of pre-image reconstruction can be stated as follows. We are given $\mathbf{z} \in \mathbb{R}^m$ and we are interested in finding $\widehat{\mathbf{z}} \in \mathbb{R}^m$ whose mapping to the feature space is closest to the projection of $\phi(\mathbf{z})$ onto the learned MC-UoS in $\mathcal{F}$. This involves first finding the index $\tau$ such that $\tau = \arg\min_\ell \|\widetilde{\phi}(\mathbf{z}) - P_{\mathcal{S}_\ell}\widetilde{\phi}(\mathbf{z})\|_2^2$, which can be easily done using the kernel subspace assignment step described in (19). Next, we need to solve $\widehat{\mathbf{z}} = \arg\min_{\boldsymbol{\varrho}\in\mathbb{R}^m} \|\phi(\boldsymbol{\varrho}) - P_{\mathcal{S}_\tau}\phi(\mathbf{z})\|_2^2$. To solve this problem, we leverage the ideas in [50], [64] that only use feature-space distances to find $\widehat{\mathbf{z}}$ (equivalently, to find the pre-image of $P_{\mathcal{S}_\tau}\phi(\mathbf{z})$). We first study this problem when the training samples $\mathbf{Y}$ are complete.

*1) Pre-Image Reconstruction Using Complete Data:* We first calculate the squared "feature distance" between $P_{\mathcal{S}_\tau}\phi(\mathbf{z})$ and any $\phi(\mathbf{y}_i)$, $i = 1 \ldots, N$, defined as [50]

$$\begin{aligned}&d_{\mathcal{F}}^2(\phi(\mathbf{y}_i), P_{\mathcal{S}_\tau}\phi(\mathbf{z})) \\ &= \|P_{\mathcal{S}_\tau}\phi(\mathbf{z})\|_2^2 + \|\phi(\mathbf{y}_i)\|_2^2 - 2(P_{\mathcal{S}_\tau}\phi(\mathbf{z}))^T\phi(\mathbf{y}_i). \quad (23)\end{aligned}$$

Notice that $\|P_{\mathcal{S}_\tau}\phi(\mathbf{z})\|_2^2$ and $(P_{\mathcal{S}_\tau}\phi(\mathbf{z}))^T\phi(\mathbf{y}_i)$ can be calculated in terms of kernel representation as follows:

$$\begin{aligned}&\|P_{\mathcal{S}_\tau}\phi(\mathbf{z})\|_2^2 \\ &= \widetilde{\phi}(\mathbf{z})^T\mathbf{D}_\tau\mathbf{D}_\tau^T\widetilde{\phi}(\mathbf{z}) + \overline{\boldsymbol{\phi}}^T\overline{\boldsymbol{\phi}} + 2\widetilde{\phi}(\mathbf{z})^T\mathbf{D}_\tau\mathbf{D}_\tau^T\overline{\boldsymbol{\phi}} \\ &= \widetilde{\phi}(\mathbf{z})^T\widetilde{\phi}(\mathbf{Y}_\tau)\mathbf{E}_\tau\mathbf{E}_\tau^T\widetilde{\phi}(\mathbf{Y}_\tau)^T\widetilde{\phi}(\mathbf{z}) + \frac{1}{N^2}\mathbf{1}_N^T\mathbf{G}\mathbf{1}_N \\ &\quad + \frac{2}{N}\widetilde{\phi}(\mathbf{z})^T\widetilde{\phi}(\mathbf{Y}_\tau)\mathbf{E}_\tau\mathbf{E}_\tau^T\widetilde{\phi}(\mathbf{Y}_\tau)^T\phi(\mathbf{Y})\mathbf{1}_N \\ &= \widetilde{\boldsymbol{\psi}}_\tau(\mathbf{z})^T\mathbf{E}_\tau\mathbf{E}_\tau^T\Big(\widetilde{\boldsymbol{\psi}}_\tau(\mathbf{z}) + \frac{2}{N}[\mathbf{G}]_{\mathbf{c}_\tau,:}\mathbf{1}_N - \frac{2}{N^2}\mathbf{1}_{N_\tau}\mathbf{1}_N^T\mathbf{G}\mathbf{1}_N\Big) \\ &\quad + \frac{1}{N^2}\mathbf{1}_N^T\mathbf{G}\mathbf{1}_N,\end{aligned}$$

and

$$\begin{aligned}&(P_{\mathcal{S}_\tau}\phi(\mathbf{z}))^T\phi(\mathbf{y}_i) \\ &= \widetilde{\boldsymbol{\psi}}_\tau(\mathbf{z})^T\mathbf{E}_\tau\mathbf{E}_\tau^T\Big(\boldsymbol{\psi}_\tau(\mathbf{y}_i) - \frac{1}{N}\mathbf{1}_{N_\tau}\mathbf{1}_N^T\boldsymbol{k}_{\mathbf{y}_i}\Big) + \frac{1}{N}\mathbf{1}_N^T\boldsymbol{k}_{\mathbf{y}_i}.\end{aligned}$$

Therefore, (23) becomes

$$\begin{aligned}
&d^2_{\mathcal{F}}(\phi(\mathbf{y}_i), P_{\mathcal{S}_\tau}\phi(\mathbf{z})) \\
&= \widetilde{\boldsymbol{\psi}}_\tau(\mathbf{z})^T\mathbf{E}_\tau\mathbf{E}_\tau^T\Big(\widetilde{\boldsymbol{\psi}}_\tau(\mathbf{z}) + \frac{2}{N}[\mathbf{G}]_{\mathbf{c}_\tau,:}\mathbf{1}_N - 2\boldsymbol{\psi}_\tau(\mathbf{y}_i) \\
&\quad - \frac{2}{N^2}\mathbf{1}_{N_\tau}\mathbf{1}_N^T\mathbf{G}\mathbf{1}_N + \frac{2}{N}\mathbf{1}_{N_\tau}\mathbf{1}_N^T\boldsymbol{k}_{\mathbf{y}_i}\Big) + g_{i,i} \\
&\quad + \frac{1}{N^2}\mathbf{1}_N^T\mathbf{G}\mathbf{1}_N - \frac{2}{N}\mathbf{1}_N^T\boldsymbol{k}_{\mathbf{y}_i}
\end{aligned}$$

with $g_{i,i} = \kappa(\mathbf{y}_i, \mathbf{y}_i)$.

We now describe our method for pre-image reconstruction using the Gaussian kernel $\kappa(\mathbf{y}_i, \mathbf{y}_j) = \exp(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|_2^2}{c})$ first. In this case, the problem of minimizing $\|\phi(\widehat{\mathbf{z}}) - P_{\mathcal{S}_\tau}\phi(\mathbf{z})\|_2^2$ is equivalent to maximizing the function $\rho(\widehat{\mathbf{z}}) = (P_{\mathcal{S}_\tau}\phi(\mathbf{z}))^T\phi(\widehat{\mathbf{z}})$ [9], whose extremum can be obtained by setting $\nabla_{\widehat{\mathbf{z}}}\rho = 0$, where $\nabla_{\widehat{\mathbf{z}}}\rho$ denotes the gradient of $\rho$ with respect to $\widehat{\mathbf{z}}$. To do so, we express $\rho(\widehat{\mathbf{z}})$ as

$$\begin{aligned}
&\rho(\widehat{\mathbf{z}}) \\
&= (\mathbf{D}_\tau\mathbf{D}_\tau^T\widetilde{\phi}(\mathbf{z}) + \overline{\boldsymbol{\phi}})^T\phi(\widehat{\mathbf{z}}) \\
&= \widetilde{\phi}(\mathbf{z})^T\widetilde{\phi}(\mathbf{Y}_\tau)\mathbf{E}_\tau\mathbf{E}_\tau^T\widetilde{\phi}(\mathbf{Y}_\tau)^T\phi(\widehat{\mathbf{z}}) + \frac{1}{N}\mathbf{1}_N^T\phi(\mathbf{Y})^T\phi(\widehat{\mathbf{z}}) \\
&= \widetilde{\boldsymbol{\psi}}_\tau(\mathbf{z})^T\mathbf{E}_\tau\mathbf{E}_\tau^T(\boldsymbol{\psi}_\tau(\widehat{\mathbf{z}}) - \frac{1}{N}\mathbf{1}_{N_\tau}\mathbf{1}_N^T\boldsymbol{k}_{\widehat{\mathbf{z}}}) + \frac{1}{N}\mathbf{1}_N^T\boldsymbol{k}_{\widehat{\mathbf{z}}} \\
&= \boldsymbol{\zeta}_\tau(\mathbf{z})^T(\boldsymbol{\psi}_\tau(\widehat{\mathbf{z}}) - \frac{1}{N}\mathbf{1}_{N_\tau}\mathbf{1}_N^T\boldsymbol{k}_{\widehat{\mathbf{z}}}) + \frac{1}{N}\mathbf{1}_N^T\boldsymbol{k}_{\widehat{\mathbf{z}}}, \qquad (24)
\end{aligned}$$

where $\boldsymbol{\zeta}_\tau(\mathbf{z}) = \mathbf{E}_\tau\mathbf{E}_\tau^T\widetilde{\boldsymbol{\psi}}_\tau(\mathbf{z}) \in \mathbb{R}^{|N_\tau|}$. Next, we define $\boldsymbol{\chi} = \frac{1}{N}(1 - \boldsymbol{\zeta}_\tau(\mathbf{z})^T\mathbf{1}_{N_\tau})\mathbf{1}_N \in \mathbb{R}^N$ and let $\widehat{\boldsymbol{\chi}}$ be an $N$-dimensional vector such that $[\widehat{\boldsymbol{\chi}}]_{\mathbf{c}_\tau} = [\boldsymbol{\chi}]_{\mathbf{c}_\tau} + \boldsymbol{\zeta}_\tau(\mathbf{z})$ and $[\widehat{\boldsymbol{\chi}}]_{\boldsymbol{\mathcal{I}}_N\setminus\mathbf{c}_\tau} = [\boldsymbol{\chi}]_{\boldsymbol{\mathcal{I}}_N\setminus\mathbf{c}_\tau}$ (recall that $\boldsymbol{\mathcal{I}}_N = \{1, \ldots, N\}$ and $\mathbf{c}_\tau$ contains all the indices of $\widetilde{\phi}(\mathbf{y}_i)$'s that are assigned to $\mathcal{S}_\tau$), which means $\rho(\widehat{\mathbf{z}}) = \widehat{\boldsymbol{\chi}}^T\boldsymbol{k}_{\widehat{\mathbf{z}}} = \sum_{i=1}^N \widehat{\boldsymbol{\chi}}_{(i)}\kappa(\widehat{\mathbf{z}}, \mathbf{y}_i)$. By setting $\nabla_{\widehat{\mathbf{z}}}\rho = 0$, we get

$$\widehat{\mathbf{z}} = \frac{\sum_{i=1}^N \widehat{\boldsymbol{\chi}}_{(i)}\exp(-\|\widehat{\mathbf{z}} - \mathbf{y}_i\|_2^2/c)\mathbf{y}_i}{\sum_{i=1}^N \widehat{\boldsymbol{\chi}}_{(i)}\exp(-\|\widehat{\mathbf{z}} - \mathbf{y}_i\|_2^2/c)}.$$

By using the approximation $P_{\mathcal{S}_\tau}\phi(\mathbf{z}) \approx \phi(\widehat{\mathbf{z}})$ and the relation $\|\widehat{\mathbf{z}} - \mathbf{y}_i\|_2^2 = -c\log(\frac{1}{2}(2 - d^2_{\mathcal{F}}(\phi(\mathbf{y}_i), \phi(\widehat{\mathbf{z}}))))$ [50], a unique pre-image can now be obtained by the following formula:

$$\widehat{\mathbf{z}} = \frac{\sum_{i=1}^N \widehat{\boldsymbol{\chi}}_{(i)}\Big(\frac{1}{2}\big(2 - d^2_{\mathcal{F}}(P_{\mathcal{S}_\tau}\phi(\mathbf{z}), \phi(\mathbf{y}_i))\big)\Big)\mathbf{y}_i}{\sum_{i=1}^N \widehat{\boldsymbol{\chi}}_{(i)}\Big(\frac{1}{2}\big(2 - d^2_{\mathcal{F}}(P_{\mathcal{S}_\tau}\phi(\mathbf{z}), \phi(\mathbf{y}_i))\big)\Big)}. \qquad (25)$$

Next, for the polynomial kernel $\kappa(\mathbf{y}_i, \mathbf{y}_j) = (\langle \mathbf{y}_i, \mathbf{y}_j\rangle + c)^d$ with an odd degree $d$, we can follow a similar procedure and have the following expression for an approximate solution for pre-image reconstruction:

$$\widehat{\mathbf{z}} = \sum_{i=1}^N \widehat{\boldsymbol{\chi}}_{(i)}\Big(\frac{(P_{\mathcal{S}_\tau}\phi(\mathbf{z}))^T\phi(\mathbf{y}_i)}{\|P_{\mathcal{S}_\tau}\phi(\mathbf{z})\|_2^2}\Big)^{\frac{d-1}{d}}\mathbf{y}_i. \qquad (26)$$

*2) Pre-Image Reconstruction Using Missing Data:* We next consider the problem of reconstructing the pre-image of $P_{\mathcal{S}_\tau}\phi(\mathbf{z})$ when the training samples have missing entries. As can be easily seen from (25), the solution of a pre-image for the Gaussian kernel can be written as $\widehat{\mathbf{z}} = \frac{\sum_{i=1}^N e_i\mathbf{y}_i}{\sum_{i=1}^N e_i}$, where $e_i = \widehat{\boldsymbol{\chi}}_{(i)}\big(\frac{1}{2}(2 - d^2_{\mathcal{F}}(P_{\mathcal{S}_\tau}\phi(\mathbf{z}), \phi(\mathbf{y}_i)))\big)$. Similarly, from (26), we can also write the solution of $\widehat{\mathbf{z}}$ to be $\widehat{\mathbf{z}} = \sum_{i=1}^N e_i\mathbf{y}_i$ for the polynomial kernel, where $e_i = \widehat{\boldsymbol{\chi}}_{(i)}\big(\frac{(P_{\mathcal{S}_\tau}\phi(\mathbf{z}))^T\phi(\mathbf{y}_i)}{\|P_{\mathcal{S}_\tau}\phi(\mathbf{z})\|_2^2}\big)^{\frac{d-1}{d}}$ in this case. In words, the pre-image solution is a linear combination of the training data, where the weights $e_i$'s can be explicitly computed using the respective kernel functions. In this regard, as described in Sec. IV-B, for each $i = 1, \ldots, N$, we can estimate $\kappa(\mathbf{z}, \mathbf{y}_i)$ using entries of $\mathbf{z}$ belonging to $\boldsymbol{\Omega}_i$ (i.e., $[\mathbf{z}]_{\boldsymbol{\Omega}_i}$) and $[\mathbf{y}_i]_{\boldsymbol{\Omega}_i}$, where the estimated kernel function value is denoted by $h(\mathbf{z}, [\mathbf{y}_i]_{\boldsymbol{\Omega}_i})$.

Based on the estimated kernel function values $h(\mathbf{z}, [\mathbf{y}_i]_{\boldsymbol{\Omega}_i})$'s, we can then find the solution of $\tau$ such that $\tau = \arg\min_\ell \|\widetilde{\phi}(\mathbf{z}) - P_{\mathcal{S}_\ell}\widetilde{\phi}(\mathbf{z})\|_2^2$, and calculate the weights $e_i$'s ($i = 1, \ldots, N$). Note that unlike the complete data case, we do need to compute the entries of $\widehat{\mathbf{z}}$ separately in this case. To be specific, for the $u$-th entry of $\widehat{\mathbf{z}}$, $u = 1, \ldots, m$, we define $\mathbf{r}_u$ to be the set containing the indices of the samples $\mathbf{y}_i$'s whose $u$-th entry are observed. Then $\widehat{\mathbf{z}}_{(u)} = \frac{\sum_{i\in\mathbf{r}_u} e_i\mathbf{y}_{i(u)}}{(\sum_{i=1}^N e_i)|\mathbf{r}_u|/N}$ for the Gaussian kernel and $\widehat{\mathbf{z}}_{(u)} = \sum_{i\in\mathbf{r}_u} e_i\mathbf{y}_{i(u)}$ for the polynomial kernel. We conclude this section by noting that the methods described in here can also be applied to the case when the test sample $\mathbf{z}$ has missing entries.

## V. Experimental Results

In this section, we present several experimental results demonstrating the effectiveness of our proposed methods for data representation. In particular, we are interested in learning an MC-UoS from complete/missing noisy training data, followed by denoising of complete noisy test samples using the learned geometric structures. In the case of MC-KUoS learning, we evaluate the performance of our algorithms by focusing on $(i)$ denoising of complete noisy test samples, and $(ii)$ clustering of complete/missing training data.

### A. Experiments for MC-UoS Learning

In this section, we examine the effectiveness of MC-UoS learning using Algorithms 1–3. For the complete data experiments, we compare MiCUSaL/aMiCUSaL with several state-of-the-art UoS learning algorithms such as Block-Sparse Dictionary Design (SAC+BK-SVD) [33], $K$-subspace clustering ($K$-sub) [41], Sparse Subspace Clustering (SSC) [16], Robust Sparse Subspace Clustering (RSSC) [37], Robust Subspace Clustering via Thresholding (TSC) [36], as well as with Principal Component Analysis (PCA) [4]. In the case of the UoS learning algorithms, we use codes provided by their respective authors. In the case of SSC, we use the noisy variant of the optimization program in [16] and set $\lambda_z = \alpha_z/\mu_z$ in all experiments, where $\lambda_z$ and $\mu_z$ are as defined in [16, (13) & (14)], while the parameter $\alpha_z$ varies in different experiments. In the case of RSSC, we set $\lambda = 1/\sqrt{s}$ as per [37], while the tuning parameter in TSC is set $q = \max(3, \lceil N/(L \times 20)\rceil)$ when $L$ is provided. For the case of training data having missing entries, we compare the results of rMiCUSaL with $k$-GROUSE [25] and GROUSE [61].[6]

[6]As discussed in [1], we omit the results for SSC with missing data in this paper because it fills in the missing entries with random values, resulting in relatively poor performance for problems with missing data.

In order to generate noisy training and test data in these experiments, we start with sets of "clean" training and test samples, denoted by $\mathbf{X}$ and $\mathbf{X}^{te}$, respectively. We then add white Gaussian noise to these samples to generate noisy training and test samples $\mathbf{Y}$ and $\mathbf{Z}$, respectively. In the following, we use $\sigma_{tr}^2$ and $\sigma_{te}^2$ to denote variance of noise added to training and test samples, respectively. In the missing data experiments, for every fixed noise variance $\sigma_{tr}^2$, we create training (but not test) data with different percentages of missing values, where the number of missing entries is set to be $10\%$, $30\%$ and $50\%$ of the signal dimension. Our reported results are based on random initializations of MiCUSaL and aMiCUSaL algorithms. In this regard, we adopt the following simple approach to mitigate any stability issues that might arise due to random initializations. We perform multiple random initializations for every fixed $\mathbf{Y}$ and $\lambda$, and then retain the learned MC-UoS structure that results in the smallest value of the objective function in (2). We also use a similar approach for selecting the final structures returned by $K$-subspace clustering and Block-Sparse Dictionary Design, with the only difference being that (2) in this case is replaced by the approximation error of training data.

*1) Experiments on Synthetic Data:* In the first set of synthetic experiments, we consider $L = 5$ subspaces of the same dimension $s = 13$ in an $m = 180$-dimensional ambient space. The five subspaces $\mathcal{S}_\ell$'s of $\mathbb{R}^{180}$ are defined by their orthonormal bases $\{\mathbf{T}_\ell \in \mathbb{R}^{m\times s}\}_{\ell=1}^5$ as follows. We start with a random orthonormal basis $\mathbf{T}_1 \in \mathbb{R}^{m\times s}$ and for every $\ell \geq 2$, we set $\mathbf{T}_\ell = \mathrm{orth}(\mathbf{T}_{\ell-1} + t_s \mathbf{W}_\ell)$ where every entry in $\mathbf{W}_\ell \in \mathbb{R}^{m\times s}$ is a uniformly distributed random number between 0 and 1, and $\mathrm{orth}(\cdot)$ denotes the orthogonalization process. The parameter $t_s$ controls the distance between subspaces, and we set $t_s = 0.04$ in these experiments.

After generating the subspaces, we generate a set of $n_\ell$ points from $\mathcal{S}_\ell$ as $\mathbf{X}_\ell = \mathbf{T}_\ell \mathbf{C}_\ell$, where $\mathbf{C}_\ell \in \mathbb{R}^{s\times n_\ell}$ is a matrix whose elements are drawn independently and identically from $\mathcal{N}(0,1)$ distribution. In here, we set $n_1 = n_3 = n_5 = 150$, and $n_2 = n_4 = 100$; hence, $N = 650$. We then stack all the data into a matrix $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_5] = \{\mathbf{x}_i\}_{i=1}^N$ and normalize all the samples to unit $\ell_2$ norms. Test data $\mathbf{X}^{te} \in \mathbb{R}^{m\times N}$ are produced using the same foregoing strategy. Then we add white Gaussian noise with different expected noise power to both $\mathbf{X}$ and $\mathbf{X}^{te}$. Specifically, we set $\sigma_{tr}^2$ to be $0.1$, while $\sigma_{te}^2$ ranges from $0.1$ to $0.5$. We generate $\mathbf{X}$ and $\mathbf{X}^{te}$ 10 times, while Monte Carlo simulations for noisy data are repeated 20 times for every fixed $\mathbf{X}$ and $\mathbf{X}^{te}$. Therefore, the results reported in here correspond to an average of 200 random trials.

We make use of the collection of noisy samples, $\mathbf{Y}$, to learn a union of $L$ subspaces of dimension $s$ and stack the learned orthonormal bases $\{\mathbf{D}_\ell\}_{\ell=1}^L$ into $\mathbf{D}$. In this set of experiments, we use MiCUSaL and rMiCUSaL for complete and missing data experiments, respectively. The number of random initializations used to select the final geometric structure in these experiments is 8 for every fixed $\mathbf{Y}$ and $\lambda$. We use the following metrics for performance analysis of MC-UoS learning. Since we have knowledge of the ground truth $\mathcal{S}_\ell$'s, represented by their ground truth orthonormal bases $\mathbf{T}_\ell$'s, we first find the pairs of estimated and true subspaces that are the best match, i.e., $\mathbf{D}_\ell$ is matched to $\mathbf{T}_{\widehat{\ell}}$ using $\widehat{\ell} = \arg\max_p \|\mathbf{D}_\ell^T \mathbf{T}_p\|_F$. We also ensure that no two $\mathbf{D}_\ell$'s are matched to the same $\mathbf{T}_{\widehat{\ell}}$. Then we define $d_{avg}$ to be the *average normalized subspace distances* between these pairs, i.e., $d_{avg} = \frac{1}{L}\sum_{\ell=1}^L \sqrt{\frac{s-\mathrm{tr}(\mathbf{D}_\ell^T \mathbf{T}_{\widehat{\ell}} \mathbf{T}_{\widehat{\ell}}^T \mathbf{D}_\ell)}{s}}$. A smaller $d_{avg}$ indicates better performance of MC-UoS learning. Also, if the learned subspaces are close to the ground truth, they are expected to have good representation performance on test data. A good measure in this regard is the *mean of relative reconstruction errors of the test samples* using learned subspaces. To be specific, if the training data are complete, we first represent every test signal $\mathbf{z}$ such that $\mathbf{z} \approx \mathbf{D}_\tau \boldsymbol{\alpha}^{te} + \bar{\mathbf{y}}$ where $\tau = \arg\max_\ell \|\mathbf{D}_\ell^T(\mathbf{z} - \bar{\mathbf{y}})\|_2^2$ (recall that $\bar{\mathbf{y}} = \frac{1}{N}\sum_{i=1}^N \mathbf{y}_i$) and $\boldsymbol{\alpha}^{te} = \mathbf{D}_\tau^T(\mathbf{z} - \bar{\mathbf{y}})$. The relative reconstruction error with respect to its noiseless part, $\mathbf{x}$, is then defined as $\frac{\|\mathbf{x} - (\mathbf{D}_\tau \boldsymbol{\alpha}^{te} + \bar{\mathbf{y}})\|_2^2}{\|\mathbf{x}\|_2^2}$. On the other hand, if the training data have missing entries then for a test signal $\mathbf{z}$, the reconstruction error with respect to $\mathbf{x}$ is simply calculated by $\frac{\|\mathbf{x} - \mathbf{D}_\tau \mathbf{D}_\tau^T \mathbf{z}\|_2^2}{\|\mathbf{x}\|_2^2}$, where $\tau = \arg\max_\ell \|\mathbf{D}_\ell^T \mathbf{z}\|_2^2$.

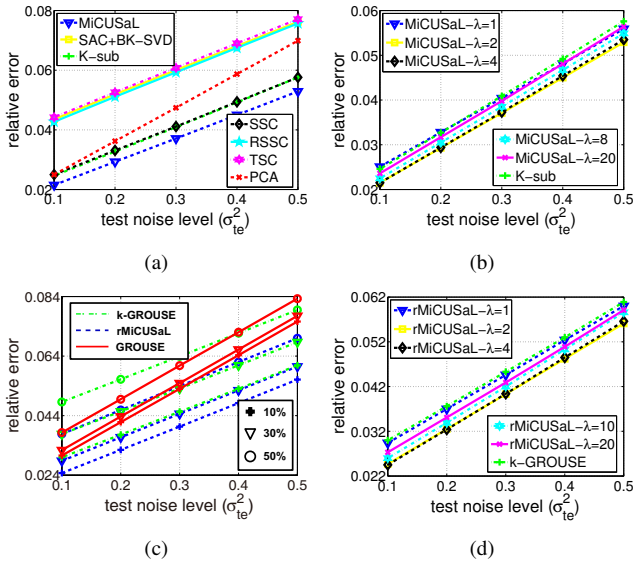



Fig. 2. Comparison of MC-UoS learning performance on synthetic data. (a) and (c) show relative errors of test signals for complete and missing data experiments, where $\lambda = 2$ for both MiCUSaL and rMiCUSaL. The numbers in the legend of (c) indicate the percentages of missing entries within the training data. (b) and (d) show relative errors of test signals for MiCUSaL and rMiCUSaL (with $10\%$ missing entries) using different $\lambda$'s.

To compare with other UoS learning methods, we choose $\lambda = 2$ for both MiCUSaL and rMiCUSaL. In the complete data experiments, we perform SSC with $\alpha_z = 60$. We set the subspace dimension for PCA to be the (unrealizable, ideal) one that yields the best denoising result on *training* samples. We also use the same subspace dimension (again, unrealizable and ideal) for GROUSE in the corresponding missing data experiments. Table I summarizes the $d_{avg}$'s of different UoS learning algorithms for both complete and missing data experiments. As can be seen, MiCUSaL produces smaller $d_{avg}$'s, which in turn leads to smaller relative errors of test data; see Fig. 2(a) for a validation of this claim. For MC-UoS learning with missing data, rMiCUSaL also learns

TABLE I
$d_{avg}$ OF DIFFERENT UOS LEARNING ALGORITHMS FOR SYNTHETIC DATA

| $d_{avg}$ | Algorithms | | | | | |
|---|---|---|---|---|---|---|
| Complete | MiCUSaL($\lambda = 2$) | SAC+BK-SVD | $K$-sub | SSC | RSSC | TSC |
| | **0.1331** | 0.2187 | 0.1612 | 0.1386 | 0.2215 | 0.2275 |
| Missing | rMiCUSaL-10% | kGROUSE-10% | rMiCUSaL-30% | kGROUSE-30% | rMiCUSaL-50% | kGROUSE-50% |
| | **0.1661** | 0.3836 | **0.1788** | 0.4168 | **0.2047** | 0.4649 |

a better MC-UoS in that: ($i$) for a fixed percentage of the number of missing observations in the training data, the $d_{avg}$ for rMiCUSaL is much smaller than the one for $k$-GROUSE (see Table I); and ($ii$) rMiCUSaL outperforms $k$-GROUSE and GROUSE in terms of smaller reconstruction errors of test data. Moreover, we can infer from Fig. 2(c) that for a fixed $\sigma_{te}$, when the number of missing entries increases, the performance of rMiCUSaL degrades less compared to $k$-GROUSE. We also test the UoS learning performance with complete data when the subspaces are not close to each other (e.g., $t_s = 0.2$). In this case, all the UoS learning algorithms, including MiCUSaL, learn the subspaces successfully. We omit these plots because of space constraints.

TABLE II
$d_{avg}$ OF MICUSAL AND RMICUSAL FOR DIFFERENT $\lambda$'S USING SYNTHETIC DATA

| $d_{avg}$ | $\lambda$ | | | | |
|---|---|---|---|---|---|
| MiCUSaL | $\lambda = 1$ | $\lambda = 2$ | $\lambda = 4$ | $\lambda = 8$ | $\lambda = 20$ |
| | 0.1552 | 0.1331 | **0.1321** | 0.1378 | 0.1493 |
| rMiCUSaL (10% missing) | $\lambda = 1$ | $\lambda = 2$ | $\lambda = 4$ | $\lambda = 10$ | $\lambda = 20$ |
| | 0.2096 | **0.1661** | 0.1725 | 0.2065 | 0.2591 |

For both MiCUSaL and rMiCUSaL, we also analyze the effect of the key parameter, $\lambda$, on the UoS learning performance. We implement MiCUSaL with $\lambda \in \{1, 2, 4, 8, 20\}$ in the complete data experiments and select $\lambda \in \{1, 2, 4, 10, 20\}$ for rMiCUSaL in the missing data experiments, where the number of missing entries in the training data is $10\%$ of the signal dimension. The results are shown in Fig. 2(b), Fig. 2(d) and Table II. We can see when $\lambda = 1$, both the $d_{avg}$'s and reconstruction errors of the test data are large for MiCUSaL and rMiCUSaL. This is because the learned subspaces are too close to each other, which results in poor data representation capability of the learned $\mathbf{D}_\ell$'s. When $\lambda = 2$ or $4$, both these algorithms achieve good performance in terms of small $d_{avg}$'s and relative errors of test data. As $\lambda$ increases further, both $d_{avg}$ and relative errors of test data also increase. Furthermore, as $\lambda$ grows, the curves of relative errors of test data for MiCUSaL and rMiCUSaL get closer to the ones for $K$-sub and $k$-GROUSE, respectively. This phenomenon coincides with our discussion in Sec. III. Finally, we note that both MiCUSaL and rMiCUSaL achieve their best performance when $\lambda \in [2, 4]$, and deviations of the representation errors of test data are very small when $\lambda$ falls in this range.

Next, we study the effect of random initialization of subspaces on MiCUSaL performance by calculating the standard deviation of the mean of the reconstruction errors of the test data for the $8$ random initializations. The mean of these $200$ standard deviations ends up being only $0.003$ for all $\sigma_{te}$'s when $\lambda = 2$. In addition, as $\lambda$ gets larger, the variation of the results increases only slightly (the mean of the standard deviations is $0.0034$ for $\lambda = 8$). On the other hand, the mean of the standard deviations for $K$-sub is $0.0039$. Furthermore, the performance gaps between MiCUSaL and all other methods are larger than $0.003$. Finally, the learned MC-UoS structure that results in the smallest value of the objective function (2) always results in the best denoising performance. This suggests that MiCUSaL always generates the best results and it is mostly insensitive to the choice of initial subspaces during the random initialization.

(a) (b)

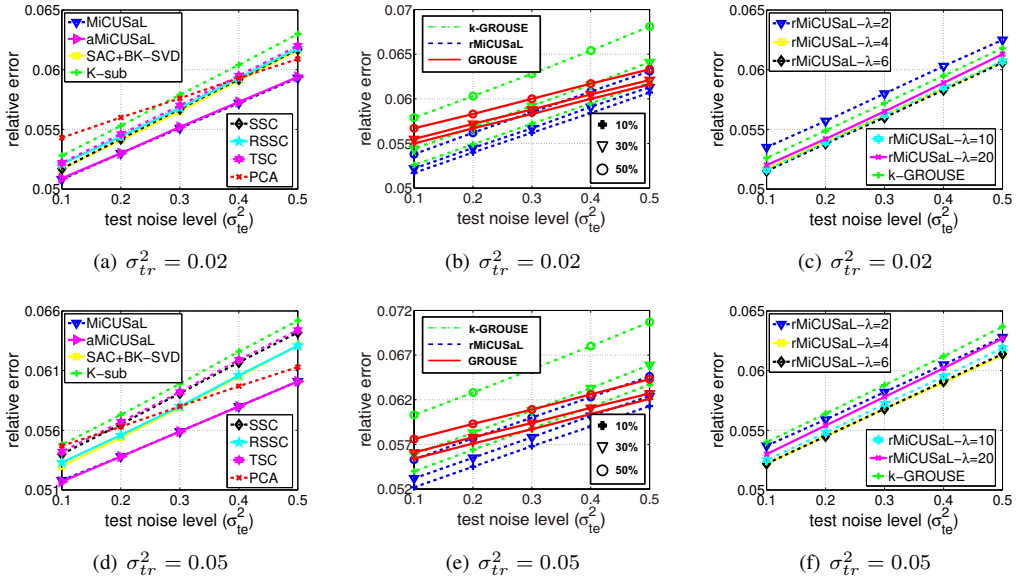
Fig. 3. (a) San Francisco City Hall image. (b) Paris City Hall image.

We also examine the running times of rMiCUSaL and $k$-GROUSE per iteration, which include both the subspace assignment and subspace update stages. For each subspace $\mathcal{S}_\ell$, we implement the optimization over $\mathbf{D}_\ell$ (i.e., Steps $8$ to $17$ in Algorithm 3) for $100$ iterations. All experiments are carried out using Matlab R2013a on an Intel i7-2600 3.4GHz CPU with 16 GB RAM. From the fourth row of Table III, we observe rMiCUSaL takes slightly more time compared to $k$-GROUSE because rMiCUSaL needs two more steps for updating $\mathbf{D}_\ell$ (Steps $10$ and $11$ in Algorithm 3). However, the advantage of rMiCUSaL over $k$-GROUSE in learning a better UoS significantly outweighs this slight increase in computational complexity. We can also see that as the number of missing entries increases, both algorithms become faster. The reason for this is that when $|\mathbf{\Omega}_i|$ decreases for all $i$'s, less time is needed during the subspace assignment step and for computing $\boldsymbol{\theta}$ and $\mathbf{r}$ in Algorithm 3.

*2) Experiments on City Scene Data:* To further show the effectiveness of the proposed approaches, we test our proposed methods on real-world city scene data. First, we study the performance of our methods on San Francisco City Hall image, as shown in Fig. 3(a). To generate the clean training and test data, we split the image into left and right subimages of equal size. Then we extract all $30 \times 20$ nonoverlapping image patches from the left subimage and reshape them into $N = 722$ column vectors of dimension $m = 600$. All these vectors are normalized to have unit $\ell_2$ norms and are then used as signals in $\mathbf{X}$. Test signals in $\mathbf{X}^{te} \in \mathbb{R}^{600\times 722}$ are extracted in the same way from the right subimage. White Gaussian

TABLE III
RUNNING TIME COMPARISON (IN SEC) FOR RMICUSAL AND $k$-GROUSE

| Data | rMiCUSaL | | | $k$-GROUSE | | |
|---|---|---|---|---|---|---|
| Synthetic, $m = 180$, $N = 650$, $L = 5$, $s = 13$ | Missing entries (%) | | | Missing entries (%) | | |
| | 10% | 30% | 50% | 10% | 30% | 50% |
| | 8.02 | 7.41 | 6.62 | 7.46 | 6.95 | 6.11 |
| San Francisco, $m = 600$, $N = 722$, $L = 5$, $s = 12$ | Missing entries (%) | | | Missing entries (%) | | |
| | 10% | 30% | 50% | 10% | 30% | 50% |
| | 23.19 | 22.46 | 20.31 | 16.56 | 14.75 | 12.53 |




(a) $\sigma_{tr}^2 = 0.02$ (b) $\sigma_{tr}^2 = 0.02$ (c) $\sigma_{tr}^2 = 0.02$

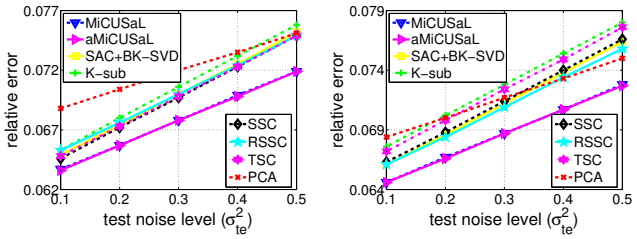
(d) $\sigma_{tr}^2 = 0.05$ (e) $\sigma_{tr}^2 = 0.05$ (f) $\sigma_{tr}^2 = 0.05$

Fig. 4. Comparison of MC-UoS learning performance on San Francisco City Hall data. (a) and (d) show relative errors of test signals for complete data experiments. (b) and (e) show relative errors of test signals for missing data experiments. The numbers in the legend of (b) and (e) indicate the percentages of missing entries within the training data. (c) and (f) show relative errors of test signals for rMiCUSaL (with 10% missing entries) using different $\lambda$'s.

noise is then added to $\mathbf{X}$ and $\mathbf{X}^{te}$ separately, forming $\mathbf{Y}$ and $\mathbf{Z}$, respectively. In these experiments, $\sigma_{tr}^2$ is set to be $0.02$ and $0.05$, while $\sigma_{te}^2$ again ranges from $0.1$ to $0.5$. The Monte Carlo simulations for noisy data are repeated $50$ times and the results reported here correspond to the average of these $50$ trials. Note that each patch is treated as a single signal here, and our goal is to learn an MC-UoS from $\mathbf{Y}$ such that every test patch can be reliably denoised using the learned subspaces.

We perform aMiCUSaL on the training data $\mathbf{Y}$ with parameters $L_{max} = 8$, $s_{max} = 20$, $\lambda = 4$, $k_1 = 6$, $k_2 = 10$ and $\epsilon_{min} = 0.08$. The number of random initializations that are used to arrive at the final MC-UoS structure using aMiCUSaL is $10$ for every fixed $\mathbf{Y}$. The output $L$ from aMiCUSaL is $4$ or $5$ and $s$ is always between $11$ or $13$. We also perform MiCUSaL with the same $L$ and $s$ $10$ times. For fair comparison, we also use the method in this paper to get the dimension of the subspace for PCA, in which case the estimated $s$ is always $10$. Note that for all state-of-the-art UoS learning algorithms, we use the same $L$ and $s$ as aMiCUSaL instead of using the $L$ generated by the algorithms themselves. The reason for this is as follows. The returned $L$ by SSC (with $\alpha_z = 40$) is $1$. Therefore SSC reduces to PCA in this setting. The output $L$ for RSSC is also $4$ or $5$, which coincides with our algorithm. The estimation of $L$ (with $q = 2\max(3, \lceil N/(L \times 20) \rceil)$) for TSC is sensitive to the noise and data. Specifically, the estimated $L$ is always from $6$ to $9$ for $\sigma_{tr}^2 = 0.02$ and $L$ is always $1$ when $\sigma_{tr}^2 = 0.05$, which results in poorer performance compared to the case when $L = 4$ or $5$ for both training noise levels. In the missing data experiments, we set $L = 5$ and $s = 12$ for rMiCUSaL (with $\lambda = 4$) and $k$-GROUSE, and $s = 10$ for GROUSE. Fig. 4(a) and Fig. 4(d) describe the relative reconstruction errors of test samples when the training data are complete. We see both MiCUSaL and aMiCUSaL learn a better MC-UoS since they give rise to smaller relative errors of test data. Further, the average standard deviation of the mean of relative errors for test data is around $0.00015$ for MiCUSaL and $0.00045$ for $K$-sub. It can be inferred from Fig. 4(b) and Fig. 4(e) that rMiCUSaL also yields better data representation performance for the missing data case.

To examine the effect of $\lambda$ on the denoising result in both complete and missing data experiments, we first run aMiCUSaL with $\lambda \in \{1, 2, 4, 6, 8, 10\}$ without changing other parameters. When $\lambda = 1$ or $2$, aMiCUSaL always returns $L = 2$ or $3$ subspaces, but the reconstruction errors of the test data are slightly larger than those for $\lambda = 4$. When $\lambda \geq 6$, the distances between the learned subspaces become larger, and the resulting $L$ will be at least $6$ when $\epsilon_{min}$ is fixed. However, the relative errors of test data are still very close to the ones for $\lambda = 4$. This suggests that $\lambda = 4$ is a good choice in this setting since it leads to the smallest number of subspaces $L$ and the best representation performance. We also perform rMiCUSaL



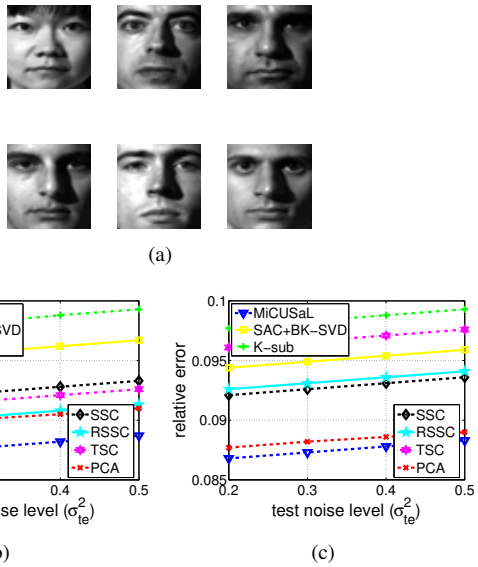

(a) $\sigma^2_{tr} = 0.02$ (b) $\sigma^2_{tr} = 0.05$

Fig. 5. Comparison of MC-UoS learning performance on Paris City Hall data when the training data are complete.

with $\lambda \in \{2, 4, 6, 10, 20\}$ while keeping $L$ and $s$ fixed, where the number of missing entries in the training data is again $10\%$ of the signal dimension. We show the relative errors of test data in Fig. 4(c) and Fig. 4(f). Similar to the results of the experiments with synthetic data, we again observe the fact that when $\lambda$ is small (e.g., $\lambda = 2$), the reconstruction errors of the test data are large because the subspace closeness metric dominates in learning the UoS. The results for $\lambda = 4$ and $6$ are very similar. As $\lambda$ increases further, the performance of rMiCUSaL gets closer to that of $k$-GROUSE. We again report the running time of rMiCUSaL and $k$-GROUSE per iteration in the seventh row of Table III, where we perform the optimization over each $\mathbf{D}_\ell$ for $100$ iterations in each subspace update step for both rMiCUSaL and $k$-GROUSE. In these experiments, rMiCUSaL appears much slower than $k$-GROUSE. However, as presented in Fig. 4(b) and Fig. 4(e), the performance of rMiCUSaL is significantly better than $k$-GROUSE.

Next, we repeat these experiments for the complete data experiments using Paris City Hall image in Fig. 3(b), forming $\mathbf{X}, \mathbf{X}^{te} \in \mathbb{R}^{600\times 950}$. We perform aMiCUSaL using the same parameters ($\lambda = 4$) as in the previous experiments. The estimated $L$ in this case is always between $5$ and $6$ and $s$ is always between $11$ and $12$. The estimated dimension of the subspace in PCA is $9$ or $10$ when $\sigma^2_{tr} = 0.02$ and it is always $10$ when $\sigma^2_{tr} = 0.05$. In these experiments, we again use the same $L$ and $s$ as aMiCUSaL for all state-of-the-art UoS learning algorithms. This is because the returned $L$ by SSC (with $\alpha_z = 20$) is again $1$ in this case. The estimated $L$ by RSSC is usually $7$ or $8$, and the reconstruction errors of test data are very close to the ones reported here. If we apply TSC using the $L$ estimated by itself (again, with $q = 2\max(3, \lceil N/(L \times 20)\rceil)$), we will have $L = 4$ when $\sigma^2_{tr} = 0.02$, while the relative errors of test data are very close to the results shown here. For $\sigma^2_{tr} = 0.05$, TSC will again result in only one subspace. The relative reconstruction errors of test data with different training noise levels are shown in Fig. 5, from which we make the conclusion that our methods obtain small errors, thereby outperforming all other algorithms. The average standard deviation of the mean of relative errors for test data is also smaller for MiCUSaL (around $0.00023$) compared to $K$-sub (around $0.00037$).

*3) Experiments on Face Dataset:* In this section, we work with the Extended Yale B dataset [48], which contains a set of $192 \times 168$ cropped images of $38$ subjects. For each individual,




(a)

(b) (c)

Fig. 6. Comparison of MC-UoS learning performance on Extended Yale B dataset. The first row of (a) shows some images of subject $5, 6, 8$ and the second row presents some images of subject $22, 28, 30$. (b) and (c) show relative errors of test data in the two experiments.

there are $64$ images taken under varying illumination conditions. We downsample the images to $48 \times 42$ pixels and each image is vectorized and treated as a signal; thus, $m = 2016$. It has been shown in [49] that the set of images of a given subject taken under varying illumination conditions can be well represented by a 9-dimensional subspace.

We first focus on a collection of images of subjects $5, 6$ and $8$ and normalize all the images to have unit $\ell_2$ norms. Some representative images are presented in the first row of Fig. 6(a). Here we assume the images of these three subjects lie close to an MC-UoS with $L = 3$ and $s = 9$. For each set of images from one subject, we randomly select half of them for training and the remaining $32$ images belong to test samples; therefore, $\mathbf{X}, \mathbf{X}^{te} \in \mathbb{R}^{2016\times 96}$. Then we add white Gaussian noise to both $\mathbf{X}$ and $\mathbf{X}^{te}$ and obtain $\mathbf{Y}$ and $\mathbf{Z}$. The random selection for generating $\mathbf{X}$ and $\mathbf{X}^{te}$ is repeated $10$ times and we conduct Monte Carlo simulations for noisy data $10$ times for every fixed $\mathbf{X}$ and $\mathbf{X}^{te}$. In these experiments, the value $\sigma^2_{tr}$ is equal to $0.2$ and $\sigma^2_{te}$ is from $0.2$ to $0.5$. For fair comparison, the dimension of the subspace for PCA is set to be $9$. We apply MiCUSaL with parameter $\lambda = 2$ and SSC with $\alpha_z = 40$. The number of random initializations in these experiments for both MiCUSaL and $K$-sub is set at $8$ for every fixed $\mathbf{Y}$. Once again, we observe that MiCUSaL outperforms other learning methods, since it results in smaller relative errors (cf. Fig. 6(b)). Moreover, the average standard deviation of MiCUSaL for the $100$ realizations of $\mathbf{Y}$ and $\mathbf{Z}$ is only $0.0013$ for all $\sigma_{te}$'s, which is again smaller than that of $K$-sub (the corresponding value is $0.002$).

We then repeat these experiments using a set of images of subjects $22, 28$ and $30$, and show some image samples in the second row of Fig. 6(a). We set $L = 3$, $s = 9$ and $\lambda = 2$ for MiCUSaL and $\alpha_z = 40$ for SSC. We again provide evidence in Fig. 6(c) that MiCUSaL yields better data representation performance in this setting. The average standard deviation of

the mean of the reconstruction errors for test data is around $0.0012$ for MiCUSaL and $0.0019$ for $K$-sub in this case.

### B. Experiments for MC-KUoS Learning

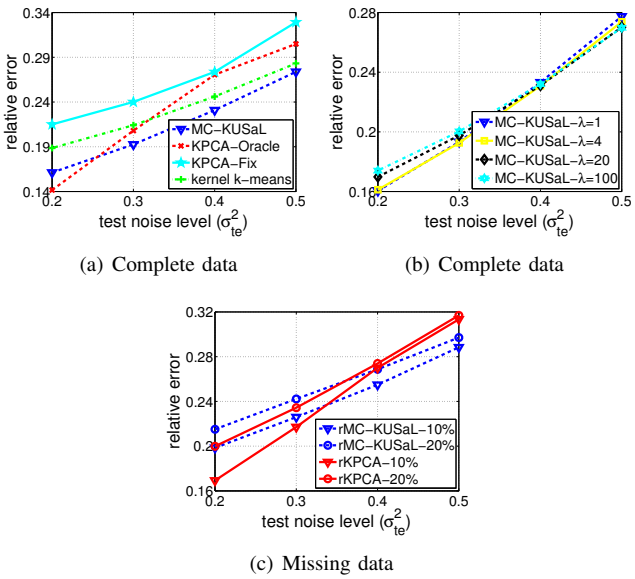
In this section, we evaluate the performance of the MC-KUoS learning approaches in terms of the following two problems: image denoising using the learned MC-KUoS and clustering of training data points. For both these problems, we consider the USPS dataset [65],[7] which contains a collection of $m = 256$-dimensional handwritten digits. The authors in [9] have demonstrated that using nonlinear features can improve the denoising performance of this dataset. Unlike the experiments for MC-UoS learning, the training data we use in this set of experiments are noiseless. For denoising experiments, we assume every noisy *test* sample $\mathbf{z} = \mathbf{x} + \boldsymbol{\xi}$, where $\widetilde{\phi}(\mathbf{x}) = \phi(\mathbf{x}) - \overline{\boldsymbol{\phi}}$ belongs to one of the $\mathcal{S}_\ell$'s in $\mathcal{F}$ (again $\|\mathbf{x}\|_2^2 = 1$) and $\boldsymbol{\xi}$ has $\mathcal{N}(\mathbf{0}, (\sigma_{te}^2/m)\mathbf{I}_m)$ distribution. In these experiments, $\sigma_{te}^2$ ranges from $0.2$ to $0.5$.

*1) Experiments on Image Denoising:* For denoising experiments, we compare the result of MC-KUSaL with three other methods: $(i)$ kernel $k$-means clustering (kernel $k$-means) [18], where for each test signal $\mathbf{z}$, we first assign $\phi(\mathbf{z})$ to a cluster whose centroid is closest to $\phi(\mathbf{z})$ in $\mathcal{F}$, followed by kernel PCA and the method in [64] to calculate the pre-image; $(ii)$ kernel PCA [29] with the same number of eigenvectors as in MC-KUSaL (KPCA-Fix); and $(iii)$ kernel PCA with the number of eigenvectors chosen by $s = \arg\min_s ||P_S\phi(\mathbf{z}) - \phi(\mathbf{x})||_2^2$ (KPCA-Oracle), where $\mathbf{x}$ and $\mathbf{z}$ are clean and noisy test samples, respectively. In this manner, the number of eigenvectors $s$ for KPCA-Oracle will be different for different noise levels $\sigma_{te}$'s. We use the same dimension of the subspaces for MC-KUSaL, kernel $k$-means clustering and KPCA-Fix, while the number of subspaces $L$ for kernel $k$-means clustering also equals the one for MC-KUSaL. For the case of missing training data, we report the results of rMC-KUSaL as well as rKPCA. For every fixed test noise level $\sigma_{te}$, we set the dimension of the subspace $s$ for rKPCA to be the same as the one for KPCA-Oracle. The relative reconstruction error of a clean test signal $\mathbf{x} \in \mathbf{X}^{te}$ is calculated by $\frac{\|\mathbf{x}-\widehat{\mathbf{z}}\|_2^2}{\|\mathbf{x}\|_2^2}$, where $\widehat{\mathbf{z}}$ denotes the pre-image with respect to the noisy test sample $\mathbf{z}$.

We experiment with Gaussian kernel with parameter $c = 4$. We choose the digits "0" and "4" and for each digit we select the first $200$ samples in the dataset ($400$ images in total) for our experiments. All these $400$ samples are then vectorized and normalized to unit $\ell_2$ norms. From these samples, we randomly choose $120$ samples (without replacement) from each class for training and the remaining $80$ samples of each class for testing, forming $\mathbf{X} \in \mathbb{R}^{256\times 240}$ and $\mathbf{X}^{te} \in \mathbb{R}^{256\times 160}$. This random selection of test and training samples is repeated $20$ times for cross-validation purposes. We perform $10$ Monte Carlo trials for noisy test data and report the mean over these $200$ random trials.

In these experiments, we implement MC-KUSaL with parameters $L = 2$, $s = 45$ and $\lambda \in \{1, 4, 20, 100\}$ to learn an MC-UoS in the feature space $\mathcal{F}$. Fig. 7(a) shows the mean of relative reconstruction errors of test data for different methods

[7] Available at: http://www.cs.nyu.edu/~roweis/data.html.




(a) Complete data (b) Complete data

(c) Missing data

Fig. 7. Comparison of MC-KUoS learning performance on USPS dataset using Gaussian kernel $\kappa(\mathbf{y}, \mathbf{y}') = \exp(-\frac{\|\mathbf{y}-\mathbf{y}'\|_2^2}{4})$. In (a), we perform MC-KUSaL with $\lambda = 4$. Note that the KPCA-Oracle algorithm is the ideal case of kernel PCA. The numbers in the legend of (c) indicate the percentages of missing entries within the training data.

in the presence of complete training data, where we use the result of MC-KUSaL with $\lambda = 4$ for comparison with other methods. We observe that for almost all noise levels, our method produces better results than other methods. The only exception is when $\sigma_{te}^2 = 0.2$, in which case MC-KUSaL is the second best of all methods. The caveat here is that in practice, we cannot know beforehand the dimension of the subspace in the feature space for kernel PCA, which yields the best denoising result at this particular noise level. We show the denoising performance of MC-KUSaL with different $\lambda$'s in Fig. 7(b), and we observe that a small $\lambda$ usually results in good performance when $\sigma_{te}^2$ is relatively small, while increasing the $\lambda$ will slightly improve the denoising performance when the SNR of test data gets small.

In the missing data experiments, we set the number of missing entries in the training data to be $10\%$ and $20\%$ of the signal dimension. We use parameters $L = 2$, $s = 45$ and $\lambda = 4$ for rMC-KUSaL. It can be inferred from Fig. 7(c) that $(i)$ the performance of rKPCA and rMC-KUSaL is comparable for all noise levels; and $(ii)$ when the number of missing elements is fixed, rMC-KUSaL outperforms the rKPCA when the SNR of the test data is small and vice versa.

*2) Experiments on Clustering:* In this section, we empirically compare the clustering performance of MC-KUSaL with $(i)$ kernel $k$-means clustering (kernel $k$-means) [18], $(ii)$ standard $k$-means clustering ($k$-means) [66], and $(iii)$ spectral clustering [67] when the training data are complete. In the case of spectral clustering, we make use of the similarity matrix returned by the noisy variant of the SSC optimization program in [16]. We also present the clustering performance of rMC-KUSaL, with the number of missing entries in the training data being set to $10\%$ and $20\%$ of the signal dimension. We compute the clustering error for MC-KUSaL/rMC-KUSaL by

TABLE IV
CLUSTERING ERROR (%) ON THE USPS DATASET

| Digits | Kernel Function | Algorithms | | | | | |
|---|---|---|---|---|---|---|---|
| | | MC-KUSaL | kernel $k$-means | $k$-means | SSC | rMC-KUSaL(10%) | rMC-KUSaL(20%) |
| 1,7 | $\kappa(\mathbf{y},\mathbf{y}')=\exp(-\frac{\|\mathbf{y}-\mathbf{y}'\|_2^2}{8})$ | **7.21** | 20.94 | 21.19 | 12.13 | 11.52 | 12.69 |
| | $\kappa(\mathbf{y},\mathbf{y}')=(\langle\mathbf{y},\mathbf{y}'\rangle+2)^3$ | **8.23** | 20.54 | | | 10.60 | 11.85 |
| 1,6 | $\kappa(\mathbf{y},\mathbf{y}')=\exp(-\frac{\|\mathbf{y}-\mathbf{y}'\|_2^2}{4})$ | **5.00** | 11.04 | 11.29 | 8.71 | 6.27 | 7.88 |
| | $\kappa(\mathbf{y},\mathbf{y}')=(\langle\mathbf{y},\mathbf{y}'\rangle+1)^3$ | **4.85** | 10.60 | | | 7.54 | 8.04 |

using the final kernel subspace assignment labels $\{l_i\}_{i=1}^N$. For all the following experiments, we select $L=2$ (since we only have 2 classes) and $\lambda=200$.

We first experiment with digits "1" and "7" in the USPS dataset, where in every trial we randomly choose 120 $\ell_2$ normalized samples from the first 200 samples of these two digits and use these 240 samples in these experiments. This random selection is repeated 20 times. We perform MC-KUSaL and rMC-KUSaL using Gaussian kernel $\kappa(\mathbf{y},\mathbf{y}')=\exp(-\frac{\|\mathbf{y}-\mathbf{y}'\|_2^2}{8})$ with $s=35$ and polynomial kernel $\kappa(\mathbf{y},\mathbf{y}')=(\langle\mathbf{y},\mathbf{y}'\rangle+2)^3$ with $s=40$. The parameter $\alpha_z$ for SSC is set to be 20. The clustering results are listed in Table IV, where we can see the clustering error for MC-KUSaL is roughly $40\%$ of the ones for kernel/standard $k$-means clustering and MC-KUSaL is much better than SSC (with $32\%$ reduction) in these experiments. In addition, the clustering error for rMC-KUSaL is an increasing function of the number of missing entries for both Gaussian kernel and polynomial kernel.

As another example, we repeat the above experiments using digits "1" and "6", where we again apply MC-KUSaL and rMC-KUSaL using Gaussian kernel $\kappa(\mathbf{y},\mathbf{y}')=\exp(-\frac{\|\mathbf{y}-\mathbf{y}'\|_2^2}{4})$ with $s=35$ and polynomial kernel $\kappa(\mathbf{y},\mathbf{y}')=(\langle\mathbf{y},\mathbf{y}'\rangle+1)^3$ with $s=40$. In these experiments, SSC is performed with $\alpha_z=10$. From Table IV, we again observe that MC-KUSaL outperforms other clustering algorithms with $42\%$ reduction (compared to SSC) in the clustering error. In the missing data experiments, the clustering performance of rMC-KUSaL using Gaussian kernel degrades as the number of missing entries of the data increases. When we use polynomial kernel for rMC-KUSaL, increasing the number of entries in the missing data does not result in much degradation of the clustering performance.

We conclude by noting that the choice of kernels in these experiments is agnostic to the training data. Nonetheless, data-driven learning of kernels is an active area of research, which is sometimes studied under the rubric of *multiple kernel learning* [68]–[71]. While some of these works can be leveraged to further improve the performance of our proposed algorithms, a careful investigation of this is beyond the scope of this work.

## VI. CONCLUSION

In this paper, we proposed a novel extension of the canonical union-of-subspaces model, termed the metric-constrained union-of-subspaces (MC-UoS) model. We first proposed several efficient iterative approaches for learning of an MC-UoS in the ambient space using both complete and missing data. Moreover, these methods are extended to the case of a higher-dimensional feature space such that one can deal with MC-UoS learning problem in the feature space using complete and missing data. Experiments on both synthetic and real data showed the effectiveness of our algorithms and their superiority over the state-of-the-art union-of-subspaces learning algorithms. Our future work includes estimation of the number and dimension of the subspaces from the training data for MC-UoS learning in the feature space.

## APPENDIX
## PROOF OF LEMMA 1

*Proof:* First, we have $\langle\mathbf{y}_i,\mathbf{y}_j\rangle=\sum_{u=1}^m\mathbf{z}^*_{ij\,(u)}$ and $\langle[\mathbf{y}_i]_{\mathbf{\Omega}_{ij}},[\mathbf{y}_j]_{\mathbf{\Omega}_{ij}}\rangle=\sum_{v=1}^n\mathbf{z}^*_{ij\,(\mathbf{\Omega}_{ij\,(v)})}$ with $n=|\mathbf{\Omega}_{ij}|$. Here, $\mathbf{z}^*_{ij\,(u)}$ denotes the $u$-th entry of a vector $\mathbf{z}^*_{ij}$ and $\mathbf{\Omega}_{ij\,(v)}$ denotes the $v$-th element of $\mathbf{\Omega}_{ij}$. Let $\hbar(Z_1,\ldots,Z_n)=\sum_{v=1}^n Z_v$ be the sum of $n$ random variables and $Z_v=\mathbf{z}^*_{ij\,(\mathbf{\Omega}_{ij\,(v)})}$. We prove the bound under the assumption that these $n$ variables are drawn uniformly from a set $\{\mathbf{z}^*_{ij\,(1)},\ldots,\mathbf{z}^*_{ij\,(m)}\}$ with replacement. This means they are independent and we have $\mathbb{E}[\sum_{v=1}^n Z_v]=\mathbb{E}[\sum_{v=1}^n\mathbf{z}^*_{ij\,(\mathbf{\Omega}_{ij\,(v)})}]=\frac{n}{m}\sum_{u=1}^m\mathbf{z}^*_{ij\,(u)}$. If the value of one variable in the sum is replaced by any other of its possible values, the sum changes at most $2\|\mathbf{z}^*_{ij}\|_\infty$, i.e., $|\sum_{v=1}^n Z_v-\sum_{v\neq v'}Z_v-\widehat{Z}_{v'}|=|Z_{v'}-\widehat{Z}_{v'}|\leq 2\|\mathbf{z}^*_{ij}\|_\infty$ for any $v'\in\{1,\ldots,n\}$. Therefore, McDiarmid's Inequality [72] implies that for $\beta>0$,

$$\mathbb{P}\Big[|\sum_{v=1}^n Z_v-\frac{n}{m}\sum_{u=1}^m\mathbf{z}^*_{ij\,(u)}|\geq\frac{n}{m}\beta\Big]\leq 2\exp\Big(\frac{-n\beta^2}{2m^2\|\mathbf{z}^*_{ij}\|_\infty^2}\Big),$$

or equivalently,

$$\mathbb{P}\Big[\sum_{u=1}^m\mathbf{z}^*_{ij\,(u)}-\beta\leq\frac{m}{n}\sum_{v=1}^n Z_v\leq\sum_{u=1}^m\mathbf{z}^*_{ij\,(u)}+\beta\Big]\geq 1-2\exp\Big(\frac{-n\beta^2}{2m^2\|\mathbf{z}^*_{ij}\|_\infty^2}\Big).$$

Taking the definition of $\beta=\sqrt{\frac{2m^2\|\mathbf{z}^*_{ij}\|_\infty^2}{|\mathbf{\Omega}_{ij}|}\log(\frac{1}{\delta})}$ yields the result. ∎

## REFERENCES


[1] T. Wu and W. U. Bajwa, "Revisiting robustness of the union-of-subspaces model for data-adaptive learning of nonlinear signal models," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 3390–3394.

[2] ——, "Subspace detection in a kernel space: The missing data case," in *Proc. IEEE Statistical Signal Processing Workshop (SSP)*, 2014, pp. 93–96.

[3] ——, "Metric-constrained kernel union of subspaces," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[4] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *J. Educ. Psych.*, vol. 24, pp. 417–441, 498–520, 1933.

[5] T. F. Cox and M. A. A. Cox, *Multidimensional scaling*. Chapman & Hall, 2000.

[6] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[7] M. Elad, R. Goldenberg, and R. Kimmel, "Low bit-rate compression of facial images," *IEEE Trans. Image Process.*, vol. 16, no. 9, pp. 2379–2383, 2007.

[8] R. G. Baraniuk, V. Cevher, and M. B. Wakin, "Low-dimensional models for dimensionality reduction and signal recovery: A geometric perspective," *Proc. IEEE*, vol. 98, no. 6, pp. 959–971, 2010.

[9] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, "Kernel PCA and de-noising in feature spaces," in *Advances in Neural Information Processing Systems (NIPS)*, 1998, pp. 536–542.

[10] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, 2006.

[11] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 1991, pp. 586–591.

[12] D. L. Swets and J. Weng, "Using discriminant eigenfeatures for image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 8, pp. 831–836, 1996.

[13] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, 2009.

[14] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 791–804, 2012.

[15] S. Rao, R. Tron, R. Vidal, and Y. Ma, "Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 10, pp. 1832–1845, 2010.

[16] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2765–2781, 2013.

[17] H. H. Harman, *Modern factor analysis*. University of Chicago Press, 1976.

[18] B. Schölkopf, A. J. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, 1998.

[19] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, 2006.

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

[21] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek, "An overview of JPEG-2000," in *Proc. IEEE Data Compression Conf. (DCC)*, 2000, pp. 523–541.

[22] J.-L. Starck, E. J. Candès, and D. L. Donoho, "The curvelet transform for image denoising," *IEEE Trans. Image Process.*, vol. 11, no. 6, pp. 670–684, 2002.

[23] T. Zhang, A. Szlam, and G. Lerman, "Median K-flats for hybrid linear modeling with many outliers," in *Proc. IEEE Int. Conf. Computer Vision Workshops*, 2009, pp. 234–241.

[24] B. V. Gowreesunker and A. H. Tewfik, "Learning sparse representation using iterative subspace identification," *IEEE Trans. Signal Process.*, vol. 58, no. 6, pp. 3055–3065, 2010.

[25] L. Balzano, A. Szlam, B. Recht, and R. Nowak, "K-subspaces with missing data," in *Proc. IEEE Statistical Signal Processing Workshop (SSP)*, 2012, pp. 612–615.

[26] W. Hong, J. Wright, K. Huang, and Y. Ma, "Multiscale hybrid linear models for lossy image representation," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3655–3671, 2006.

[27] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philos. Mag.*, vol. 2, no. 6, pp. 559–572, 1901.

[28] K. Fukunaga, *Introduction to statistical pattern recognition*. Academic Press, 1990.

[29] B. Schölkopf, A. J. Smola, and K.-R. Müller, "Kernel principal component analysis," *Advances in kernel methods: support vector learning*, pp. 327–352, 1999.

[30] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, 2003.

[31] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf, "A kernel view of the dimensionality reduction of manifolds," in *Proc. Int. Conf. Machine Learning (ICML)*, 2004, pp. 47–54.

[32] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (GPCA)," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1945–1959, 2005.

[33] L. Zelnik-Manor, K. Rosenblum, and Y. C. Eldar, "Dictionary optimization for block-sparse representations," *IEEE Trans. Signal Process.*, vol. 60, no. 5, pp. 2386–2395, 2012.

[34] M. Soltanolkotabi and E. J. Candès, "A geometric analysis of subspace clustering with outliers," *Ann. Stat.*, vol. 40, no. 4, pp. 2195–2238, 2012.

[35] E. L. Dyer, A. C. Sankaranarayanan, and R. G. Baraniuk, "Greedy feature selection for subspace clustering," *J. Mach. Learn. Res.*, vol. 14, pp. 2487–2517, 2013.

[36] R. Heckel and H. Bölcskei, "Robust subspace clustering via thresholding," *arXiv:1307.4891*, 2013.

[37] M. Soltanolkotabi, E. Elhamifar, and E. J. Candès, "Robust subspace clustering," *Ann. Stat.*, vol. 42, no. 2, pp. 669–699, 2014.

[38] T. Zhang, A. Szlam, Y. Wang, and G. Lerman, "Hybrid linear modeling via local best-fit flats," *Int. J. Comput. Vis.*, vol. 100, no. 3, pp. 217–240, 2012.

[39] Z. Ghahramani and G. E. Hinton, "The EM algorithm for mixtures of factor analyzers," CRG-TR-96-1, University of Toronto, Tech. Rep., 1997.

[40] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proc. IEEE*, vol. 98, no. 6, pp. 1031–1044, 2010.

[41] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman, "Clustering appearances of objects under varying illumination conditions," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2003, pp. 11–18.

[42] Y. M. Lu and M. N. Do, "A theory for sampling signals from a union of subspaces," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2334–2345, 2008.

[43] L. Wang, X. Wang, and J. Feng, "Subspace distance analysis with application to adaptive Bayesian algorithm for face recognition," *Pattern Recognition*, vol. 39, no. 3, pp. 456–464, 2006.

[44] X. Sun, L. Wang, and J. Feng, "Further results on the subspace distance," *Pattern Recognition*, vol. 40, no. 1, pp. 328–329, 2007.

[45] L. Wolf and A. Shashua, "Kernel principal angles for classification machines with applications to image sequence interpretation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2003, pp. 635–640.

[46] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. Int. Joint Conf. Artificial Intelligence (IJCAI)*, 1995, pp. 1137–1143.

[47] K. Pelckmans, J. De Brabanter, J. A. K. Suykens, and B. De Moor, "Convex clustering shrinkage," in *Statistics and Optimization of Clustering Workshop (PASCAL)*, 2005.

[48] K.-C. Lee, J. Ho, and D. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 684–698, 2005.

[49] R. Basri and D. W. Jacobs, "Lambertian reflectance and linear subspaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 2, pp. 218–233, 2003.

[50] J. T.-Y. Kwok and I. W.-H. Tsang, "The pre-image problem in kernel methods," *IEEE Trans. Neural Netw.*, vol. 15, no. 6, pp. 1517–1525, 2004.

[51] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.

[52] J. C. Bezdek and R. J. Hathaway, "Convergence of alternating optimization," *Neural Parallel Sci. Comput.*, vol. 11, no. 4, pp. 351–368, 2003.

[53] E. Kokiopoulou, J. Chen, and Y. Saad, "Trace optimization and eigenproblems in dimension reduction methods," *Numer. Linear Algebra Appl.*, vol. 18, no. 3, pp. 565–602, 2011.

[54] E. Levina and P. J. Bickel, "Maximum likelihood estimation of intrinsic dimension," in *Advances in Neural Information Processing Systems (NIPS)*, 2004, pp. 777–784.

[55] J. M. Bioucas-Dias and J. M. P. Nascimento, "Hyperspectral subspace identification," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 8, pp. 2435–2445, 2008.

[56] S. Kritchman and B. Nadler, "Determining the number of components in a factor model from limited noisy data," *Chemometr. Intell. Lab. Syst.*, vol. 94, no. 1, pp. 19–32, 2008.

[57] P. O. Perry and P. J. Wolfe, "Minimax rank estimation for subspace tracking," *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 3, pp. 504–513, 2010.

[58] L. Balzano, B. Recht, and R. Nowak, "High-dimensional matched subspace detection when data are missing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2010, pp. 1638–1642.

[59] A. Edelman, T. A. Arias, and S. T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM J. Matrix Anal. Appl.*, vol. 20, no. 2, pp. 303–353, 1998.
[60] A. Nedić and D. P. Bertsekas, "Incremental subgradient methods for nondifferentiable optimization," *SIAM J. Optim.*, vol. 12, no. 1, pp. 109–138, 2001.
[61] L. Balzano, R. Nowak, and B. Recht, "Online identification and tracking of subspaces from highly incomplete information," in *Proc. Allerton Conf. Communication, Control, and Computing*, 2010, pp. 704–711.
[62] B. Eriksson, L. Balzano, and R. Nowak, "High-rank matrix completion," in *Proc. Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2012, pp. 373–381.
[63] C. J. C. Burges, "Simplified support vector decision rules," in *Proc. Int. Conf. Machine Learning (ICML)*, 1996, pp. 71–77.
[64] Y. Rathi, S. Dambreville, and A. Tannenbaum, "Statistical shape analysis using kernel PCA," in *Proc. SPIE*, vol. 6064, 2006.
[65] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, 1994.
[66] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, 1982.
[67] U. von Luxburg, "A tutorial on spectral clustering," *Stat. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.
[68] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," in *Proc. Int. Conf. Machine Learning (ICML)*, 2004, pp. 41–48.
[69] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Mach. Learn. Res.*, vol. 5, pp. 27–72, 2004.
[70] C. Cortes, M. Mohri, and A. Rostamizadeh, "Learning non-linear combinations of kernels," in *Advances in Neural Information Processing Systems (NIPS)*, 2009, pp. 396–404.
[71] M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien, "$\ell_p$-norm multiple kernel learning," *J. Mach. Learn. Res.*, vol. 12, pp. 953–997, 2011.
[72] C. McDiarmid, "On the method of bounded differences," *Surveys in Combinatorics*, vol. 141, pp. 148–188, 1989.

**Tong Wu** received BE degree in Instrument Science and Engineering from Shanghai Jiao Tong University, China in 2009, and MS degree in Electrical Engineering from Duke University in 2011. Since September 2012, he has been working towards the PhD degree at the Department of Electrical and Computer Engineering, Rutgers, The State University of New Jersey. His research interests include high-dimensional data analysis, statistical signal processing, image and video processing, and machine learning.

**Waheed U. Bajwa** received BE (with Honors) degree in electrical engineering from the National University of Sciences and Technology, Pakistan in 2001, and MS and PhD degrees in electrical engineering from the University of Wisconsin-Madison in 2005 and 2009, respectively. He was a Postdoctoral Research Associate in the Program in Applied and Computational Mathematics at Princeton University from 2009 to 2010, and a Research Scientist in the Department of Electrical and Computer Engineering at Duke University from 2010 to 2011. He is currently an Assistant Professor in the Department of Electrical and Computer Engineering at Rutgers University. His research interests include harmonic analysis, high-dimensional statistics, machine learning, statistical signal processing, and wireless communications.

Dr. Bajwa has more than three years of industry experience, including a summer position at GE Global Research, Niskayuna, NY. He received the Best in Academics Gold Medal and President's Gold Medal in Electrical Engineering from the National University of Sciences and Technology in 2001, the Morgridge Distinguished Graduate Fellowship from the University of Wisconsin-Madison in 2003, the Army Research Office Young Investigator Award in 2014, and the National Science Foundation CAREER Award in 2015. He co-guest edited a special issue of Elsevier Physical Communication Journal on "Compressive Sensing in Communications" (2012), and co-chaired CPSWeek 2013 Workshop on Signal Processing Advances in Sensor Networks and IEEE GlobalSIP 2013 Symposium on New Sensing and Statistical Inference Methods. He currently serves as the Publicity and Publications Chair of IEEE CAMSAP 2015, and is an Associate Editor of the IEEE Signal Processing Letters.